OpenSSL PKI Tutorial

Release v2.0

Stefan H. Holek

Jul 26, 2021

CONTENTS

Create and operate Public Key Infrastructures with OpenSSL.

CHAPTER

ONE

OVERVIEW

This tutorial shows how to implement real-world PKIs with the OpenSSL toolkit.

In the first part of the tutorial we introduce the necessary terms and concepts. The second part consists of examples, where we build increasingly more sophisticated PKIs using nothing but the `openssl` utility. The tutorial puts a special focus on configuration files, which are key to taming the `openssl` command line. It also serves to promote what we have found to be the most effective way of partinioning the configuration space:

- One configuration file per CA, and
- One configuration file per CSR type.

Please study the configuration files included in the examples, it's where most of the treasure is buried.

CHAPTER

TWO

DEVELOPMENT CONTINUATION

This project is a continuation of the existing *OpenSSL PKI Tutorial v1.1 <https://pki-tutorial.readthedocs.io/>*

Changes - Protocol to differenciate versions:

- Move from bitbucket to github as collaboration platform
- Updating layout to new default template which enhances collaboration
- Updated defaults to SHA256 and 4096bit

CHAPTER

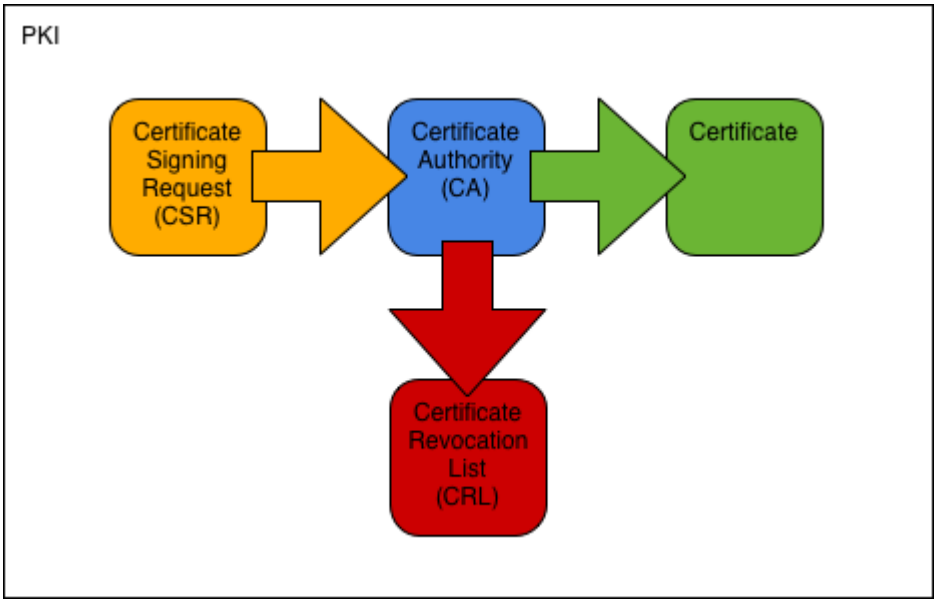
THREE

PKI CONCEPTS

At its heart, an X.509 PKI is a security architecture that uses well-established cryptographic mechanisms to support use-cases like email protection and web server authentication. In this regard it is similar to other systems based on public-key cryptography, for example OpenPGP [RFC 4880]. In the realm of X.509 however, and thanks to its roots in a globe-spanning scheme devised by the telecom industry, these mechanisms come with a fair amount of administrative overhead.

One thing to keep in mind is that X.509 is not an application, but a specification upon which applications like Secure Multipurpose Internet Mail Extensions (S/MIME) and Transport Layer Security (TLS) are based. The building blocks are very generic and derive most of their meaning from the relations that exist/are established between them. It's called an infrastructure for a reason.

3.1 Process



1. A requestor generates a CSR and submits it to the CA.
2. The CA issues a certificate based on the CSR and returns it to the requestor.
3. Should the certificate at some point be revoked, the CA adds it to its CRL.

3.2 Components

Public Key Infrastructure (PKI) Security architecture where trust is conveyed through the signature of a trusted CA.

Certificate Authority (CA) Entity issuing certificates and CRLs.

Registration Authority (RA) Entity handling PKI enrollment. May be identical with the CA.

Certificate Public key and ID bound by a CA signature.

Certificate Signing Request (CSR) Request for certification. Contains public key and ID to be certified.

Certificate Revocation List (CRL) List of revoked certificates. Issued by a CA at regular intervals.

Certification Practice Statement (CPS) Document describing structure and processes of a CA.

3.3 CA Types

Root CA CA at the root of a PKI hierarchy. Issues only CA certificates.

Intermediate CA CA below the root CA but not a signing CA. Issues only CA certificates.

Signing CA CA at the bottom of a PKI hierarchy. Issues only user certificates.

3.4 Certificate Types

CA Certificate Certificate of a CA. Used to sign certificates and CRLs.

Root Certificate Self-signed CA certificate at the root of a PKI hierarchy. Serves as the PKI's trust anchor.

Cross Certificate CA certificate issued by a CA external to the primary PKI hierarchy. Used to connect two PKIs and thus usually comes in pairs.[1]

User Certificate End-user certificate issued for one or more purposes: email-protection, server-auth, client-auth, code-signing, etc. A user certificate cannot sign other certificates.

3.5 File Formats

Privacy Enhanced Mail (PEM) Text format. Base-64 encoded data with header and footer lines. Preferred format in OpenSSL and most software based on it (e.g. Apache mod_ssl, stunnel).

Distinguished Encoding Rules (DER) Binary format. Preferred format in Windows environments. Also the official format for Internet download of certificates and CRLs.

[1] The RFC classifies any CA-signs-CA scenario as cross-certification, to distinguish it from self-issuing. Outside of specs however, the term normally only refers to inter-PKI cross-certification.

CHAPTER

FOUR

EXAMPLES

The examples are meant to be done in order, each providing the basis for the ones that follow. They are deliberately low on prose, we prefer to let the configuration files and command lines speak for themselves.

You will find a reference section at the bottom of each page, with links to relevant parts of the OpenSSL documentation. Please use the links for details on command line options and configuration file settings.

Note: You need at least OpenSSL 1.0.1. Check with:

```
openssl version
```

4.1 Simple PKI

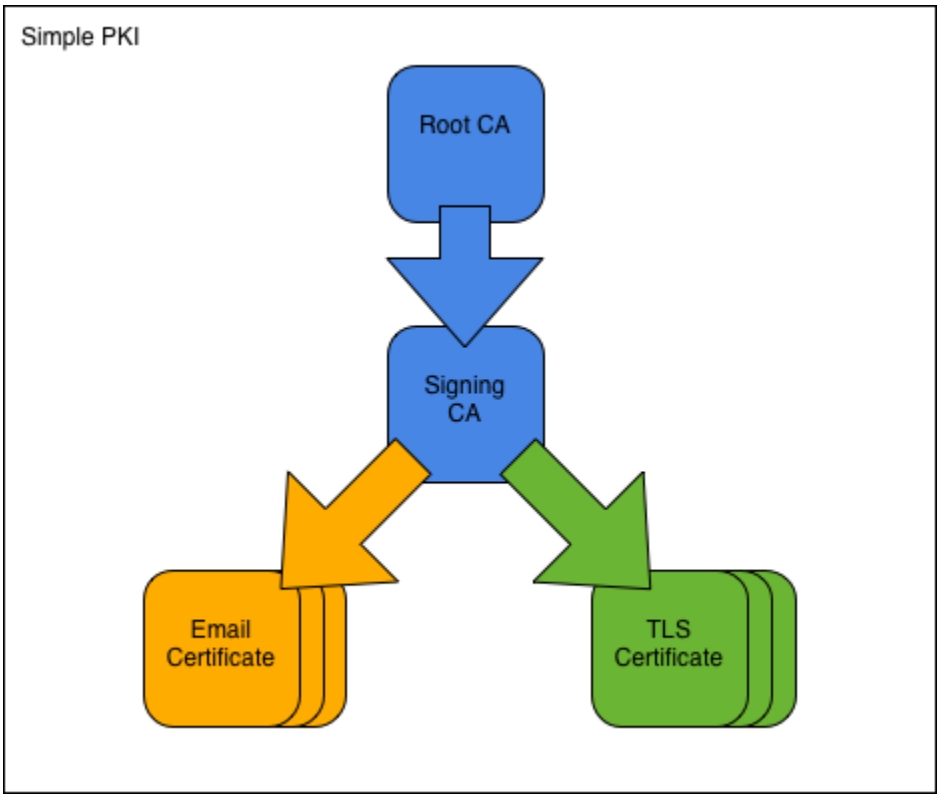
In this example we create the simplest possible PKI: One root CA and one signing CA. We use the CA to issue two types of user certificates.

4.1.1 Simple PKI

The Simple PKI consists of one root CA and one signing CA.

Overview

We assume an organisation named **Simple Inc**, controlling the domain simple.org. The organisation runs a small PKI to secure its email and intranet traffic.



To construct the PKI, we first create the Simple Root CA and its CA certificate. We then use the root CA to create the Simple Signing CA. Once the CAs are in place, we issue an email-protection certificate to employee Fred Flintstone and a TLS-server certificate to the webserver at www.simple.org. Finally, we look at the output formats the CA needs to support and show how to view the contents of files we have created.

All commands are ready to be copy/pasted into a terminal session. When you have reached the end of this page, you will have performed all operations you are likely to encounter in a PKI.

To get started, fetch the Simple PKI example files and change into the new directory:

```
git clone https://github.com/mwiora/pki-example-1.git
cd pki-example-1
```

Configuration Files

We use one configuration file per CA:

Root CA Configuration File

```
# Simple Root CA

# The [default] section contains global constants that can be referred to from
# the entire configuration file. It may also hold settings pertaining to more
# than one openssl command.

[ default ]
ca                      = root-ca               # CA name
dir                     = .                     # Top dir

# The next part of the configuration file is used by the openssl req command.
# It defines the CA's key pair, its DN, and the desired extensions for the CA
# certificate.

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = no                    # Don't prompt for DN
distinguished_name      = ca_dn                 # DN section
req_extensions          = ca_reqext             # Desired extensions

[ ca_dn ]
0.domainComponent       = "org"
1.domainComponent       = "simple"
organizationName        = "Simple Inc"
organizationalUnitName  = "Simple Root CA"
commonName              = "Simple Root CA"

[ ca_reqext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true
subjectKeyIdentifier    = hash

# The remainder of the configuration file is used by the openssl ca command.
# The CA section defines the locations of CA assets, as well as the policies
# applying to the CA.

[ ca ]
default_ca              = root_ca               # The default CA section

[ root_ca ]
certificate             = $dir/ca/$ca.crt       # The CA cert
private_key             = $dir/ca/$ca/private/$ca.key # CA private key
```

(continues on next page)

(continued from previous page)

```
new_certs_dir           = $dir/ca/$ca           # Certificate archive
serial                  = $dir/ca/$ca/db/$ca.crt.srl # Serial number file
crlnumber               = $dir/ca/$ca/db/$ca.crl.srl # CRL number file
database                = $dir/ca/$ca/db/$ca.db # Index file
unique_subject          = no                    # Require unique subject
default_days            = 3652                  # How long to certify for
default_md              = sha256                # MD to use
policy                  = match_pol             # Default naming policy
email_in_dn             = no                    # Add email to cert DN
preserve                = no                    # Keep passed DN ordering
name_opt                = ca_default            # Subject DN display options
cert_opt                = ca_default            # Certificate display options
copy_extensions         = none                  # Copy extensions from CSR
x509_extensions         = signing_ca_ext        # Default cert extensions
default_crl_days        = 365                   # How long before next CRL
crl_extensions          = crl_ext               # CRL extensions

# Naming policies control which parts of a DN end up in the certificate and
# under what circumstances certification should be denied.

[ match_pol ]
domainComponent         = match                 # Must match 'simple.org'
organizationName        = match                 # Must match 'Simple Inc'
organizationalUnitName  = optional              # Included if present
commonName              = supplied              # Must be present

[ any_pol ]
domainComponent         = optional
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = optional
emailAddress            = optional

# Certificate extensions define what types of certificates the CA is able to
# create.

[ root_ca_ext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always

[ signing_ca_ext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true,pathlen:0
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always

# CRL extensions exist solely to point to the CA certificate that has issued
```

(continues on next page)

(continued from previous page)

```
# the CRL.

[ crl_ext ]
authorityKeyIdentifier  = keyid:always
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/ca.html#CONFIGURATION_FILE_OPTIONS
- http://www.openssl.org/docs/apps/x509v3_config.html

Signing CA Configuration File

```
# Simple Signing CA

# The [default] section contains global constants that can be referred to from
# the entire configuration file. It may also hold settings pertaining to more
# than one openssl command.

[ default ]
ca                      = signing-ca            # CA name
dir                     = .                     # Top dir

# The next part of the configuration file is used by the openssl req command.
# It defines the CA's key pair, its DN, and the desired extensions for the CA
# certificate.

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = no                    # Don't prompt for DN
distinguished_name      = ca_dn                 # DN section
req_extensions          = ca_reqext             # Desired extensions

[ ca_dn ]
0.domainComponent       = "org"
1.domainComponent       = "simple"
organizationName        = "Simple Inc"
organizationalUnitName  = "Simple Signing CA"
commonName              = "Simple Signing CA"

[ ca_reqext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true,pathlen:0
```

(continues on next page)

(continued from previous page)

```
subjectKeyIdentifier    = hash

# The remainder of the configuration file is used by the openssl ca command.
# The CA section defines the locations of CA assets, as well as the policies
# applying to the CA.

[ ca ]
default_ca              = signing_ca            # The default CA section

[ signing_ca ]
certificate             = $dir/ca/$ca.crt       # The CA cert
private_key             = $dir/ca/$ca/private/$ca.key # CA private key
new_certs_dir           = $dir/ca/$ca           # Certificate archive
serial                  = $dir/ca/$ca/db/$ca.crt.srl # Serial number file
crlnumber               = $dir/ca/$ca/db/$ca.crl.srl # CRL number file
database                = $dir/ca/$ca/db/$ca.db # Index file
unique_subject          = no                    # Require unique subject
default_days            = 730                   # How long to certify for
default_md              = sha256                # MD to use
policy                  = match_pol             # Default naming policy
email_in_dn             = no                    # Add email to cert DN
preserve                = no                    # Keep passed DN ordering
name_opt                = ca_default            # Subject DN display options
cert_opt                = ca_default            # Certificate display options
copy_extensions         = copy                  # Copy extensions from CSR
x509_extensions         = email_ext             # Default cert extensions
default_crl_days        = 7                     # How long before next CRL
crl_extensions          = crl_ext               # CRL extensions

# Naming policies control which parts of a DN end up in the certificate and
# under what circumstances certification should be denied.

[ match_pol ]
domainComponent         = match                 # Must match 'simple.org'
organizationName        = match                 # Must match 'Simple Inc'
organizationalUnitName  = optional              # Included if present
commonName              = supplied              # Must be present

[ any_pol ]
domainComponent         = optional
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = optional
emailAddress            = optional

# Certificate extensions define what types of certificates the CA is able to
# create.

[ email_ext ]
```

(continues on next page)

(continued from previous page)

```
keyUsage                = critical,digitalSignature,keyEncipherment
basicConstraints        = CA:false
extendedKeyUsage        = emailProtection,clientAuth
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always

[ server_ext ]
keyUsage                = critical,digitalSignature,keyEncipherment
basicConstraints        = CA:false
extendedKeyUsage        = serverAuth,clientAuth
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always

# CRL extensions exist solely to point to the CA certificate that has issued
# the CRL.

[ crl_ext ]
authorityKeyIdentifier  = keyid:always
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/ca.html#CONFIGURATION_FILE_OPTIONS
- http://www.openssl.org/docs/apps/x509v3_config.html

And one configuration file per CSR type:

Email Certificate Request Configuration File

```
# Email certificate request

# This file is used by the openssl req command. Since we cannot know the DN in
# advance the user is prompted for DN information.

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = yes                   # Prompt for DN
distinguished_name      = email_dn              # DN template
req_extensions          = email_reqext          # Desired extensions

[ email_dn ]
0.domainComponent       = "1. Domain Component         (eg, com)      "
1.domainComponent       = "2. Domain Component         (eg, company)  "
```

(continues on next page)

(continued from previous page)

```
2.domainComponent       = "3. Domain Component         (eg, pki)      "
organizationName        = "4. Organization Name        (eg, company)  "
organizationalUnitName  = "5. Organizational Unit Name (eg, section)  "
commonName              = "6. Common Name              (eg, full name)"
commonName_max          = 64
emailAddress            = "7. Email Address            (eg, name@fqdn)"
emailAddress_max        = 40

[ email_reqext ]
keyUsage                = critical,digitalSignature,keyEncipherment
extendedKeyUsage        = emailProtection,clientAuth
subjectKeyIdentifier    = hash
subjectAltName          = email:move
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/x509v3_config.html

TLS Server Certificate Request Configuration File

```
# TLS server certificate request

# This file is used by the openssl req command. The subjectAltName cannot be
# prompted for and must be specified in the SAN environment variable.

[ default ]
SAN                     = DNS:yourdomain.tld    # Default value

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = no                    # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = yes                   # Prompt for DN
distinguished_name      = server_dn             # DN template
req_extensions          = server_reqext         # Desired extensions

[ server_dn ]
0.domainComponent       = "1. Domain Component         (eg, com)      "
1.domainComponent       = "2. Domain Component         (eg, company)  "
2.domainComponent       = "3. Domain Component         (eg, pki)      "
organizationName        = "4. Organization Name        (eg, company)  "
organizationalUnitName  = "5. Organizational Unit Name (eg, section)  "
commonName              = "6. Common Name              (eg, FQDN)     "
commonName_max          = 64
```

(continues on next page)

(continued from previous page)

```
[ server_reqext ]
keyUsage                = critical,digitalSignature,keyEncipherment
extendedKeyUsage        = serverAuth,clientAuth
subjectKeyIdentifier    = hash
subjectAltName          = $ENV::SAN
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/x509v3_config.html

Please familiarize yourself with the configuration files before you continue.

1. Create Root CA

1.1 Create directories

```
mkdir -p ca/root-ca/private ca/root-ca/db crl certs
chmod 700 ca/root-ca/private
```

The `ca` directory holds CA resources, the `crl` directory holds CRLs, and the `certs` directory holds user certificates.

1.2 Create database

```
cp /dev/null ca/root-ca/db/root-ca.db
cp /dev/null ca/root-ca/db/root-ca.db.attr
echo 01 > ca/root-ca/db/root-ca.crt.srl
echo 01 > ca/root-ca/db/root-ca.crl.srl
```

The database files must exist before the `openssl ca` command can be used. The file contents are described in *Appendix B: CA Database*.

1.3 Create CA request

```
openssl req -new \
    -config etc/root-ca.conf \
    -out ca/root-ca.csr \
    -keyout ca/root-ca/private/root-ca.key
```

With the `openssl req -new` command we create a private key and a certificate signing request (CSR) for the root CA. You will be asked for a passphrase to protect the private key. The `openssl req` command takes its configuration from the [req] section of the *configuration file*.

1.4 Create CA certificate

```
openssl ca -selfsign \
    -config etc/root-ca.conf \
    -in ca/root-ca.csr \
    -out ca/root-ca.crt \
    -extensions root_ca_ext
```

With the `openssl ca` command we issue a root CA certificate based on the CSR. The root certificate is self-signed and serves as the starting point for all trust relationships in the PKI. The `openssl ca` command takes its configuration from the [ca] section of the *configuration file*.

2. Create Signing CA

2.1 Create directories

```
mkdir -p ca/signing-ca/private ca/signing-ca/db crl certs
chmod 700 ca/signing-ca/private
```

The `ca` directory holds CA resources, the `crl` directory holds CRLs, and the `certs` directory holds user certificates. We will use this layout for all CAs in this tutorial.

2.2 Create database

```
cp /dev/null ca/signing-ca/db/signing-ca.db
cp /dev/null ca/signing-ca/db/signing-ca.db.attr
echo 01 > ca/signing-ca/db/signing-ca.crt.srl
echo 01 > ca/signing-ca/db/signing-ca.crl.srl
```

The contents of these files are described in *Appendix B: CA Database*.

2.3 Create CA request

```
openssl req -new \
    -config etc/signing-ca.conf \
    -out ca/signing-ca.csr \
    -keyout ca/signing-ca/private/signing-ca.key
```

With the `openssl req -new` command we create a private key and a CSR for the signing CA. You will be asked for a passphrase to protect the private key. The `openssl req` command takes its configuration from the [req] section of the *configuration file*.

2.4 Create CA certificate

```
openssl ca \
    -config etc/root-ca.conf \
    -in ca/signing-ca.csr \
    -out ca/signing-ca.crt \
    -extensions signing_ca_ext
```

With the `openssl ca` command we issue a certificate based on the CSR. The command takes its configuration from the [ca] section of the *configuration file*. Note that it is the root CA that issues the signing CA certificate! Note also that we attach a different set of extensions.

3. Operate Signing CA

3.1 Create email request

```
openssl req -new \
    -config etc/email.conf \
    -out certs/fred.csr \
    -keyout certs/fred.key
```

With the `openssl req -new` command we create the private key and CSR for an email-protection certificate. We use a *request configuration file* specifically prepared for the task. When prompted enter these DN components: DC=org, DC=simple, O=Simple Inc, CN=Fred Flintstone, emailAddress=fred@simple.org. Leave other fields empty.

3.2 Create email certificate

```
openssl ca \
    -config etc/signing-ca.conf \
    -in certs/fred.csr \
    -out certs/fred.crt \
    -extensions email_ext
```

We use the signing CA to issue the email-protection certificate. The certificate type is defined by the extensions we attach. A copy of the certificate is saved in the certificate archive under the name `ca/signing-ca/01.pem` (01 being the certificate serial number in hex.)

3.3 Create TLS server request

```
SAN=DNS:www.simple.org \
openssl req -new \
    -config etc/server.conf \
    -out certs/simple.org.csr \
    -keyout certs/simple.org.key
```

Next we create the private key and CSR for a TLS-server certificate using another *request configuration file*. When prompted enter these DN components: DC=org, DC=simple, O=Simple Inc, CN=www.simple.org. Note that the subjectAltName must be specified as environment variable. Note also that server keys typically have no passphrase.

3.4 Create TLS server certificate

```
openssl ca \
    -config etc/signing-ca.conf \
    -in certs/simple.org.csr \
    -out certs/simple.org.crt \
    -extensions server_ext
```

We use the signing CA to issue the server certificate. The certificate type is defined by the extensions we attach. A copy of the certificate is saved in the certificate archive under the name `ca/signing-ca/02.pem`.

3.5 Revoke certificate

```
openssl ca \
    -config etc/signing-ca.conf \
    -revoke ca/signing-ca/01.pem \
    -crl_reason superseded
```

Certain events, like certificate replacement or loss of private key, require a certificate to be revoked before its scheduled expiration date. The `openssl ca -revoke` command marks a certificate as revoked in the CA database. It will from then on be included in CRLs issued by the CA. The above command revokes the certificate with serial number 01 (hex).

3.6 Create CRL

```
openssl ca -gencrl \
    -config etc/signing-ca.conf \
    -out crl/signing-ca.crl
```

The `openssl ca -gencrl` command creates a certificate revocation list (CRL). The CRL contains all revoked, not-yet-expired certificates from the CA database. A new CRL must be issued at regular intervals.

4. Output Formats

4.1 Create DER certificate

```
openssl x509 \
    -in certs/fred.crt \
    -out certs/fred.cer \
    -outform der
```

All published certificates must be in DER format [**RFC 2585#section-3**]. Also see *Appendix A: MIME Types*.

4.2 Create DER CRL

```
openssl crl \
    -in crl/signing-ca.crl \
    -out crl/signing-ca.crl \
    -outform der
```

All published CRLs must be in DER format [**RFC 2585#section-3**]. Also see *Appendix A: MIME Types*.

4.3 Create PKCS#7 bundle

```
openssl crl2pkcs7 -nocrl \
    -certfile ca/signing-ca.crt \
    -certfile ca/root-ca.crt \
    -out ca/signing-ca-chain.p7c \
    -outform der
```

PKCS#7 is used to bundle two or more certificates. The format would also allow for CRLs but they are not used in practice.

4.4 Create PKCS#12 bundle

```
openssl pkcs12 -export \
    -name "Fred Flintstone" \
    -inkey certs/fred.key \
    -in certs/fred.crt \
    -out certs/fred.p12
```

PKCS#12 is used to bundle a certificate and its private key. Additional certificates may be added, typically the certificates comprising the chain up to the Root CA.

4.5 Create PEM bundle

```
cat ca/signing-ca.crt ca/root-ca.crt > \
    ca/signing-ca-chain.pem

cat certs/fred.key certs/fred.crt > \
    certs/fred.pem
```

PEM bundles are created by concatenating other PEM-formatted files. The most common forms are “cert chain”, “key + cert”, and “key + cert chain”. PEM bundles are supported by OpenSSL and most software based on it (e.g. Apache mod_ssl and stunnel.)

5. View Results

5.1 View request

```
openssl req \
    -in certs/fred.csr \
    -noout \
    -text
```

The `openssl req` command can be used to display the contents of CSR files. The `-noout` and `-text` options select a human-readable output format.

5.2 View certificate

```
openssl x509 \
    -in certs/fred.crt \
    -noout \
    -text
```

The `openssl x509` command can be used to display the contents of certificate files. The `-noout` and `-text` options have the same purpose as before.

5.3 View CRL

```
openssl crl \
    -in crl/signing-ca.crl \
    -inform der \
    -noout \
    -text
```

The `openssl crl` command can be used to view the contents of CRL files. Note that we specify `-inform der` because we have already converted the CRL in step 4.2.

5.4 View PKCS#7 bundle

```
openssl pkcs7 \
    -in ca/signing-ca-chain.p7c \
    -inform der \
    -noout \
    -text \
    -print_certs
```

The `openssl pkcs7` command can be used to display the contents of PKCS#7 bundles.

5.5 View PKCS#12 bundle

```
openssl pkcs12 \
    -in certs/fred.p12 \
    -nodes \
    -info
```

The `openssl pkcs12` command can be used to display the contents of PKCS#12 bundles.

References

- http://www.openssl.org/docs/apps/req.html
- http://www.openssl.org/docs/apps/ca.html
- http://www.openssl.org/docs/apps/x509.html
- http://www.openssl.org/docs/apps/crl.html
- http://www.openssl.org/docs/apps/crl2pkcs7.html
- http://www.openssl.org/docs/apps/pkcs7.html
- http://www.openssl.org/docs/apps/pkcs12.html

4.2 Advanced PKI

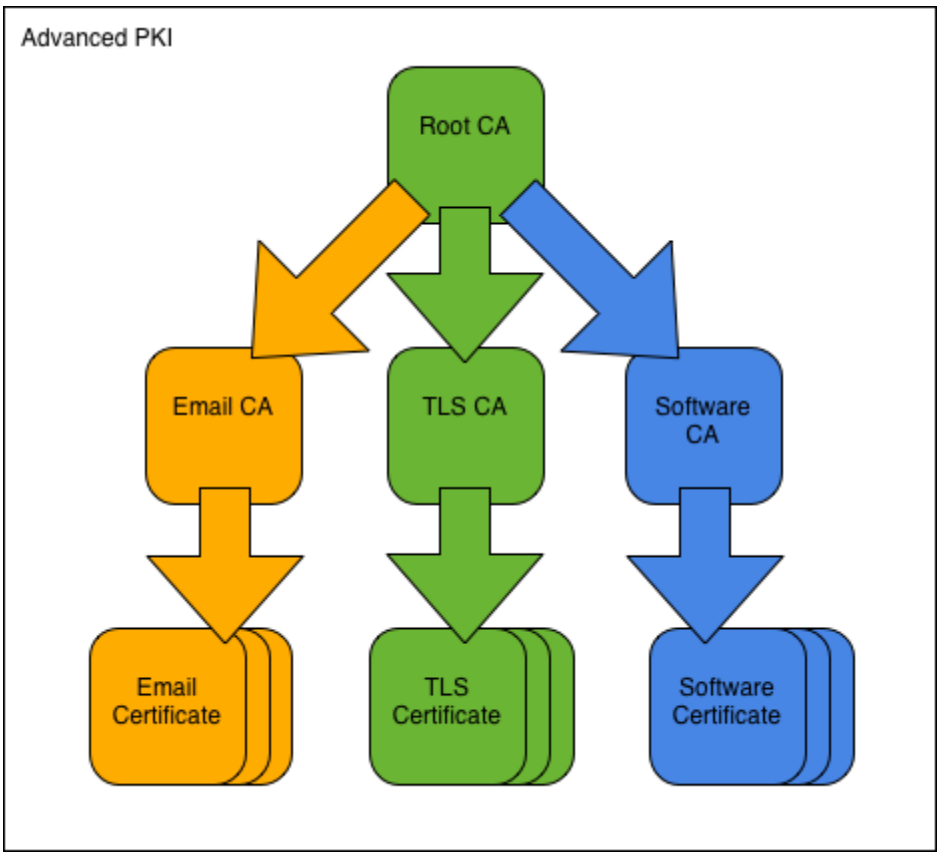
In this example we create a larger setup, consisting of a root CA and three signing CAs. We use the CAs to issue 4 different types of user certificates. We also encounter two new certificate extensions: authorityInfoAccess and crlDistributionPoints.

4.2.1 Advanced PKI

The Advanced PKI consists of a root CA and a layer of subordinate CAs.

Overview

We assume a company named **Green AS**, controlling the domain green.no. The company runs a three-pronged PKI to serve its security needs.



To implement the PKI, we first create the Green Root CA and its CA certificate. We then use the root CA to create the three signing CAs: Green Email CA, Green TLS CA, and Green Software CA. The CAs in place we proceed to show them in operation, issuing user certificates for email-protection, TLS-authentication, and code-signing purposes respectively.

All commands are ready to be copy/pasted into a terminal session. When you have reached the end of this page, you will have built a PKI with multiple signing CAs and issued 4 different types of user certificates.

To get started, fetch the Advanced PKI example files and change into the new directory:

```
git clone https://github.com/mwiora/pki-example-2.git
cd pki-example-2
```

Configuration Files

We use one configuration file per CA:

Root CA Configuration File

```
# Green Root CA

[ default ]
ca                      = root-ca               # CA name
dir                     = .                     # Top dir
base_url                = http://green.no/ca    # CA base URL
aia_url                 = $base_url/$ca.cer     # CA certificate URL
crl_url                 = $base_url/$ca.crl     # CRL distribution point
name_opt                = multiline,-esc_msb,utf8 # Display UTF-8 characters

# CA certificate request

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = no                    # Don't prompt for DN
distinguished_name      = ca_dn                 # DN section
req_extensions          = ca_reqext             # Desired extensions

[ ca_dn ]
countryName             = "NO"
organizationName        = "Green AS"
organizationalUnitName  = "Green Certificate Authority"
commonName              = "Green Root CA"

[ ca_reqext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true
subjectKeyIdentifier    = hash

# CA operational settings

[ ca ]
default_ca              = root_ca               # The default CA section

[ root_ca ]
certificate             = $dir/ca/$ca.crt       # The CA cert
private_key             = $dir/ca/$ca/private/$ca.key # CA private key
new_certs_dir           = $dir/ca/$ca           # Certificate archive
serial                  = $dir/ca/$ca/db/$ca.crt.srl # Serial number file
crlnumber               = $dir/ca/$ca/db/$ca.crl.srl # CRL number file
database                = $dir/ca/$ca/db/$ca.db # Index file
unique_subject          = no                    # Require unique subject
```

(continues on next page)

(continued from previous page)

```
default_days            = 3652                  # How long to certify for
default_md              = sha256                # MD to use
policy                  = match_pol             # Default naming policy
email_in_dn             = no                    # Add email to cert DN
preserve                = no                    # Keep passed DN ordering
name_opt                = $name_opt             # Subject DN display options
cert_opt                = ca_default            # Certificate display options
copy_extensions         = none                  # Copy extensions from CSR
x509_extensions         = signing_ca_ext        # Default cert extensions
default_crl_days        = 365                   # How long before next CRL
crl_extensions          = crl_ext               # CRL extensions

[ match_pol ]
countryName             = match                 # Must match 'NO'
stateOrProvinceName     = optional              # Included if present
localityName            = optional              # Included if present
organizationName        = match                 # Must match 'Green AS'
organizationalUnitName  = optional              # Included if present
commonName              = supplied              # Must be present

[ any_pol ]
domainComponent         = optional
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = optional
emailAddress            = optional

# Extensions

[ root_ca_ext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always

[ signing_ca_ext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true,pathlen:0
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @issuer_info
crlDistributionPoints   = @crl_info

[ crl_ext ]
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @issuer_info

[ issuer_info ]
caIssuers;URI.0         = $aia_url
```

(continues on next page)

(continued from previous page)

```
[ crl_info ]
URI.0                   = $crl_url
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/x509.html#NAME_OPTIONS
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/ca.html#CONFIGURATION_FILE_OPTIONS
- http://www.openssl.org/docs/apps/x509v3_config.html

Email CA Configuration File

```
# Green Email CA

[ default ]
ca                      = email-ca              # CA name
dir                     = .                     # Top dir
base_url                = http://green.no/ca    # CA base URL
aia_url                 = $base_url/$ca.cer     # CA certificate URL
crl_url                 = $base_url/$ca.crl     # CRL distribution point
name_opt                = multiline,-esc_msb,utf8 # Display UTF-8 characters

# CA certificate request

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = no                    # Don't prompt for DN
distinguished_name      = ca_dn                 # DN section
req_extensions          = ca_reqext             # Desired extensions

[ ca_dn ]
countryName             = "NO"
organizationName        = "Green AS"
organizationalUnitName  = "Green Certificate Authority"
commonName              = "Green Email CA"

[ ca_reqext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true,pathlen:0
subjectKeyIdentifier    = hash
```

(continues on next page)

(continued from previous page)

```
# CA operational settings

[ ca ]
default_ca              = email_ca              # The default CA section

[ email_ca ]
certificate             = $dir/ca/$ca.crt       # The CA cert
private_key             = $dir/ca/$ca/private/$ca.key # CA private key
new_certs_dir           = $dir/ca/$ca           # Certificate archive
serial                  = $dir/ca/$ca/db/$ca.crt.srl # Serial number file
crlnumber               = $dir/ca/$ca/db/$ca.crl.srl # CRL number file
database                = $dir/ca/$ca/db/$ca.db # Index file
unique_subject          = no                    # Require unique subject
default_days            = 730                   # How long to certify for
default_md              = sha256                # MD to use
policy                  = match_pol             # Default naming policy
email_in_dn             = no                    # Add email to cert DN
preserve                = no                    # Keep passed DN ordering
name_opt                = $name_opt             # Subject DN display options
cert_opt                = ca_default            # Certificate display options
copy_extensions         = copy                  # Copy extensions from CSR
x509_extensions         = email_ext             # Default cert extensions
default_crl_days        = 1                     # How long before next CRL
crl_extensions          = crl_ext               # CRL extensions

[ match_pol ]
countryName             = match                 # Must match 'NO'
stateOrProvinceName     = optional              # Included if present
localityName            = optional              # Included if present
organizationName        = match                 # Must match 'Green AS'
organizationalUnitName  = optional              # Included if present
commonName              = supplied              # Must be present

[ any_pol ]
domainComponent         = optional
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = optional
emailAddress            = optional

# Extensions

[ email_ext ]
keyUsage                = critical,digitalSignature,keyEncipherment
basicConstraints        = CA:false
extendedKeyUsage        = emailProtection,clientAuth,anyExtendedKeyUsage
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @issuer_info
```

(continues on next page)

(continued from previous page)

```
crlDistributionPoints   = @crl_info

[ crl_ext ]
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @issuer_info

[ issuer_info ]
caIssuers;URI.0         = $aia_url

[ crl_info ]
URI.0                   = $crl_url
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/x509.html#NAME_OPTIONS
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/ca.html#CONFIGURATION_FILE_OPTIONS
- http://www.openssl.org/docs/apps/x509v3_config.html

TLS CA Configuration File

```
# Green TLS CA

[ default ]
ca                      = tls-ca                # CA name
dir                     = .                     # Top dir
base_url                = http://green.no/ca    # CA base URL
aia_url                 = $base_url/$ca.cer     # CA certificate URL
crl_url                 = $base_url/$ca.crl     # CRL distribution point
name_opt                = multiline,-esc_msb,utf8 # Display UTF-8 characters

# CA certificate request

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = no                    # Don't prompt for DN
distinguished_name      = ca_dn                 # DN section
req_extensions          = ca_reqext             # Desired extensions

[ ca_dn ]
countryName             = "NO"
organizationName        = "Green AS"
```

(continues on next page)

(continued from previous page)

```
organizationalUnitName  = "Green Certificate Authority"
commonName              = "Green TLS CA"

[ ca_reqext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true,pathlen:0
subjectKeyIdentifier    = hash

# CA operational settings

[ ca ]
default_ca              = tls_ca                # The default CA section

[ tls_ca ]
certificate             = $dir/ca/$ca.crt       # The CA cert
private_key             = $dir/ca/$ca/private/$ca.key # CA private key
new_certs_dir           = $dir/ca/$ca           # Certificate archive
serial                  = $dir/ca/$ca/db/$ca.crt.srl # Serial number file
crlnumber               = $dir/ca/$ca/db/$ca.crl.srl # CRL number file
database                = $dir/ca/$ca/db/$ca.db # Index file
unique_subject          = no                    # Require unique subject
default_days            = 730                   # How long to certify for
default_md              = sha256                # MD to use
policy                  = match_pol             # Default naming policy
email_in_dn             = no                    # Add email to cert DN
preserve                = no                    # Keep passed DN ordering
name_opt                = $name_opt             # Subject DN display options
cert_opt                = ca_default            # Certificate display options
copy_extensions         = copy                  # Copy extensions from CSR
x509_extensions         = server_ext            # Default cert extensions
default_crl_days        = 1                     # How long before next CRL
crl_extensions          = crl_ext               # CRL extensions

[ match_pol ]
countryName             = match                 # Must match 'NO'
stateOrProvinceName     = optional              # Included if present
localityName            = optional              # Included if present
organizationName        = match                 # Must match 'Green AS'
organizationalUnitName  = optional              # Included if present
commonName              = supplied              # Must be present

[ extern_pol ]
countryName             = supplied              # Must be present
stateOrProvinceName     = optional              # Included if present
localityName            = optional              # Included if present
organizationName        = supplied              # Must be present
organizationalUnitName  = optional              # Included if present
commonName              = supplied              # Must be present

[ any_pol ]
domainComponent         = optional
countryName             = optional
```

(continues on next page)

(continued from previous page)

```
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = optional
emailAddress            = optional

# Extensions

[ server_ext ]
keyUsage                = critical,digitalSignature,keyEncipherment
basicConstraints        = CA:false
extendedKeyUsage        = serverAuth,clientAuth
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @issuer_info
crlDistributionPoints   = @crl_info

[ client_ext ]
keyUsage                = critical,digitalSignature
basicConstraints        = CA:false
extendedKeyUsage        = clientAuth
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @issuer_info
crlDistributionPoints   = @crl_info

[ crl_ext ]
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @issuer_info

[ issuer_info ]
caIssuers;URI.0         = $aia_url

[ crl_info ]
URI.0                   = $crl_url
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/x509.html#NAME_OPTIONS
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/ca.html#CONFIGURATION_FILE_OPTIONS
- http://www.openssl.org/docs/apps/x509v3_config.html

Software CA Configuration File

```
# Green Software CA

[ default ]
ca                      = software-ca           # CA name
dir                     = .                     # Top dir
base_url                = http://green.no/ca    # CA base URL
aia_url                 = $base_url/$ca.cer     # CA certificate URL
crl_url                 = $base_url/$ca.crl     # CRL distribution point
name_opt                = multiline,-esc_msb,utf8 # Display UTF-8 characters

# CA certificate request

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = no                    # Don't prompt for DN
distinguished_name      = ca_dn                 # DN section
req_extensions          = ca_reqext             # Desired extensions

[ ca_dn ]
countryName             = "NO"
organizationName        = "Green AS"
organizationalUnitName  = "Green Certificate Authority"
commonName              = "Green Software CA"

[ ca_reqext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true,pathlen:0
subjectKeyIdentifier    = hash

# CA operational settings

[ ca ]
default_ca              = software_ca           # The default CA section

[ software_ca ]
certificate             = $dir/ca/$ca.crt       # The CA cert
private_key             = $dir/ca/$ca/private/$ca.key # CA private key
new_certs_dir           = $dir/ca/$ca           # Certificate archive
serial                  = $dir/ca/$ca/db/$ca.crt.srl # Serial number file
crlnumber               = $dir/ca/$ca/db/$ca.crl.srl # CRL number file
database                = $dir/ca/$ca/db/$ca.db # Index file
unique_subject          = no                    # Require unique subject
default_days            = 1826                  # How long to certify for
default_md              = sha256                # MD to use
policy                  = match_pol             # Default naming policy
email_in_dn             = no                    # Add email to cert DN
```

(continues on next page)

(continued from previous page)

```
preserve                = no                    # Keep passed DN ordering
name_opt                = $name_opt             # Subject DN display options
cert_opt                = ca_default            # Certificate display options
copy_extensions         = copy                  # Copy extensions from CSR
x509_extensions         = codesign_ext          # Default cert extensions
default_crl_days        = 30                    # How long before next CRL
crl_extensions          = crl_ext               # CRL extensions

[ match_pol ]
countryName             = match                 # Must match 'NO'
stateOrProvinceName     = optional              # Included if present
localityName            = optional              # Included if present
organizationName        = match                 # Must match 'Green AS'
organizationalUnitName  = optional              # Included if present
commonName              = supplied              # Must be present

[ any_pol ]
domainComponent         = optional
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = optional
emailAddress            = optional

# Extensions

[ codesign_ext ]
keyUsage                = critical,digitalSignature
basicConstraints        = CA:false
extendedKeyUsage        = critical,codeSigning
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @issuer_info
crlDistributionPoints   = @crl_info

[ crl_ext ]
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @issuer_info

[ issuer_info ]
caIssuers;URI.0         = $aia_url

[ crl_info ]
URI.0                   = $crl_url
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/x509.html#NAME_OPTIONS
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/ca.html#CONFIGURATION_FILE_OPTIONS
- http://www.openssl.org/docs/apps/x509v3_config.html

And one configuration file per CSR type:

Email Certificate Request Configuration File

```
# Email certificate request

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = yes                   # Prompt for DN
distinguished_name      = email_dn              # DN template
req_extensions          = email_reqext          # Desired extensions

[ email_dn ]
countryName             = "1. Country Name (2 letters) (eg, US)       "
countryName_max         = 2
stateOrProvinceName     = "2. State or Province Name   (eg, region)   "
localityName            = "3. Locality Name            (eg, city)     "
organizationName        = "4. Organization Name        (eg, company)  "
organizationalUnitName  = "5. Organizational Unit Name (eg, section)  "
commonName              = "6. Common Name              (eg, full name)"
commonName_max          = 64
emailAddress            = "7. Email Address            (eg, name@fqdn)"
emailAddress_max        = 40

[ email_reqext ]
keyUsage                = critical,digitalSignature,keyEncipherment
extendedKeyUsage        = emailProtection,clientAuth
subjectKeyIdentifier    = hash
subjectAltName          = email:move
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/x509v3_config.html

TLS Server Certificate Request Configuration File

```
# TLS server certificate request

[ default ]
SAN                     = DNS:yourdomain.tld    # Default value

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = no                    # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = yes                   # Prompt for DN
distinguished_name      = server_dn             # DN template
req_extensions          = server_reqext         # Desired extensions

[ server_dn ]
countryName             = "1. Country Name (2 letters) (eg, US)       "
countryName_max         = 2
stateOrProvinceName     = "2. State or Province Name   (eg, region)   "
localityName            = "3. Locality Name            (eg, city)     "
organizationName        = "4. Organization Name        (eg, company)  "
organizationalUnitName  = "5. Organizational Unit Name (eg, section)  "
commonName              = "6. Common Name              (eg, FQDN)     "
commonName_max          = 64

[ server_reqext ]
keyUsage                = critical,digitalSignature,keyEncipherment
extendedKeyUsage        = serverAuth,clientAuth
subjectKeyIdentifier    = hash
subjectAltName          = $ENV::SAN
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/x509v3_config.html

TLS Client Certificate Request Configuration File

```
# TLS client certificate request

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = yes                   # Prompt for DN
distinguished_name      = client_dn             # DN template
req_extensions          = client_reqext         # Desired extensions

[ client_dn ]
countryName             = "1. Country Name (2 letters) (eg, US)       "
countryName_max         = 2
stateOrProvinceName     = "2. State or Province Name   (eg, region)   "
localityName            = "3. Locality Name            (eg, city)     "
organizationName        = "4. Organization Name        (eg, company)  "
organizationalUnitName  = "5. Organizational Unit Name (eg, section)  "
commonName              = "6. Common Name              (eg, full name)"
commonName_max          = 64
emailAddress            = "7. Email Address            (eg, name@fqdn)"
emailAddress_max        = 40

[ client_reqext ]
keyUsage                = critical,digitalSignature
extendedKeyUsage        = clientAuth
subjectKeyIdentifier    = hash
subjectAltName          = email:move
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/x509v3_config.html

Code-Signing Certificate Request Configuration File

```
# Code-signing certificate request

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = yes                   # Prompt for DN
```

(continues on next page)

(continued from previous page)

```
distinguished_name      = codesign_dn           # DN template
req_extensions          = codesign_reqext       # Desired extensions

[ codesign_dn ]
countryName             = "1. Country Name (2 letters) (eg, US)       "
countryName_max         = 2
stateOrProvinceName     = "2. State or Province Name   (eg, region)   "
localityName            = "3. Locality Name            (eg, city)     "
organizationName        = "4. Organization Name        (eg, company)  "
organizationalUnitName  = "5. Organizational Unit Name (eg, section)  "
commonName              = "6. Common Name              (eg, full name)"
commonName_max          = 64

[ codesign_reqext ]
keyUsage                = critical,digitalSignature
extendedKeyUsage        = critical,codeSigning
subjectKeyIdentifier    = hash
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/x509v3_config.html

Please study the configuration files before you continue.

1. Create Root CA

1.1 Create directories

```
mkdir -p ca/root-ca/private ca/root-ca/db crl certs
chmod 700 ca/root-ca/private
```

The `ca` directory holds CA resources, the `crl` directory holds CRLs, and the `certs` directory holds user certificates. The directory layout stays the same throughout the tutorial.

1.2 Create database

```
cp /dev/null ca/root-ca/db/root-ca.db
cp /dev/null ca/root-ca/db/root-ca.db.attr
echo 01 > ca/root-ca/db/root-ca.crt.srl
echo 01 > ca/root-ca/db/root-ca.crl.srl
```

The files must exist before the `openssl ca` command can be used. Also see *Appendix B: CA Database*.

1.3 Create CA request

```
openssl req -new \
    -config etc/root-ca.conf \
    -out ca/root-ca.csr \
    -keyout ca/root-ca/private/root-ca.key
```

With the `openssl req -new` command we create a private key and a CSR for the root CA. The configuration is taken from the [req] section of the *root CA configuration file*.

1.4 Create CA certificate

```
openssl ca -selfsign \
    -config etc/root-ca.conf \
    -in ca/root-ca.csr \
    -out ca/root-ca.crt \
    -extensions root_ca_ext \
    -enddate 20301231235959Z
```

With the `openssl ca` command we create a self-signed root certificate from the CSR. The configuration is taken from the [ca] section of the *root CA configuration file*. Note that we specify an end date based on the key length. 2048-bit RSA keys are deemed safe until 2030 (RSA Labs).

1.5 Create initial CRL

```
openssl ca -gencrl \
    -config etc/root-ca.conf \
    -out crl/root-ca.crl
```

With the `openssl ca -gencrl` command we generate an initial (empty) CRL.

2. Create Email CA

2.1 Create directories

```
mkdir -p ca/email-ca/private ca/email-ca/db crl certs
chmod 700 ca/email-ca/private
```

2.2 Create database

```
cp /dev/null ca/email-ca/db/email-ca.db
cp /dev/null ca/email-ca/db/email-ca.db.attr
echo 01 > ca/email-ca/db/email-ca.crt.srl
echo 01 > ca/email-ca/db/email-ca.crl.srl
```

2.3 Create CA request

```
openssl req -new \
    -config etc/email-ca.conf \
    -out ca/email-ca.csr \
    -keyout ca/email-ca/private/email-ca.key
```

We create a private key and a CSR for the email CA. The configuration is taken from the [req] section of the *email CA configuration file*.

2.4 Create CA certificate

```
openssl ca \
    -config etc/root-ca.conf \
    -in ca/email-ca.csr \
    -out ca/email-ca.crt \
    -extensions signing_ca_ext
```

We use the root CA to issue the email CA certificate. Points if you noticed that -extensions could have been omitted.

2.5 Create initial CRL

```
openssl ca -gencrl \
    -config etc/email-ca.conf \
    -out crl/email-ca.crl
```

We create an initial, empty CRL.

2.6 Create PEM bundle

```
cat ca/email-ca.crt ca/root-ca.crt > \
    ca/email-ca-chain.pem
```

We create a certificate chain file from the email CA and root CA certificates. It will come handly later as input for the `openssl pkcs12` command.

3. Create TLS CA

3.1 Create directories

```
mkdir -p ca/tls-ca/private ca/tls-ca/db crl certs
chmod 700 ca/tls-ca/private
```

3.2 Create database

```
cp /dev/null ca/tls-ca/db/tls-ca.db
cp /dev/null ca/tls-ca/db/tls-ca.db.attr
echo 01 > ca/tls-ca/db/tls-ca.crt.srl
echo 01 > ca/tls-ca/db/tls-ca.crl.srl
```

3.3 Create CA request

```
openssl req -new \
    -config etc/tls-ca.conf \
    -out ca/tls-ca.csr \
    -keyout ca/tls-ca/private/tls-ca.key
```

We create a private key and a CSR for the TLS CA. The configuration is taken from the [req] section of the *TLS CA configuration file*.

3.4 Create CA certificate

```
openssl ca \
    -config etc/root-ca.conf \
    -in ca/tls-ca.csr \
    -out ca/tls-ca.crt \
    -extensions signing_ca_ext
```

We use the root CA to issue the TLS CA certificate.

3.5 Create initial CRL

```
openssl ca -gencrl \
    -config etc/tls-ca.conf \
    -out crl/tls-ca.crl
```

We create an empty CRL.

3.6 Create PEM bundle

```
cat ca/tls-ca.crt ca/root-ca.crt > \
    ca/tls-ca-chain.pem
```

We create a certificate chain file.

4. Create Software CA

4.1 Create directories

```
mkdir -p ca/software-ca/private ca/software-ca/db crl certs
chmod 700 ca/software-ca/private
```

4.2 Create database

```
cp /dev/null ca/software-ca/db/software-ca.db
cp /dev/null ca/software-ca/db/software-ca.db.attr
echo 01 > ca/software-ca/db/software-ca.crt.srl
echo 01 > ca/software-ca/db/software-ca.crl.srl
```

4.3 Create CA request

```
openssl req -new \
    -config etc/software-ca.conf \
    -out ca/software-ca.csr \
    -keyout ca/software-ca/private/software-ca.key
```

We create a private key and a CSR for the software CA. The configuration is taken from the [req] section of the *software CA configuration file*.

4.4 Create CA certificate

```
openssl ca \
    -config etc/root-ca.conf \
    -in ca/software-ca.csr \
    -out ca/software-ca.crt \
    -extensions signing_ca_ext
```

We use the root CA to issue the software CA certificate.

4.5 Create initial CRL

```
openssl ca -gencrl \
    -config etc/software-ca.conf \
    -out crl/software-ca.crl
```

We create an empty CRL.

4.6 Create PEM bundle

```
cat ca/software-ca.crt ca/root-ca.crt > \
    ca/software-ca-chain.pem
```

We create a certificate chain file.

5. Operate Email CA

5.1 Create email request

```
openssl req -new \
    -config etc/email.conf \
    -out certs/fred.csr \
    -keyout certs/fred.key
```

We create the private key and CSR for an email-protection certificate using a *request configuration file*. When prompted enter these DN components: C=NO, O=Green AS, CN=Fred Flintstone, emailAddress=fred@green.no. Leave other fields empty.

5.2 Create email certificate

```
openssl ca \
    -config etc/email-ca.conf \
    -in certs/fred.csr \
    -out certs/fred.crt \
    -extensions email_ext
```

We use the email CA to issue Fred's email-protection certificate. A copy of the certificate is saved in the certificate archive under the name `ca/email-ca/01.pem` (01 being the certificate serial number in hex.)

5.3 Create PKCS#12 bundle

```
openssl pkcs12 -export \
    -name "Fred Flintstone (Email Security)" \
    -caname "Green Email CA" \
    -caname "Green Root CA" \
    -inkey certs/fred.key \
    -in certs/fred.crt \
    -certfile ca/email-ca-chain.pem \
    -out certs/fred.p12
```

We pack the private key, the certificate, and the CA chain into a PKCS#12 bundle. This format (often with a .pfx extension) is used to distribute keys and certificates to end users. The friendly names help identify individual certificates within the bundle.

5.4 Revoke certificate

```
openssl ca \
    -config etc/email-ca.conf \
    -revoke ca/email-ca/01.pem \
    -crl_reason keyCompromise
```

When Fred's laptop goes missing, we revoke his certificate.

5.5 Create CRL

```
openssl ca -gencrl \
    -config etc/email-ca.conf \
    -out crl/email-ca.crl
```

The next CRL contains the revoked certificate.

6. Operate TLS CA

6.1 Create TLS server request

```
SAN=DNS:green.no,DNS:www.green.no \
openssl req -new \
    -config etc/server.conf \
    -out certs/green.no.csr \
    -keyout certs/green.no.key
```

We create the private key and CSR for a TLS-server certificate using the appropriate *request configuration file*. When prompted enter these DN components: C=NO, O=Green AS, CN=www.green.no. The subjectAltName cannot be prompted for and must be specified as environment variable.

6.2 Create TLS server certificate

```
openssl ca \
    -config etc/tls-ca.conf \
    -in certs/green.no.csr \
    -out certs/green.no.crt \
    -extensions server_ext
```

We use the TLS CA to issue the server certificate.

6.3 Create PKCS#12 bundle

```
openssl pkcs12 -export \
    -name "green.no (Network Component)" \
    -caname "Green TLS CA" \
    -caname "Green Root CA" \
    -inkey certs/green.no.key \
    -in certs/green.no.crt \
    -certfile ca/tls-ca-chain.pem \
    -out certs/green.no.p12
```

We pack the private key, the certificate, and the CA chain into a PKCS#12 bundle for distribution.

6.4 Create TLS client request

```
openssl req -new \
    -config etc/client.conf \
    -out certs/barney.csr \
    -keyout certs/barney.key
```

We create the private key and CSR for a TLS-client certificate using the *client request configuration file*. When prompted enter these DN components: C=NO, O=Telenor AS, OU=Support, CN=Barney Rubble, emailAddress=barney@telenor.no.

6.5 Create TLS client certificate

```
openssl ca \
    -config etc/tls-ca.conf \
    -in certs/barney.csr \
    -out certs/barney.crt \
    -policy extern_pol \
    -extensions client_ext
```

We use the TLS CA to issue Barney's client certificate. Note that we must specify the 'extern' naming policy because the DN would not satisfy the default 'match' policy.

6.6 Create PKCS#12 bundle

```
openssl pkcs12 -export \
    -name "Barney Rubble (Network Access)" \
    -caname "Green TLS CA" \
    -caname "Green Root CA" \
    -inkey certs/barney.key \
    -in certs/barney.crt \
    -certfile ca/tls-ca-chain.pem \
    -out certs/barney.p12
```

We pack everything up into a PKCS#12 bundle for distribution.

6.7 Revoke certificate

```
openssl ca \
    -config etc/tls-ca.conf \
    -revoke ca/tls-ca/02.pem \
    -crl_reason affiliationChanged
```

When the support contract ends, we revoke the certificate.

6.8 Create CRL

```
openssl ca -gencrl \
    -config etc/tls-ca.conf \
    -out crl/tls-ca.crl
```

The next CRL contains the revoked certificate.

7. Operate Software CA

7.1 Create code-signing request

```
openssl req -new \
    -config etc/codesign.conf \
    -out certs/software.csr \
    -keyout certs/software.key
```

We create the private key and CSR for a code-signing certificate using another *request configuration file*. When prompted enter these DN components: C=NO, O=Green AS, OU=Green Certificate Authority, CN=Green Software Certificate.

7.2 Create code-signing certificate

```
openssl ca \
    -config etc/software-ca.conf \
    -in certs/software.csr \
    -out certs/software.crt \
    -extensions codesign_ext
```

We use the software CA to issue the code-signing certificate.

7.3 Create PKCS#12 bundle

```
openssl pkcs12 -export \
    -name "Green Software Certificate" \
    -caname "Green Software CA" \
    -caname "Green Root CA" \
    -inkey certs/software.key \
    -in certs/software.crt \
    -certfile ca/software-ca-chain.pem \
    -out certs/software.p12
```

We create a PKCS#12 bundle for distribution.

7.4 Revoke certificate

```
openssl ca \
    -config etc/software-ca.conf \
    -revoke ca/software-ca/01.pem \
    -crl_reason unspecified
```

To complete the example, we revoke the software certificate.

7.5 Create CRL

```
openssl ca -gencrl \
    -config etc/software-ca.conf \
    -out crl/software-ca.crl
```

The next CRL contains the revoked certificate.

8. Publish Certificates

8.1 Create DER certificate

```
openssl x509 \
    -in ca/root-ca.crt \
    -out ca/root-ca.cer \
    -outform der
```

All published certificates must be in DER format. MIME type: application/pkix-cert. [RFC 2585#section-4.1]

8.2 Create DER CRL

```
openssl crl \
    -in crl/email-ca.crl \
    -out crl/email-ca.crl \
    -outform der
```

All published CRLs must be in DER format. MIME type: application/pkix-crl. [RFC 2585#section-4.2]

8.3 Create PKCS#7 bundle

```
openssl crl2pkcs7 -nocrl \
    -certfile ca/email-ca-chain.pem \
    -out ca/email-ca-chain.p7c \
    -outform der
```

PKCS#7 is used to bundle two or more certificates. MIME type: application/pkcs7-mime. [RFC 5273#page-3]

References

- http://www.openssl.org/docs/apps/req.html
- http://www.openssl.org/docs/apps/ca.html
- http://www.openssl.org/docs/apps/x509.html
- http://www.openssl.org/docs/apps/crl.html
- http://www.openssl.org/docs/apps/crl2pkcs7.html
- http://www.openssl.org/docs/apps/pkcs7.html
- http://www.openssl.org/docs/apps/pkcs12.html

4.3 Expert PKI

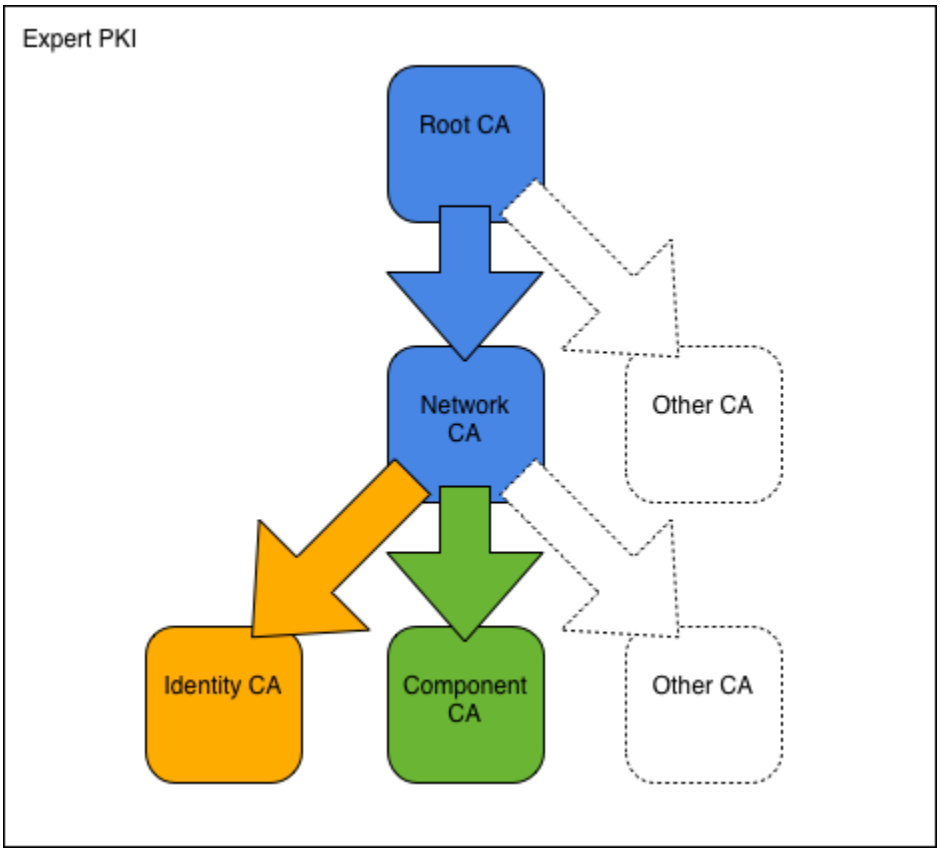
In this example we create a 3-tier CA hierarchy: One root CA, one intermediate CA, and two signing CAs. We use the CAs to issue 6 types of user certificates. We introduce certificate policies and the certificatePolicies extension. We also show how to configure an OCSP responder.

4.3.1 Expert PKI

The Expert PKI consist of a root CA, an intermediate CA, and two signing CAs.

Overview

We assume a company named **Blue AB**, controlling the domain blue.se. The company operates a flexible, multi-level PKI.



To construct the PKI, we start with the Blue Root CA followed by the intermediate Blue Network CA. We then use the Network CA to create the two signing CAs: Blue Identity CA and Blue Component CA. With the CAs in place, we proceed to issue certificates to users and network components respectively.

All commands are ready to be copy/pasted into a terminal session. When you have reached the end of this page, you will have built a PKI with both intermediate and signing CAs and issued all major types of user certificates.

To get started, fetch the Expert PKI example files and change into the new directory:

```
git clone https://github.com/mwiora/pki-example-3.git
cd pki-example-3
```

Configuration Files

We use one configuration file per CA:

Root CA Configuration File

```
# Blue Root CA

[ default ]
ca                      = root-ca               # CA name
dir                     = .                     # Top dir
base_url                = http://pki.blue.se    # CA base URL
aia_url                 = $base_url/$ca.cer     # CA certificate URL
crl_url                 = $base_url/$ca.crl     # CRL distribution point
name_opt                = multiline,-esc_msb,utf8 # Display UTF-8 characters
openssl_conf            = openssl_init          # Library config section

# CA certificate request

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = no                    # Don't prompt for DN
distinguished_name      = ca_dn                 # DN section
req_extensions          = ca_reqext             # Desired extensions

[ ca_dn ]
countryName             = "SE"
organizationName        = "Blue AB"
organizationalUnitName  = "Blue Root CA"
commonName              = "Blue Root CA"

[ ca_reqext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true
subjectKeyIdentifier    = hash

# CA operational settings

[ ca ]
default_ca              = root_ca               # The default CA section

[ root_ca ]
certificate             = $dir/ca/$ca.crt       # The CA cert
private_key             = $dir/ca/$ca/private/$ca.key # CA private key
new_certs_dir           = $dir/ca/$ca           # Certificate archive
serial                  = $dir/ca/$ca/db/$ca.crt.srl # Serial number file
crlnumber               = $dir/ca/$ca/db/$ca.crl.srl # CRL number file
database                = $dir/ca/$ca/db/$ca.db # Index file
```

(continues on next page)

(continued from previous page)

```
unique_subject          = no                    # Require unique subject
default_days            = 3652                  # How long to certify for
default_md              = sha256                # MD to use
policy                  = match_pol             # Default naming policy
email_in_dn             = no                    # Add email to cert DN
preserve                = no                    # Keep passed DN ordering
name_opt                = $name_opt             # Subject DN display options
cert_opt                = ca_default            # Certificate display options
copy_extensions         = none                  # Copy extensions from CSR
x509_extensions         = intermediate_ca_ext   # Default cert extensions
default_crl_days        = 30                    # How long before next CRL
crl_extensions          = crl_ext               # CRL extensions

[ match_pol ]
countryName             = match
stateOrProvinceName     = optional
localityName            = optional
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied

[ any_pol ]
domainComponent         = optional
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = optional
emailAddress            = optional

# Extensions

[ root_ca_ext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always

[ intermediate_ca_ext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @issuer_info
crlDistributionPoints   = @crl_info
certificatePolicies     = blueMediumAssurance,blueMediumDevice

[ crl_ext ]
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @issuer_info
```

(continues on next page)

(continued from previous page)

```
[ issuer_info ]
caIssuers;URI.0         = $aia_url

[ crl_info ]
URI.0                   = $crl_url

# Policy OIDs

[ openssl_init ]
oid_section             = additional_oids

[ additional_oids ]
blueMediumAssurance     = Blue Medium Assurance, 1.3.6.1.4.1.0.1.7.8
blueMediumDevice        = Blue Medium Device Assurance, 1.3.6.1.4.1.0.1.7.9
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/x509.html#NAME_OPTIONS
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/ca.html#CONFIGURATION_FILE_OPTIONS
- http://www.openssl.org/docs/apps/x509v3_config.html

Network CA Configuration File

```
# Blue Network CA

[ default ]
ca                      = network-ca            # CA name
dir                     = .                     # Top dir
base_url                = http://pki.blue.se    # CA base URL
aia_url                 = $base_url/$ca.cer     # CA certificate URL
crl_url                 = $base_url/$ca.crl     # CRL distribution point
name_opt                = multiline,-esc_msb,utf8 # Display UTF-8 characters
openssl_conf            = openssl_init          # Library config section

# CA certificate request

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = no                    # Don't prompt for DN
distinguished_name      = ca_dn                 # DN section
req_extensions          = ca_reqext             # Desired extensions
```

(continues on next page)

(continued from previous page)

```
[ ca_dn ]
countryName             = "SE"
organizationName        = "Blue AB"
organizationalUnitName  = "Blue Network CA"
commonName              = "Blue Network CA"

[ ca_reqext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true
subjectKeyIdentifier    = hash

# CA operational settings

[ ca ]
default_ca              = network_ca            # The default CA section

[ network_ca ]
certificate             = $dir/ca/$ca.crt       # The CA cert
private_key             = $dir/ca/$ca/private/$ca.key # CA private key
new_certs_dir           = $dir/ca/$ca           # Certificate archive
serial                  = $dir/ca/$ca/db/$ca.crt.srl # Serial number file
crlnumber               = $dir/ca/$ca/db/$ca.crl.srl # CRL number file
database                = $dir/ca/$ca/db/$ca.db # Index file
unique_subject          = no                    # Require unique subject
default_days            = 3652                  # How long to certify for
default_md              = sha256                # MD to use
policy                  = match_pol             # Default naming policy
email_in_dn             = no                    # Add email to cert DN
preserve                = no                    # Keep passed DN ordering
name_opt                = $name_opt             # Subject DN display options
cert_opt                = ca_default            # Certificate display options
copy_extensions         = none                  # Copy extensions from CSR
x509_extensions         = signing_ca_ext        # Default cert extensions
default_crl_days        = 1                     # How long before next CRL
crl_extensions          = crl_ext               # CRL extensions

[ match_pol ]
countryName             = match
stateOrProvinceName     = optional
localityName            = optional
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied

[ any_pol ]
domainComponent         = optional
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
```

(continues on next page)

(continued from previous page)

```
commonName              = optional
emailAddress            = optional

# Extensions

[ intermediate_ca_ext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @issuer_info
crlDistributionPoints   = @crl_info
certificatePolicies     = blueMediumAssurance,blueMediumDevice

[ signing_ca_ext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true,pathlen:0
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @issuer_info
crlDistributionPoints   = @crl_info
certificatePolicies     = blueMediumAssurance,blueMediumDevice

[ crl_ext ]
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @issuer_info

[ issuer_info ]
caIssuers;URI.0         = $aia_url

[ crl_info ]
URI.0                   = $crl_url

# Policy OIDs

[ openssl_init ]
oid_section             = additional_oids

[ additional_oids ]
blueMediumAssurance     = Blue Medium Assurance, 1.3.6.1.4.1.0.1.7.8
blueMediumDevice        = Blue Medium Device Assurance, 1.3.6.1.4.1.0.1.7.9
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/x509.html#NAME_OPTIONS
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/ca.html#CONFIGURATION_FILE_OPTIONS
- http://www.openssl.org/docs/apps/x509v3_config.html

Identity CA Configuration File

```
# Blue Identity CA

[ default ]
ca                      = identity-ca           # CA name
dir                     = .                     # Top dir
base_url                = http://pki.blue.se    # CA base URL
aia_url                 = $base_url/$ca.cer     # CA certificate URL
crl_url                 = $base_url/$ca.crl     # CRL distribution point
name_opt                = multiline,-esc_msb,utf8 # Display UTF-8 characters
openssl_conf            = openssl_init          # Library config section

# CA certificate request

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = no                    # Don't prompt for DN
distinguished_name      = ca_dn                 # DN section
req_extensions          = ca_reqext             # Desired extensions

[ ca_dn ]
countryName             = "SE"
organizationName        = "Blue AB"
organizationalUnitName  = "Blue Identity CA"
commonName              = "Blue Identity CA"

[ ca_reqext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true,pathlen:0
subjectKeyIdentifier    = hash

# CA operational settings

[ ca ]
default_ca              = identity_ca           # The default CA section

[ identity_ca ]
```

(continues on next page)

(continued from previous page)

```
certificate             = $dir/ca/$ca.crt       # The CA cert
private_key             = $dir/ca/$ca/private/$ca.key # CA private key
new_certs_dir           = $dir/ca/$ca           # Certificate archive
serial                  = $dir/ca/$ca/db/$ca.crt.srl # Serial number file
crlnumber               = $dir/ca/$ca/db/$ca.crl.srl # CRL number file
database                = $dir/ca/$ca/db/$ca.db # Index file
unique_subject          = no                    # Require unique subject
default_days            = 1095                  # How long to certify for
default_md              = sha256                # MD to use
policy                  = match_pol             # Default naming policy
email_in_dn             = no                    # Add email to cert DN
preserve                = no                    # Keep passed DN ordering
name_opt                = $name_opt             # Subject DN display options
cert_opt                = ca_default            # Certificate display options
copy_extensions         = copy                  # Copy extensions from CSR
x509_extensions         = identity_ext          # Default cert extensions
default_crl_days        = 1                     # How long before next CRL
crl_extensions          = crl_ext               # CRL extensions

[ match_pol ]
countryName             = match
stateOrProvinceName     = optional
localityName            = optional
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied

[ any_pol ]
domainComponent         = optional
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = optional
emailAddress            = optional

# Extensions

[ identity_ext ]
keyUsage                = critical,digitalSignature
basicConstraints        = CA:false
extendedKeyUsage        = emailProtection,clientAuth,msSmartcardLogin
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @issuer_info
crlDistributionPoints   = @crl_info
certificatePolicies     = blueMediumAssurance

[ encryption_ext ]
keyUsage                = critical,keyEncipherment
basicConstraints        = CA:false
```

(continues on next page)

(continued from previous page)

```
extendedKeyUsage        = emailProtection,msEFS
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @issuer_info
crlDistributionPoints   = @crl_info
certificatePolicies     = blueMediumAssurance

[ crl_ext ]
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @issuer_info

[ issuer_info ]
caIssuers;URI.0         = $aia_url

[ crl_info ]
URI.0                   = $crl_url

# Policy OIDs

[ openssl_init ]
oid_section             = additional_oids

[ additional_oids ]
blueMediumAssurance     = Blue Medium Assurance, 1.3.6.1.4.1.0.1.7.8
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/x509.html#NAME_OPTIONS
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/ca.html#CONFIGURATION_FILE_OPTIONS
- http://www.openssl.org/docs/apps/x509v3_config.html

Component CA Configuration File

```
# Blue Component CA

[ default ]
ca                      = component-ca          # CA name
dir                     = .                     # Top dir
base_url                = http://pki.blue.se    # CA base URL
aia_url                 = $base_url/$ca.cer     # CA certificate URL
crl_url                 = $base_url/$ca.crl     # CRL distribution point
ocsp_url                = http://ocsp.blue.se   # OCSP responder URL
name_opt                = multiline,-esc_msb,utf8 # Display UTF-8 characters
openssl_conf            = openssl_init          # Library config section
```

(continues on next page)

(continued from previous page)

```
# CA certificate request

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = no                    # Don't prompt for DN
distinguished_name      = ca_dn                 # DN section
req_extensions          = ca_reqext             # Desired extensions

[ ca_dn ]
countryName             = "SE"
organizationName        = "Blue AB"
organizationalUnitName  = "Blue Component CA"
commonName              = "Blue Component CA"

[ ca_reqext ]
keyUsage                = critical,keyCertSign,cRLSign
basicConstraints        = critical,CA:true,pathlen:0
subjectKeyIdentifier    = hash

# CA operational settings

[ ca ]
default_ca              = component_ca          # The default CA section

[ component_ca ]
certificate             = $dir/ca/$ca.crt       # The CA cert
private_key             = $dir/ca/$ca/private/$ca.key # CA private key
new_certs_dir           = $dir/ca/$ca           # Certificate archive
serial                  = $dir/ca/$ca/db/$ca.crt.srl # Serial number file
crlnumber               = $dir/ca/$ca/db/$ca.crl.srl # CRL number file
database                = $dir/ca/$ca/db/$ca.db # Index file
unique_subject          = no                    # Require unique subject
default_days            = 730                   # How long to certify for
default_md              = sha256                # MD to use
policy                  = match_pol             # Default naming policy
email_in_dn             = no                    # Add email to cert DN
preserve                = no                    # Keep passed DN ordering
name_opt                = $name_opt             # Subject DN display options
cert_opt                = ca_default            # Certificate display options
copy_extensions         = copy                  # Copy extensions from CSR
x509_extensions         = server_ext            # Default cert extensions
default_crl_days        = 1                     # How long before next CRL
crl_extensions          = crl_ext               # CRL extensions

[ match_pol ]
countryName             = match
stateOrProvinceName     = optional
localityName            = optional
```

(continues on next page)

(continued from previous page)

```
organizationName        = match
organizationalUnitName  = optional
commonName              = supplied

[ any_pol ]
domainComponent         = optional
countryName             = optional
stateOrProvinceName     = optional
localityName            = optional
organizationName        = optional
organizationalUnitName  = optional
commonName              = optional
emailAddress            = optional

# Extensions

[ server_ext ]
keyUsage                = critical,digitalSignature,keyEncipherment
basicConstraints        = CA:false
extendedKeyUsage        = serverAuth,clientAuth
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @ocsp_info
crlDistributionPoints   = @crl_info
certificatePolicies     = blueMediumDevice

[ client_ext ]
keyUsage                = critical,digitalSignature
basicConstraints        = CA:false
extendedKeyUsage        = clientAuth
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @ocsp_info
crlDistributionPoints   = @crl_info
certificatePolicies     = blueMediumDevice

[ timestamp_ext ]
keyUsage                = critical,digitalSignature
basicConstraints        = CA:false
extendedKeyUsage        = critical,timeStamping
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @issuer_info
crlDistributionPoints   = @crl_info
certificatePolicies     = blueMediumDevice

[ ocspsign_ext ]
keyUsage                = critical,digitalSignature
basicConstraints        = CA:false
extendedKeyUsage        = critical,OCSPSigning
subjectKeyIdentifier    = hash
authorityKeyIdentifier  = keyid:always
```

(continues on next page)

(continued from previous page)

```
authorityInfoAccess     = @issuer_info
noCheck                 = null
certificatePolicies     = blueMediumDevice

[ crl_ext ]
authorityKeyIdentifier  = keyid:always
authorityInfoAccess     = @issuer_info

[ ocsp_info ]
caIssuers;URI.0         = $aia_url
OCSP;URI.0              = $ocsp_url

[ issuer_info ]
caIssuers;URI.0         = $aia_url

[ crl_info ]
URI.0                   = $crl_url

# Policy OIDs

[ openssl_init ]
oid_section             = additional_oids

[ additional_oids ]
blueMediumDevice        = Blue Medium Device Assurance, 1.3.6.1.4.1.0.1.7.9
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/x509.html#NAME_OPTIONS
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/ca.html#CONFIGURATION_FILE_OPTIONS
- http://www.openssl.org/docs/apps/x509v3_config.html

And one configuration file per CSR type:

Identity Certificate Request Configuration File

```
# Identity certificate request

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = yes                   # Prompt for DN
distinguished_name      = identity_dn           # DN template
```

(continues on next page)

(continued from previous page)

```
req_extensions          = identity_reqext       # Desired extensions

[ identity_dn ]
countryName             = "1. Country Name (2 letters) (eg, US)       "
countryName_max         = 2
stateOrProvinceName     = "2. State or Province Name   (eg, region)   "
localityName            = "3. Locality Name            (eg, city)     "
organizationName        = "4. Organization Name        (eg, company)  "
organizationalUnitName  = "5. Organizational Unit Name (eg, section)  "
commonName              = "6. Common Name              (eg, full name)"
commonName_max          = 64
emailAddress            = "7. Email Address            (eg, name@fqdn)"
emailAddress_max        = 40

[ identity_reqext ]
keyUsage                = critical,digitalSignature
extendedKeyUsage        = emailProtection,clientAuth
subjectKeyIdentifier    = hash
subjectAltName          = email:move
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/x509v3_config.html

Encryption Certificate Request Configuration File

```
# Encryption certificate request

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = yes                   # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = yes                   # Prompt for DN
distinguished_name      = encryption_dn         # DN template
req_extensions          = encryption_reqext     # Desired extensions

[ encryption_dn ]
countryName             = "1. Country Name (2 letters) (eg, US)       "
countryName_max         = 2
stateOrProvinceName     = "2. State or Province Name   (eg, region)   "
localityName            = "3. Locality Name            (eg, city)     "
organizationName        = "4. Organization Name        (eg, company)  "
organizationalUnitName  = "5. Organizational Unit Name (eg, section)  "
commonName              = "6. Common Name              (eg, full name)"
```

(continues on next page)

(continued from previous page)

```
commonName_max          = 64
emailAddress            = "7. Email Address            (eg, name@fqdn)"
emailAddress_max        = 40

[ encryption_reqext ]
keyUsage                = critical,keyEncipherment
extendedKeyUsage        = emailProtection
subjectKeyIdentifier    = hash
subjectAltName          = email:move
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/x509v3_config.html

TLS Server Certificate Request Configuration File

```
# TLS server certificate request

[ default ]
SAN                     = DNS:yourdomain.tld    # Default value

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = no                    # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = yes                   # Prompt for DN
distinguished_name      = server_dn             # DN template
req_extensions          = server_reqext         # Desired extensions

[ server_dn ]
countryName             = "1. Country Name (2 letters) (eg, US)       "
countryName_max         = 2
stateOrProvinceName     = "2. State or Province Name   (eg, region)   "
localityName            = "3. Locality Name            (eg, city)     "
organizationName        = "4. Organization Name        (eg, company)  "
organizationalUnitName  = "5. Organizational Unit Name (eg, section)  "
commonName              = "6. Common Name              (eg, FQDN)     "
commonName_max          = 64

[ server_reqext ]
keyUsage                = critical,digitalSignature,keyEncipherment
extendedKeyUsage        = serverAuth,clientAuth
subjectKeyIdentifier    = hash
subjectAltName          = $ENV::SAN
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/x509v3_config.html

TLS Client Certificate Request Configuration File

```
# TLS client certificate request

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = no                    # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = yes                   # Prompt for DN
distinguished_name      = client_dn             # DN template
req_extensions          = client_reqext         # Desired extensions

[ client_dn ]
countryName             = "1. Country Name (2 letters) (eg, US)       "
countryName_max         = 2
stateOrProvinceName     = "2. State or Province Name   (eg, region)   "
localityName            = "3. Locality Name            (eg, city)     "
organizationName        = "4. Organization Name        (eg, company)  "
organizationalUnitName  = "5. Organizational Unit Name (eg, section)  "
commonName              = "6. Common Name              (eg, full name)"
commonName_max          = 64

[ client_reqext ]
keyUsage                = critical,digitalSignature
extendedKeyUsage        = clientAuth
subjectKeyIdentifier    = hash
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/x509v3_config.html

Time-Stamping Certificate Request Configuration File

```
# Time-stamping certificate request

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = no                    # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = yes                   # Prompt for DN
distinguished_name      = timestamp_dn          # DN template
req_extensions          = timestamp_reqext      # Desired extensions

[ timestamp_dn ]
countryName             = "1. Country Name (2 letters) (eg, US)       "
countryName_max         = 2
stateOrProvinceName     = "2. State or Province Name   (eg, region)   "
localityName            = "3. Locality Name            (eg, city)     "
organizationName        = "4. Organization Name        (eg, company)  "
organizationalUnitName  = "5. Organizational Unit Name (eg, section)  "
commonName              = "6. Common Name              (eg, full name)"
commonName_max          = 64

[ timestamp_reqext ]
keyUsage                = critical,digitalSignature
extendedKeyUsage        = critical,timeStamping
subjectKeyIdentifier    = hash
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/x509v3_config.html

OCSP-Signing Certificate Request Configuration File

```
# OCSP-signing certificate request

[ req ]
default_bits            = 4096                  # RSA key size
encrypt_key             = no                    # Protect private key
default_md              = sha256                # MD to use
utf8                    = yes                   # Input is UTF-8
string_mask             = utf8only              # Emit UTF-8 strings
prompt                  = yes                   # Prompt for DN
distinguished_name      = ocspsign_dn           # DN template
req_extensions          = ocspsign_reqext       # Desired extensions

```

(continues on next page)

(continued from previous page)

```
[ ocspsign_dn ]
countryName             = "1. Country Name (2 letters) (eg, US)       "
countryName_max         = 2
stateOrProvinceName     = "2. State or Province Name   (eg, region)   "
localityName            = "3. Locality Name            (eg, city)     "
organizationName        = "4. Organization Name        (eg, company)  "
organizationalUnitName  = "5. Organizational Unit Name (eg, section)  "
commonName              = "6. Common Name              (eg, full name)"
commonName_max          = 64

[ ocspsign_reqext ]
keyUsage                = critical,digitalSignature
extendedKeyUsage        = critical,OCSPSigning
subjectKeyIdentifier    = hash
```

References

- http://www.openssl.org/docs/apps/config.html
- http://www.openssl.org/docs/apps/req.html#CONFIGURATION_FILE_FORMAT
- http://www.openssl.org/docs/apps/x509v3_config.html

Please study the configuration files before you continue.

1. Create Root CA

1.1 Create directories

```
mkdir -p ca/root-ca/private ca/root-ca/db crl certs
chmod 700 ca/root-ca/private
```

1.2 Create database

```
cp /dev/null ca/root-ca/db/root-ca.db
cp /dev/null ca/root-ca/db/root-ca.db.attr
echo 01 > ca/root-ca/db/root-ca.crt.srl
echo 01 > ca/root-ca/db/root-ca.crl.srl
```

1.3 Create CA request

```
openssl req -new \
    -config etc/root-ca.conf \
    -out ca/root-ca.csr \
    -keyout ca/root-ca/private/root-ca.key
```

1.4 Create CA certificate

```
openssl ca -selfsign \
    -config etc/root-ca.conf \
    -in ca/root-ca.csr \
    -out ca/root-ca.crt \
    -extensions root_ca_ext \
    -enddate 20301231235959Z
```

2048-bit RSA keys are deemed safe until 2030 (RSA Labs).

1.5 Create initial CRL

```
openssl ca -gencrl \
    -config etc/root-ca.conf \
    -out crl/root-ca.crl
```

2. Create Network CA

2.1 Create directories

```
mkdir -p ca/network-ca/private ca/network-ca/db crl certs
chmod 700 ca/network-ca/private
```

2.2 Create database

```
cp /dev/null ca/network-ca/db/network-ca.db
cp /dev/null ca/network-ca/db/network-ca.db.attr
echo 01 > ca/network-ca/db/network-ca.crt.srl
echo 01 > ca/network-ca/db/network-ca.crl.srl
```

2.3 Create CA request

```
openssl req -new \
    -config etc/network-ca.conf \
    -out ca/network-ca.csr \
    -keyout ca/network-ca/private/network-ca.key
```

2.4 Create CA certificate

```
openssl ca \
    -config etc/root-ca.conf \
    -in ca/network-ca.csr \
    -out ca/network-ca.crt \
    -extensions intermediate_ca_ext \
    -enddate 20301231235959Z
```

Intermediate CAs should have the same life span as their root CAs.

2.5 Create initial CRL

```
openssl ca -gencrl \
    -config etc/network-ca.conf \
    -out crl/network-ca.crl
```

2.6 Create PEM bundle

```
cat ca/network-ca.crt ca/root-ca.crt > \
    ca/network-ca-chain.pem
```

3. Create Identity CA

3.1 Create directories

```
mkdir -p ca/identity-ca/private ca/identity-ca/db crl certs
chmod 700 ca/identity-ca/private
```

3.2 Create database

```
cp /dev/null ca/identity-ca/db/identity-ca.db
cp /dev/null ca/identity-ca/db/identity-ca.db.attr
echo 01 > ca/identity-ca/db/identity-ca.crt.srl
echo 01 > ca/identity-ca/db/identity-ca.crl.srl
```

3.3 Create CA request

```
openssl req -new \
    -config etc/identity-ca.conf \
    -out ca/identity-ca.csr \
    -keyout ca/identity-ca/private/identity-ca.key
```

3.4 Create CA certificate

```
openssl ca \
    -config etc/network-ca.conf \
    -in ca/identity-ca.csr \
    -out ca/identity-ca.crt \
    -extensions signing_ca_ext
```

3.5 Create initial CRL

```
openssl ca -gencrl \
    -config etc/identity-ca.conf \
    -out crl/identity-ca.crl
```

3.6 Create PEM bundle

```
cat ca/identity-ca.crt ca/network-ca-chain.pem > \
    ca/identity-ca-chain.pem
```

4. Create Component CA

4.1 Create directories

```
mkdir -p ca/component-ca/private ca/component-ca/db crl certs
chmod 700 ca/component-ca/private
```

4.2 Create database

```
cp /dev/null ca/component-ca/db/component-ca.db
cp /dev/null ca/component-ca/db/component-ca.db.attr
echo 01 > ca/component-ca/db/component-ca.crt.srl
echo 01 > ca/component-ca/db/component-ca.crl.srl
```

4.3 Create CA request

```
openssl req -new \
    -config etc/component-ca.conf \
    -out ca/component-ca.csr \
    -keyout ca/component-ca/private/component-ca.key
```

4.4 Create CA certificate

```
openssl ca \
    -config etc/network-ca.conf \
    -in ca/component-ca.csr \
    -out ca/component-ca.crt \
    -extensions signing_ca_ext
```

4.5 Create initial CRL

```
openssl ca -gencrl \
    -config etc/component-ca.conf \
    -out crl/component-ca.crl
```

4.6 Create PEM bundle

```
cat ca/component-ca.crt ca/network-ca-chain.pem > \
    ca/component-ca-chain.pem
```

5. Operate Identity CA

5.1 Create identity request

```
openssl req -new \
    -config etc/identity.conf \
    -out certs/fred-id.csr \
    -keyout certs/fred-id.key
```

DN: C=SE, O=Blue AB, CN=Fred Flintstone, emailAddress=fred@blue.se

5.2 Create identity certificate

```
openssl ca \
    -config etc/identity-ca.conf \
    -in certs/fred-id.csr \
    -out certs/fred-id.crt \
    -extensions identity_ext
```

5.3 Create PKCS#12 bundle

```
openssl pkcs12 -export \
    -name "Fred Flintstone (Blue Identity)" \
    -caname "Blue Identity CA" \
    -caname "Blue Network CA" \
    -caname "Blue Root CA" \
    -inkey certs/fred-id.key \
    -in certs/fred-id.crt \
    -certfile ca/identity-ca-chain.pem \
    -out certs/fred-id.p12
```

5.4 Create encryption request

```
openssl req -new \
    -config etc/encryption.conf \
    -out certs/fred-enc.csr \
    -keyout certs/fred-enc.key
```

DN: C=SE, O=Blue AB, CN=Fred Flintstone, emailAddress=fred@blue.se

5.5 Create encryption certificate

```
openssl ca \
    -config etc/identity-ca.conf \
    -in certs/fred-enc.csr \
    -out certs/fred-enc.crt \
    -extensions encryption_ext
```

5.6 Create PKCS#12 bundle

```
openssl pkcs12 -export \
    -name "Fred Flintstone (Blue Encryption)" \
    -caname "Blue Identity CA" \
    -caname "Blue Network CA" \
    -caname "Blue Root CA" \
    -inkey certs/fred-enc.key \
    -in certs/fred-enc.crt \
```

(continues on next page)

(continued from previous page)

```
    -certfile ca/identity-ca-chain.pem \
    -out certs/fred-enc.p12
```

5.7 Revoke certificate

```
openssl ca \
    -config etc/identity-ca.conf \
    -revoke ca/identity-ca/02.pem \
    -crl_reason superseded
```

5.8 Create CRL

```
openssl ca -gencrl \
    -config etc/identity-ca.conf \
    -out crl/identity-ca.crl
```

6. Operate Component CA

6.1 Create TLS server request

```
SAN=DNS:blue.se,DNS:www.blue.se \
openssl req -new \
    -config etc/server.conf \
    -out certs/blue.se.csr \
    -keyout certs/blue.se.key
```

DN: C=SE, O=Blue AB, CN=www.blue.se

6.2 Create TLS server certificate

```
openssl ca \
    -config etc/component-ca.conf \
    -in certs/blue.se.csr \
    -out certs/blue.se.crt \
    -extensions server_ext
```

6.3 Create TLS client request

```
openssl req -new \
    -config etc/client.conf \
    -out certs/net-mon.csr \
    -keyout certs/net-mon.key
```

DN: C=SE, O=Blue AB, CN=Blue Network Monitoring

6.4 Create TLS client certificate

```
openssl ca \
    -config etc/component-ca.conf \
    -in certs/net-mon.csr \
    -out certs/net-mon.crt \
    -extensions client_ext
```

6.5 Create time-stamping request

```
openssl req -new \
    -config etc/timestamp.conf \
    -out certs/tsa.csr \
    -keyout certs/tsa.key
```

DN: C=SE, O=Blue AB, OU=Blue TSA, CN=Blue TSA

6.6 Create time-stamping certificate

```
openssl ca \
    -config etc/component-ca.conf \
    -in certs/tsa.csr \
    -out certs/tsa.crt \
    -extensions timestamp_ext \
    -days 1826
```

6.7 Create OCSP-signing request

```
openssl req -new \
    -config etc/ocspsign.conf \
    -out certs/ocsp.csr \
    -keyout certs/ocsp.key
```

DN: C=SE, O=Blue AB, CN=Blue OCSP Responder

6.8 Create OCSP-signing certificate

```
openssl ca \
    -config etc/component-ca.conf \
    -in certs/ocsp.csr \
    -out certs/ocsp.crt \
    -extensions ocspsign_ext \
    -days 14
```

6.9 Revoke certificate

```
openssl ca \
    -config etc/component-ca.conf \
    -revoke ca/component-ca/02.pem \
    -crl_reason superseded
```

6.10 Create CRL

```
openssl ca -gencrl \
    -config etc/component-ca.conf \
    -out crl/component-ca.crl
```

7. Publish Certificates

7.1 Create DER certificate

```
openssl x509 \
    -in ca/root-ca.crt \
    -out ca/root-ca.cer \
    -outform der
```

All published certificates must be in DER format. MIME type: application/pkix-cert. [RFC 2585#section-4.1]

7.2 Create DER CRL

```
openssl crl \
    -in crl/network-ca.crl \
    -out crl/network-ca.crl \
    -outform der
```

All published CRLs must be in DER format. MIME type: application/pkix-crl. [RFC 2585#section-4.2]

7.3 Create PKCS#7 bundle

```
openssl crl2pkcs7 -nocrl \
    -certfile ca/identity-ca-chain.pem \
    -out ca/identity-ca-chain.p7c \
    -outform der
```

PKCS#7 is used to bundle two or more certificates. MIME type: application/pkcs7-mime. [RFC 5273#page-3]

References

- http://www.openssl.org/docs/apps/req.html
- http://www.openssl.org/docs/apps/ca.html
- http://www.openssl.org/docs/apps/x509.html
- http://www.openssl.org/docs/apps/crl.html
- http://www.openssl.org/docs/apps/crl2pkcs7.html
- http://www.openssl.org/docs/apps/pkcs7.html
- http://www.openssl.org/docs/apps/pkcs12.html

CHAPTER

FIVE

APPENDICES

5.1 MIME Types

This section takes a closer look at the MIME types and file extensions used.

5.1.1 Appendix A: MIME Types

File Types

When operating a PKI we deal with only a handful of file types:

1. PKCS#8 private keys
2. PKCS#10 CSRs
3. X.509 certificates
4. X.509 CRLs
5. PKCS#7 bundles of two or more certificates
6. PKCS#12 bundles of private key + certificate(s)

MIME Types

The list of MIME types and file extensions however is easily twice as long:

```
application/pkcs8                   .p8  .key
application/pkcs10                  .p10 .csr
application/pkix-cert               .cer
application/pkix-crl                .crl
application/pkcs7-mime              .p7c

application/x-x509-ca-cert          .crt .der
application/x-x509-user-cert        .crt
application/x-pkcs7-crl             .crl

application/x-pem-file              .pem
application/x-pkcs12                .p12 .pfx

application/x-pkcs7-certificates    .p7b .spc
application/x-pkcs7-certreqresp     .p7r
```

Where do they come from?

1. pkcs8 and the .p8 extension are defined in **RFC 5958#section-7.1**. The .key extension is Apache mod_ssl practice.[1]
2. pkcs10 and the .p10 extension are defined in **RFC 5967#section-3.1**. The .csr extension is Apache mod_ssl practice.
3. pkix-cert and the .cer extension are defined in **RFC 2585#section-4.1**.
4. pkix-crl and the .crl extension are defined in **RFC 2585#section-4.2** as well.
5. pkcs7-mime and the .p7c extension are defined in **RFC 5273#page-3**.
6. x-x509-ca-cert and the .crt extension were introduced by Netscape. File contents are the same as with pkix-cert: a DER encoded X.509 certificate. [**RFC 5280#section-4**][2]
7. x-x509-user-cert was also introduced by Netscape. It is used to install certificates into (some) browsers.
8. x-pkcs7-crl was introduced by Netscape as well. Note that the .crl extension conflicts with pkix-crl. File contents are the same in either case: a DER encoded X.509 CRL. [**RFC 5280#section-5**][3]
9. x-pem-file and the .pem extension stem from a predecessor of S/MIME: Privacy Enhanced Mail.
10. x-pkcs12 and the .p12 extension are used for PKCS#12 files. The .pfx extension is a relic from a predecessor of PKCS#12. It is still used in Microsoft environments (the extension not the format.)
11. x-pkcs7-certificates as well as the .p7b and .spc extensions were introduced by Microsoft. File contents are the same as with pkcs7-mime: a DER encoded certs-only PKCS#7 bundle. [**RFC 2315#section-9.1**]
12. x-pkcs7-certreqresp and the .p7r extension were also introduced by Microsoft. Likely yet another alias for pkcs7-mime.

5.2 CA Database

This section examines the format of the CA database.

5.2.1 Appendix B: CA Database

Index File

The index file consists of zero or more lines, each containing the following fields separated by tab characters:

1. Certificate status flag (V=valid, R=revoked, E=expired).
2. Certificate expiration date in YYMMDDHHMMSSZ format.
3. Certificate revocation date in YYMMDDHHMMSSZ[,reason] format. Empty if not revoked.
4. Certificate serial number in hex.
5. Certificate filename or literal string 'unknown'.
6. Certificate distinguished name.

The `openssl ca` command uses this file as certificate database.

[1] The presence of a MIME type does not imply the respective files should be published on the Internet. In particular, you will never want to expose files containing private keys (.p8, .p12).

[2] Since OpenSSL defaults to PEM encoding, almost all open-source software uses PEM formatted .crt files locally. See Apache mod_ssl, stunnel, etc.

[3] This is a plain CRL and not PKCS#7 wrapped. The MIME type says 'pkcs7' for historical reasons only.

Attribute File

The attribute file contains a single line: `unique_subject = no`. It reflects the setting in the CA section of the configuration file at the time the first record is added to the database.

Serial Number Files

The `openssl ca` command uses two serial number files:

1. Certificate serial number file.
2. CRL number file.

The files contain the next available serial number in hex.

Limitations

1. The entire database must fit into memory.
2. There are no provisions for concurrency handling.

CHAPTER

SIX

REFERENCES

RFC 5280 Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile

RFC 2585 Internet X.509 Public Key Infrastructure Operational Protocols: FTP and HTTP

RFC 5750 Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Certificate Handling

RFC 6125 Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)

Baseline Requirements [pdf, opens in browser] CA/Browser Forum Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates

X.509 Recommendation [pdf, direct download] ITU-T X.509 Public-Key and Attribute Certificate Frameworks Recommendation

OpenSSL TEST CA [pdf, direct download] Carillon Information Security: How to Set Up an OpenSSL TEST CA for Interoperability Testing with CertiPath

INDEX

R