## Making your own Certificate Authority

Bitcoin Australia uses its own Certificate Authority (CA) for its security. The reason that we chose to make our own CA was initially that we didn’t want to spend the $50 to get a singed TLS key. However, there are other benefits to having your own CA, such as non-forgeable certificates.

It is common practice for governments and other dubious organizations to gain ‘special’ certificates that allow them to impersonate any other ‘secure’ websites on the internet. When a group uses their own self-signed root CA, they have complete control over who gets to generate keys from this root.

With the upcoming namecoin and other systems like that, a website having their own CA allows them to have much greater control over the secure identity of their website.

In this guide we explain the rough steps required to make a secure CA root, intermediately, and end-point certificates, and guide you through the basic steps in building the supporting infrastructure.

### Getting Started

First, for production, the computer that you are generating and using the private keys for is required to be both physically and digitally secure. There are many guides on the internet that explain the, generally nontrivial, process required to make a secure system; we suggest you find a good one and use it before continuing. However, for practice or testing purposes, it is fine to do this process on any computer.

#### Required Software

We are using the free and open source crypto package called ‘OpenSSL,’ this is installed standard on virtually all UNIX based destitutions, however for windows please find some trusted binaries.

We recommend that you use the latest version (at the time of writing v. 1.0.0d), as we take advantage of some of the more advanced new features for extra security.

**No unencrypted private key data is ever saved to the hard drive in the steps described in this guide. So having an encrypted file-system is not necessary.**

#### Operating System Notes

This guide is os-agnostic.

You will be required to do this for both the commands, and the configuration file.

### First Steps

In this quite we use we the example of our own CA.

#### Making a certificate directory structure

The certificate making process requires a logical arrangement of directories for the placement of configuration files and the certificates. The structure of the OpenSSL working directory is defined in the configuration openssl configuration file. We have this configuration so everything is set to relative locations (to pwd).

**[ OpensslWorkingDir ]** #OpenSSL Working Directory

**[ BitcoinAustralia ]** #CA Working Directory

**Openssl\_BitcoinAustralia.cfg** #CA Configuration File

**[ certs ]** #Contains generated certificates

**[ database ]** #Contains data files for OpenSSL

**index.txt** #Blank File

**serial.txt** #File with “01” and new line

**[ revoked ]** #Contains the revocation certificates

**crlnumber.txt** #File with “01” and new line

**[ private ]** #Contains the private keys

**[ requests ]** #Contains the certificate requests

#### Making the Configuration File

This configuration file is where OpenSSL gets all of its settings. It is possible to complete everything via various switches, however the commands quickly become too-long and unmanageable.

There is a default configuration file that is included with your installation of OpenSSL; for our purpose this configuration is broken. So we will be using the “-config” switch to override it.

##### Environmental Variables:

This section is included to make OpenSSL run.

HOME = .

RANDFILE = $ENV::HOME/.rnd

##### Requests

This section defines how OpenSSL behaves in the ‘req’ mode. This is used when we create our root CA, and create requests to be signed by the ‘Certificate Authority’

We have defined 3 different modes of configuration within this section:

‘v3\_ca\_default’ – where we make a root self-signed certificate. This certificate is the basis of the security of the entire CA. We use this certificate to make other intermediate certificates; this certificate does not directly interact with any public request.

‘v3\_req\_ca\_option’ – where we make requests for intermediate certificates, such as what a business would obtain for all their servers.

‘ v3\_req\_default’ – where we make requests for endpoints, such as a website server would make.

##################### req #####################

[ req ]

default\_md = sha256

default\_bits = 4096

default\_keyfile = privkey.pem

utf8 = yes

string\_mask = utf8only

distinguished\_name = req\_distinguished\_name

req\_extensions = v3\_req\_default

x509\_extensions = v3\_ca\_default

[ req\_distinguished\_name ]

countryName = Country Name (2 letter code)

countryName\_default = AU

countryName\_min = 2

countryName\_max = 2

stateOrProvinceName = State, Province or County (full name)

stateOrProvinceName\_default = Victoria

localityName = Locality Name (eg, city)

localityName\_default = Melbourne

organizationName = Organisation Name (eg, company)

organizationName\_default = Bitcoin Australia

organizationalUnitName = Organisational Unit Name (eg, section)

commonName = Common Name (eg, YOUR name)

commonName\_max = 64

emailAddress = Email Address

emailAddress\_max = 64

#### To make End Cert Request ####

[ v3\_req\_default ]

basicConstraints = CA:FALSE

keyUsage = digitalSignature, keyEncipherment, keyAgreement

extendedKeyUsage = serverAuth

subjectKeyIdentifier = hash

#### To make Intermediary CA Request ####

[ v3\_req\_ca\_option ]

basicConstraints = CA:TRUE

keyUsage = keyCertSign, cRLSign

subjectKeyIdentifier = hash

crlDistributionPoints = URI:http://cert.bitcoin.com.au/ba\_cacert-crl.crl,URI:http://crl.bitcoin.com.au/ba\_cacert-crl.crl

authorityInfoAccess = caIssuers;URI:http://cert.bitcoin.com.au/ba\_cacert.crt

#### To make Root CA ####

[ v3\_ca\_default ]

basicConstraints = CA:TRUE

keyUsage = digitalSignature, keyEncipherment, keyAgreement, keyCertSign, cRLSign

subjectKeyIdentifier = hash

extendedKeyUsage = serverAuth, clientAuth, emailProtection, codeSigning, timeStamping

crlDistributionPoints = URI:http://cert.bitcoin.com.au/ba\_cacert-crl.crl,URI:http://crl.bitcoin.com.au/ba\_cacert-crl.crl

##### Certificate Authority:

This section defines how OpenSSL behaves when it is in ‘ca’ mode. In this mode OpenSSL becomes a certificate authority where it signs certificate requests or revokes existing certificates.

We have defined two different configurations for this:

‘ca\_root’ – Using the core root certificate to sign making intermediary certificates

‘ca\_intermediary\_1’ – Using the first defined intermediary certificate (there can be many), to complete requests for domains and other such purpose.

##################### ca #####################

[ ca ]

default\_ca = ca\_intermediary\_1

#### Use the root CA to sign Intermediary CA request ####

[ ca\_root ]

dir = .

certs = $dir/certs

database = $dir/database/index.txt

new\_certs\_dir = $certs

serial = $dir/database/serial.txt

crl\_dir = $dir/revoked

default\_days = 365

default\_crl\_days = 30

default\_md = sha256

name\_opt = ca\_default

cert\_opt = ca\_default

preserve = no

email\_in\_dn = no

copy\_extensions = copy

policy = policy\_match\_ca\_option

crl\_extensions = crl\_ext\_default

crlnumber = $crl\_dir/crlnumber.txt

certificate = $certs/ba\_cacert.cer

private\_key = $dir/private/ba\_cakey.pem

x509\_extensions = v3\_ca\_option

[ policy\_match\_ca\_option ]

countryName = optional

stateOrProvinceName = optional

organizationName = match

organizationalUnitName = supplied

commonName = supplied

emailAddress = optional

[ v3\_ca\_option ]

basicConstraints = CA:TRUE

keyUsage = keyCertSign, cRLSign

subjectKeyIdentifier = hash

#### Use the Intermediary CA 1 to sign end cert request ####

[ ca\_intermediary\_1 ]

dir = .

certs = $dir/certs

database = $dir/database/index.txt

new\_certs\_dir = $certs

serial = $dir/database/serial.txt

crl\_dir = $dir/revoked

default\_days = 365

default\_crl\_days = 30

default\_md = sha256

name\_opt = ca\_default

cert\_opt = ca\_default

preserve = no

email\_in\_dn = no

copy\_extensions = copy

policy = policy\_match\_default

crl\_extensions = crl\_ext\_default

crlnumber = $crl\_dir/ crlnumber.txt

certificate = $certs/ba\_crt1.cer

private\_key = $dir/private/ba\_crt1key.pem

x509\_extensions = v3\_user\_default

[ policy\_match\_default ]

countryName = supplied

stateOrProvinceName = supplied

organizationName = match

organizationalUnitName = supplied

commonName = supplied

emailAddress = optional

[ crl\_ext\_default ]

authorityKeyIdentifier = keyid, issuer

[ v3\_user\_default ]

basicConstraints = CA:FALSE

subjectKeyIdentifier = hash

crlDistributionPoints = URI:URI:http://crl.bitcoin.com.au/ba\_crt1-crl.crl

authorityInfoAccess = caIssuers;URI:http://cert.bitcoin.com.au/ba\_crt1.crt

### Making your Root Certificate

The root certificate is the core of the entire Certificate Authority

To make this we first naviagate to:

**OpensslWorkingDir / BitcoinAustralia**

We then execute the command to make our root private key:

**openssl ecparam -text -name prime256v1 -genkey | openssl pkcs8 -topk8 -v2 aes256 -out private/ba\_cakey.pem**

Enter a very long and unique secure password; keep this password private and encrypted.

For example: oZF4d1DKNzYq1gow563e3q4PvHtmSoOx (Ha. Ha. We didn’t use this password)

We next make a public certificate from this private key:

**openssl req -config Openssl\_BitcoinAustralia.cfg -new -x509 -extensions v3\_ca\_root -days 3650 -key private/ba\_cakey.pem -out certs/ba\_cacert.cer**

When requested supply the data you want, we supplied:

.

.

.

Bitcoin Australia

Secure Digital Certificate Signing

Bitcoin Australia Certification Authority

.

Done! Your root certificate is all ready to go!!

### Making your first Intermediate Certificate

The intermediate certificate is used to create certificates for your websites and servers.

Make another private key:

**openssl ecparam -text -name prime256v1 -genkey | openssl pkcs8 -topk8 -v2 aes256 -out private/ba\_crt1key.pem**

Make a request to be signed by the root certificate:

**openssl req -config Openssl\_BitcoinAustralia.cfg -reqexts v3\_req\_ca\_option -new -key private/ba\_crt1key.pem -out requests/ba\_crt1.req**

Enter the details when requested: (We entered)

.

.

.

Bitcoin Australia

Secure Digital Certificate Signing

Bitcoin Australia 1 Primary Intermediate

.

Use OpenSSL to sign request with the root key:

**openssl ca -config Openssl\_BitcoinAustralia.cfg -name ca\_root -in requests/ba\_crt1.req -out certs/ba\_crt1.cer**

Clean Up certificate:

**openssl x509 -in certs/ba\_crt1.cer -out certs/ba\_crt1.cer**

All Done!

### Processing a 2nd party end-use certificate request

Using the intermediate certificate.

Load up your request certificate into /requests/

Use OpenSSL to sign request with the intermediate key.

**openssl ca -config Openssl\_BitcoinAustralia.cfg -name ca\_intermediary\_1 -in requests/bitcoin.com.au.req -out certs/bitcoin.com.au.cer**

Clean Up Certificate:

**openssl x509 -in certs/bitcoin.com.au.cer -out certs/bitcoin.com.au.cer**

### Create the reverocation lists:

**openssl ca -config Openssl\_BitcoinAustralia.cfg -name ca\_root -gencrl -out revoked/ba\_cacert-crl.crl**

**openssl ca -config Openssl\_BitcoinAustralia.cfg -name ca\_intermediary\_1 -gencrl -out revoked/ba\_crt1-crl.crl**

### Revoke Cert

openssl ca -config Openssl\_BitcoinAustralia.cfg -name ca\_intermediary\_1 -revoke certs\bitcoin.com.au.cer

### Make an endpoint cert:

openssl ecparam -text -name prime256v1 -genkey | openssl pkcs8 -topk8 -v2 aes256 -out private/bitcoin.com.aukey.pem

openssl req -config Openssl\_BitcoinAustralia.cfg -reqexts v3\_req\_default -new -key private/bitcoin.com.aukey.pem -out requests/bitcoin.com.au.req