1 Create X.509 certificates with OpenSSL

Document-ID: 108395_en_00
Document-Description: AH EN X.509 CERT OPENSSL

Make sure you always use the latest documentation.
It can be downloaded using the following link phoenixcontact.net/products.

Contents of this document

This section explains briefly how to create X. 509 certificates using the tool *OpenSSL*.

1.1 Introduction

The enrollment of certificates requires a certification authority (CA) which issues public key certificates for a specific period of time. A CA can be a private (in-house) CA, run by your own organization, or a public CA. A public CA is operated by a third party that you trust to validate the identity of each client or server to which it issues a certificate.

There are several tools available for creating and managing certificates, as for example *Microsoft Certification Authority (CA) Server*, *OpenSSL* and *XCA*.

This application note explains how to create X.509 certificates with the tools **OpenSSL** and **XCA** for setting up a VPN connection using X.509 certificates as authentication method.

The scope of this document is not to be a complete user's guide for the described tools. It shall help you getting familiar with them and to create the required certificates in a short term.

1.1.1 Introduction OpenSSL

OpenSSL is available for several platforms (Linux, UNIX, Windows) and can be downloaded from the Internet. We have used *OpenSSL 1.1.0e* on a *Windows 7* platform. Please refer to http://www.openssl.org for getting further information about OpenSSL and the supported command line options.

OpenSSL provides various ways for specifying the required options. You can enter them at the command line, specify them in a configuration file or you'll be prompted to enter them when the *openssl* command is executed. When using configuration files, you can either

specify all required parameters in one single file or use different ones, depending on which kind of certificate you want to create. The OpenSSL configuration file, which comes with OpenSSL, is called *openssl.cnf*.

Please note that Windows hides the file extension *.cnf*, even if you have configured the *Windows Explorer* not to do so. Therefore we use the extension *.conf*.

In the following chapters we will explain how to setup OpenSSL to act as certification authority (CA). A certificate request must be signed by the CA to become a valid certificate.

Basically you can use the examples of the following chapters for creating the certificates. You only need to follow the instructions and adjust the parameters in the section *req_dn* of the OpenSSL configuration file *openssl.conf* (see chapter “Modifying the OpenSSL configuration file” on page 4) to your company needs.

Here is a small legend with **file extensions** we will use for the created files and their meaning.

File extension	Explanation
key	Private key Restrictive permissions should be set on these files.
csr	Certificate Request The request will be signed by the CA in order to create the certificate. After doing this, the file is not needed anymore and can be deleted.
crt	Certificate This certificate can be publicly distributed.
p12	PKCS#12 export of the certificate, containing its private and public key. The export file is secured by a password to protect the private key against unauthorized usage. This certificate may not be distributed publicly.

1.2 Preparing the CA environment

First of all we will create a directory structure where all certificate stuff will be kept. In the following examples we use **C:\CA** as root directory. The following subdirectories need to be created:

Subdirectory	Purpose
.\certs	Directory where the certificates will be placed.
.\newcerts	Directory where OpenSSL puts the created certificates in PEM format as *<cert serial number>.pem* (e.g. *07.pem*). OpenSSL requires this directory.
.\private	Directory for storing the private keys. Ensure that you set restrictive permissions to this directory so that they can be read only by user with the appropriate privileges.

Apart of the directory tree, the following two files (*index.txt* and *serial*) need to be created:

- **index.txt**: This file is used as certificate “database” by OpenSSL. To create this file, proceed as follows:
  - Open a DOS prompt.
  - Switch to the CA root directory (in our example *C:\CA*).
  - Execute the command: *copy NUL: index.txt*
    This command creates the empty file *index.txt*.
- **serial**: This file contains the certificate serial number counter. This counter will be incremented automatically by OpenSSL when its value has been used for creating a certificate. To create this file, proceed as follows:
  - Open a DOS prompt.
  - Switch to the CA root directory (in our example *C:\CA*).
  - Execute the command: *echo 0001 > serial*
    This command creates the file *serial* with the initial serial number 0001.

1.3 Modifying the OpenSSL configuration file

We have named the OpenSSL configuration file *openssl.conf* and placed it into the CA root directory (in our example *C:\CA*). The OpenSSL configuration file has multiple sections. Each section is used for a different purpose. The sections include the following positions:

– **ca, CA_default**: Defines certification authority configuration.
– **policy_any**: Defines request policies.
– **req, req_dn**: Defines request defaults.

In our examples the configuration file (*openssl.conf*) has the following entries:

```
[ req ]
prompt                          = yes
default_bits                    = 4096
distinguished_name              = req_dn
x509_extensions                 = req_ext
string_mask                     = utf8only

[ ca ]
default_ca                      = CA_default

[ CA_default ]
dir                             = C:/CA
certs                           = $dir/certs
database                        = $dir/index.txt
new_certs_dir                   = $dir/newcerts

certificate                     = $dir/certs/ca.crt
serial                          = $dir/serial
private_key                     = $dir/private/ca.key

default_md                      = sha256
default_days                    = 365

x509_extensions                 = req_ext
policy                          = policy_any

[ req_dn ]
countryName                     = Country Name (2 letter code)
countryName_default             = DE

organizationName                = Organization Name (company)
organizationName_default        = PHOENIX CONTACT Cyber Security AG

organizationalUnitName          = Organizational Unit Name (department, division)
organizationalUnitName_default  = Support

commonName                      = Common Name (hostname, IP, or your name)

# Not used in our example
#emailAddress                   = Email Address
#localityName                   = Locality Name (city, district)
#stateOrProvinceName            = State or Province Name (full name)

[ policy_any ]
countryName                     = supplied
organizationName                = supplied
organizationalUnitName          = optional
commonName                      = supplied
# Not used in our example
#emailAddress                   = optional
#localityName                   = optional
#stateOrProvinceName            = optional

[ req_ext ]
basicConstraints                = critical, CA:false

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

Section	Option	Description
[req]	This section is called when requesting a certificate by calling the *openssl* command wth the option **req**.	
	prompt	If set to the value **no** this disables prompting of certificate fields and just takes values from the configuration file directly. You should enable this option for being able to enter the *common name* or to modify the default values of the certificate's distinguished name for each requested certificate.
	default_bits	This specifies the default key size in bits. If not specified then 512 is used.
	distinguished_name	This specifies the section containing the distinguished name fields to prompt for when generating a certificate or certificate request. In our example this section is called **[req_dn]**.
	x509_extensions	This specifies the configuration file section containing a list of extensions to add to certificate generated when the **-x509** switch is used. It can be overridden by the **-extensions** command line switch.
	string_mask	This option masks out the use of certain string types in certain fields. If the **utf8only** option is used then only UTF8Strings will be used: this is the PKIX recommendation in RFC2459 after 2003.
[ca]	This section is called when signing certificate requests by calling the *openssl* command with the option **ca**.	
	default_ca	If the **-name** command line option is used, then it names the section to be used. Otherwise the section to be used must be named in the **default_ca** option of the **ca** section of the configuration file, in our example **[CA_default]**.

[CA_default]	This section is called when signing certificate requests by calling the *openssl* command with the option **ca**, referenced by the **default_ca** option of the **ca** section.	
	dir	Root directory of the CA environment. If the configuration file is located in this directory and if you execute all *openssl* commands from this directory, you simply can specify “dir = .”.
	certs	Certificates output directory.
	database	The text database file to use (mandatory parameter). This file must be present though initially it will be empty.
	new_certs_dir	It specifies the directory where new certificates will be placed. Mandatory.
	certificate	Location and filename of the CA certificate.
	serial	A text file containing the next serial number to use in hex. Mandatory. This file must be present and contain a valid serial number.
	private_key	Location and filename of the file which contains the CA’s private key.
	default_md	This option specifies the digest algorithm to use. Any digest supported by the OpenSSL *dgst* command can be used.
	default_days	The default number of days the certificate will be valid. This default value can be overridden by the **-days** command line switch.
	x509_extensions	This specifies the configuration file section containing a list of extensions to add to certificate generated when the **-x509** switch is used. It can be overridden by the **-extensions** command line switch.
[req_dn]	This specifies the parameters containing the distinguished name fields to prompt for when generating a certificate or certificate request, referenced by the **distinguished_name** option of the **req** section. If the **prompt** option in the **req** section is absent or set to **yes** then the section contains field prompting information. <fieldname> is the field name being used, for example commonName (or CN).	
	<fieldname> = “prompt”	The "prompt" string is used to ask the user to enter the relevant details.
	<fieldname>_default =”default field value”	If the user enters nothing then the default value is used if no default value is present then the field is omitted.

[policy_any]	This option defines the CA "policy" to use and needs to be specified by the –policy command line switch. This is a section in the configuration file which decides which fields should be mandatory or match the CA certificate. The policy section consists of a set of variables corresponding to certificate DN fields. If the value is **match** then the field value must match the same field in the CA certificate. If the value is **supplied** then it must be present. If the value is **optional** then it may be present. Any fields not mentioned in the policy section are silently deleted.	
[…_ext]	Those sections specify the X.509 extensions and are referenced by the **x509_extensions** option within the configuration file (section **[req]** and **[CA_default]**). It can be overridden by the **-extensions** command line switch.	
	basicConstraints	This flag is used to determine whether the certificate can be used as a CA certificate.

1.4 Create the CA Certificate and Key

Now, that all initial configuration is done, we may create a self signed certificate, that will be used as our CA certificate. In other words, we will use this to sign other certificate requests.

Switch to the CA root directory. From this directory we can issue all **openssl commands** because our OpenSSL configuration file (*openssl.conf*) is located here.

Syntax to create the CA certificate and private key:

```
openssl req -new  -config <filename> -x509 -extensions <section> -keyout
<filename> -out <filename> -days <nn>
```

Option	Description
req	The *req* command primarily creates and processes certificate requests. It can instead create self signed certificates when the option **–x509** is specified.
-new	This option generates a new certificate request.
-config <filename>	This allows an alternative configuration file to be specified.
-x509	This option outputs a self signed certificate instead of a certificate request.
-extensions <section>	Specifies the section in the openssl configuration file (specified by **-config <filename>**) where the X.509 certificate extensions are defined.
-keyout <filename>	Filename of the CA's private key. Although it is protected with a pass phrase you should restrict access to it, so that only authorized users can read it.

Example:

```
C:\CA>openssl req -new -config openssl.conf -x509 -extensions ca_ext -keyout
private/ca.key -out certs/ca.crt -days 3640
Generating a 4096 bit RSA private key
.....++
.....++
writing new private key to 'private/ca.key'
Enter PEM pass phrase: - enter a strong pass phrase to use for this key
Verifying - Enter PEM pass phrase: - reenter the pass phrase again for verification
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [DE]: - we have kept the default value
Organization Name (company) [PHOENIX CONTACT Cyber Security AG]: - we have kept
the default value
Organizational Unit Name (department, division) [Support]: - we have kept the default
value
Common Name (hostname, IP, or your name) []:CA – we have entered the common name
for the CA certificate


C:\CA>
```

Two files are created:

– **certs/ca.crt**: This is the CA's certificate and can be publicly available and of course world readable.

– **private/ca.key**: This is the CA's private key. Although it is protected with a pass phrase you should restrict access to it, so that only authorized users may have access to it.

1.5 Create a Certificate Request for the mGuard

For obtaining a valid mGuard certificate you need to create a certificate request first and then sign it with the CA certificate (explained in chapter “Sign the mGuard’s Certificate Request with the CA” on page 12).

Syntax for creating a certificate request for the mGuard:

```
openssl req –new -config <filename> -keyout <filename> -out <filename> -days <nn>
```

Option	Description
req	The *req* command primarily creates and processes certificate requests.
-new	This option generates a new certificate request.
-config <filename>	This allows an alternative configuration file to be specified.
-keyout <filename>	Filename of the mGuard private key. Although it is protected with a pass phrase you should restrict access to it, so that only authorized users can read it.
-out <filename>	Filename of the mGuard certificate.
-days <nn>	The number of days the certificate should be valid.

Example:

```
C:\CA>openssl req -new -config openssl.conf -keyout private/mGuard.key -out
mGuard.csr -days 364
Generating a 4096 bit RSA private key
.....++
.....
+
writing new private key to 'private/mGuard.key'
Enter PEM pass phrase: - enter a strong pass phrase to use for this key
Verifying - Enter PEM pass phrase: - reenter the pass phrase again for verification
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [DE]: - we have kept the default value
Organization Name (company) [PHOENIX CONTACT Cyber Security AG]: - we have kept
the default value
Organizational Unit Name (department, division) [Support]: - we have kept the default
value
Common Name (hostname, IP, or your name) []:mGuard – enter the common name for
the mGuard certificate


C:\CA>
```

Two files are created:

– **mGuard.csr**: This is the certificate request which needs to be signed by the CA certificate.
– **private/mGuard.key**: This is the private key, which is not protected with a pass phrase.

1.6 Sign the mGuard's Certificate Request with the CA

The mGuard's certificate request needs to be signed by the CA to become a valid certificate.

Syntax for signing the mGuard's certificate request with the CA:

```
openssl ca –config <filename> -out <filename> -infiles <filename>
```

Option	Description
ca	The *ca* command is a minimal CA application. It can be used to sign certificate requests in a variety of forms and generate CRLs it also maintains a text database of issued certificates and their status.
-config <filename>	This allows an alternative configuration file to be specified.
-out <filename>	Filename of the signed mGuard certificate.
-infiles <filename>	Filename of the mGuard's certificate request. This must be the last option.

Example:

```
C:\CA>openssl ca -config openssl.conf -out certs/mGuard.crt -infiles mGuard.csr
Using configuration from openssl.conf
Enter pass phrase for C:/CA/private/ca.key: - enter the pass phrase of the CA's private key
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'DE'
organizationName      :ASN.1 12:'PHOENIX CONTACT Cyber Security AG'
organizationalUnitName:ASN.1 12:'Support'
commonName            :ASN.1 12:'mGuard'
Certificate is to be certified until Jul 7 09:02:23 2018 GMT (365 days)
Sign the certificate? [y/n]:y


1 out of 1 certificate requests certified, commit? [y/n]y
Write out database with 1 new entries
Data Base Updated


C:\CA>
```

After all this is done two new files are created:

- **certs/mGuard.crt**: This is the mGuards's certificate, which can be made available publicly.
- **newcerts/01.pem**: This is exactly the same certificate, but with the certificate's serial number (hex number) as filename. For subsequent requests the number is incremented. This file is not needed anymore and can be removed.

Now you can delete the mGuard's certificate request (*mGuard.csr*). It's no longer needed.

1.7 Creating the mGuard's PKCS#12 file (Machine Certificate)

This file combines the private and public key and is the mGuard's machine certificate which needs to be imported through the mGuard menu **Authentication >> Certificates >> Machine Certificates**. You'll be prompted to enter a password which protects the PKCS#12 export of the certificate against unauthorized usage.

Following is the syntax to create the mGuard machine certificate:

```
openssl pkcs12 -export -in <filename> -inkey <filename> -out <filename>
```

Option	Description
pkcs12	The *pkcs12* command allows PKCS#12 files to be created and parsed.
-export	This option specifies that a PKCS#12 file will be created rather than parsed.
-in <filename>	The filename to read the certificate from. The format of the file must be PEM. This is the mGuard's certificate you have created in the previous step.
-inkey <filename>	File to read private key from. This is the file which contains the private key of the mGuard's certificate.
-out <filename>	The filename to write certificates and private keys to. They are all written in PEM format.

Example:

```
C:\CA>openssl pkcs12 -export -in certs/mGuard.crt -inkey private/mGuard.key -out
certs/mGuard.p12
Enter pass phrase for private/mGuard.key: - enter the password of the mGuard's private
key
Enter Export Password: - enter a strong pass phrase to use for this export
Verifying - Enter Export Password: - reenter the pass phrase again for verification


C:\CA>
```

This command will create a file called **certs/mGuard.p12**, containing the mGuard certificate public and private key. The file is protected by the entered password.

1.8 Example: VPN connection between two mGuard devices

We assume that you already have setup the CA environment, configured the OpenSSL's configuration file (*openssl.conf*) and created the CA certificate and key. (As described in the previous chapters.)

Step 1: Create a certificate request for each mGuard

mGuard 1

```
openssl req -new -config openssl.conf -keyout private/mGuard1.key -out
mGuard1.csr -days 364
```

mGuard 2

```
openssl req -new -config openssl.conf -keyout private/mGuard2.key -out
mGuard2.csr -days 364
```

Step 2: Sign each certificate request with the CA

mGuard 1

```
openssl ca -config openssl.conf -out certs/mGuard1.crt -infiles mGuard1.csr
```

mGuard 2

```
openssl ca -config openssl.conf -out certs/mGuard2.crt -infiles mGuard2.csr
```

The two certificates **certs/mGuard1.crt** and **certs/mGuard2.crt** are created. **mGuard1.crt** needs to be imported on mGuard **2** as connection certificate through the menu **IPsec VPN >> Connections >> Authentication**. **mGuard2.crt** on **mGuard 1** correspondingly.

Step 3: Obtain the machine certificate for each mGuard

mGuard 1

```
openssl pkcs12 -export -in certs/mGuard1.crt -inkey private/mGuard1.key -out
certs/mGuard1.p12
```

mGuard 2

```
openssl pkcs12 -export -in certs/mGuard2.crt -inkey private/mGuard2.key -out
certs/mGuard2.p12
```

The two exports **certs/mGuard1.p12** and **certs/mGuard2.p12** are created.

mGuard1.p12 needs to be imported on mGuard **1** as machine certificate through the menu **Authentication >> Certificates >> Machine Certificates**. **mGuard2.p12** on mGuard **2** correspondingly.