Nützliche openssl Befehle

Schlüssel erzeugen

```
openssl genrsa -aes256 -out dvsdnet.devoteam.de.key.pem
```

Bei RSA Schlüsseln wird automatisch ein Passwort abgefragt. Das kann man in einem zweiten Schritt wieder entfernen:

```
openssl rsa -in ${filename}.key -out ${filename}.key
```

ECDSA (elliptic curve)

Diese Schlüssel sind kleiner, bein Erzeugen wird kein Passwort verlangt

```
openssl ecparam -name prime256v1 -genkey -noout -out key.pem
```

ec Key **mit** Passwort erzeugen

```
openssl ecparam -name prime256v1 -genkey | openssl ec -aes256 -out key.pem
```

Certificate Signing Request

Standard

Der csr wird entweder selber signiert oder an eine externe CA gegeben. -days regelt die Gültigkeit des Zertifikats und überschreibt die default Werte der ssh Konfiguration.

Einfacher CSR

```
openssl req -nodes -new -newkey rsa:2048 -sha256 -out csr.pem
```

```
''openssl req -new -key private/openvpn_client_odysseus.key -out
certs/openvpn_client_odysseus.req.pem -days 730 ''
```

Achtung bei openvpn: CN muß mit "locutusvpn" beginnen, da die im Server geprüft wird (Server Parameter –verify-x509-name locutusvpn name-prefix)

Mit separater cnf Datei

```
[ req ]
default_bits           = 2048
default_keyfile        = privkey.key
distinguished_name     = req_distinguished_name
attributes             = req_attributes
req_extensions         = v3_ca
```

```
dirstring_type = nobmp

[ req_distinguished_name ]
C                      = DE
ST                     = Baden-Wuerttemberg
L                      = Stuttgart
O                      = Mercedes-Benz Group AG
CN                     = Common Name
emailAddress           = test@email.address

[ req_attributes ]
challengePassword              = A challenge password
challengePassword_min          = 4
challengePassword_max          = 20

[ v3_ca ]

subjectKeyIdentifier = hash
basicConstraints = CA:false
keyUsage = nonRepudiation, digitalSignature, keyEncipherment
```

Signieren über eigene CA

```
openssl x509 -req -in certs/openvpn_client_odysseus.req.pem -CA
certs/RootCA.cert.pem -CAkey private/RootCA.key.pem -out
certs/openvpn_client_odysseus.cert.pem -days 720
```

Achtung: Beim Signieren über die eigene CA werden die Zertifikat-Nr hochgezählt (serial in srl Datei). Die srl Datei muss auch für die ServerCA separat existieren.

```
mv newcerts/01.pem certs/ cd certs ln -s 01.pem `openssl x509 -hash -noout -
in 01.pem`.0
```

Checks

Inhalt kontrollieren

Auf der Shell im Klartext lesen: bei CERTs mit "x509", bei CSRs mit "req"

```
openssl x509 -text -noout -in ca.crt
openssl req -text -noout -verify -in csr.pem
```

Check: Passen key und crt zusammen?

Replace <public.crt> with the filename of the public certificate.

```
openssl x509 -noout -modulus -in <public.crt> | openssl md5> /tmp/crt.pub
```

```
openssl rsa -noout -modulus -in <private.key> | openssl md5> /tmp/key.pub
```

Danach ein diff der beiden Dateien:

```
diff /tmp/crt.pub /tmp/key.pub
```

If nothing is printed to the console, they were found to be a pair. Any differences are printed to the console in detail.

Formate und Konvertierung

x509 nach pem

```
openssl x509 -in certificatename.cer -outform PEM -out certificatename.pem
```

Von pfx nach pem

```
openssl pkcs12 -in [pkcs-12-certificate-and-key-file] -out [pem-certificate-and-key-file]
```

ca-certs extrahieren

```
openssl pkcs12 -in ${filename}.pfx -nodes -nokeys -cacerts -out ${filename}-ca.crt
```

key extrahieren

```
openssl pkcs12 -in ${filename}.pfx -nocerts -out ${filename}.key
```

cert extrahieren

```
openssl pkcs12 -in ${filename}.pfx -clcerts -nokeys -out ${filename}.crt
```

ca-certs und cert zusammen kleben

```
cat ${filename}.crt ${filename}-ca.crt> ${filename}-full.crt
```

Passphäse aus key löschen

```
openssl rsa -in ${filename}.key -out ${filename}.key
```

CA und Sub CA erstellen

Mit der ServerCA lassen sich jetzt Zertifikate für Server erstellen. Diese sollten - weil sie ihren Private Key ohne menschliches Zutun benutzen müssen - keine Passphrase haben. Dies geschieht durch Weglassen des Verschlüsselungs-Algoritmus beim Erstellen des Private Key. Ansonsten sind das auch

nur normale Zertifikate. Allerdings werden sie nicht von der RootCA, sondern von der ServerCA signiert.

Apache mit mod_ssl

xxx

Apache mit mTLS

https://vignesh-thirunavukkarasu.medium.com/mtls-with-apache-http-server-fbfd702106ca

EASYrsa (openvpn)

xxx

SSL Test mit openssl s_client

xxx