# OPENSSL

## privaten Schlüssel erzeugen (RSA)

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

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

Passphrase aus key löschen (RSA)

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

## Schlüssel mit 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 (CSR)

-days regelt die Gültigkeit des Zertifikats und überschreibt die default Werte der ssh Konfiguration.

Der csr wird entweder selber signiert oder an eine externe CA gegeben.

Einfacher CSR

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

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

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

### 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
```

# Erstellen eines csr mit einer cnf Datei

```
openssl req -new -key radius_rsa.key -out radius.xdc.dev.gspp-
eu.corpinter.net_rsa.csr -config openssl.cnf
```

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
```

# Check: Passen ein key und ein signiertes cert 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.

## Check: Inhalte 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
```

## Revocation list

xx

## Test einer ssl Verbindung mit openssl s_client

Das Standard-Tool für die Analyse von SSL-Funktionen ist das Kommandozeilenprogramm von OpenSSL.

```
openssl s_client -connect imap.1und1.de:993
```

Kommt als Ergebnis eine Ciphersuite heraus, die mit DH oder ECDH beginnt, haben sich die beiden Kommunikationspartner auf Forward Secrecy geeinigt. Ob ein Server überhaupt Forward Secrecy beherrscht, verrät:

```
openssl s_client -cipher 'ECDH:DH' -connect login.live.com:443
```

Dieses Beispiel zeigt, dass auch Microsofts Server durchaus DH-Verfahren im Repertoire haben. Ob ein Server Diffie-Hellman erzwingt, auch wenn der Client RSA bevorzugt, verrät die cipher-Spezifikation 'RSA:ECDH:DH'. OpenSSL unterstützt auch Verbindungen, bei denen man die Verschlüsselung über den starttls anfordern muss, wie es etwa im Mail-Versandprotokoll SMTP üblich ist:

```
openssl s_client -starttls smtp -connect smtp.gmx.net:587
```

Dokumentation zu openssl siehe auch http://www.openssl.org/docs/