

# Easy-RSA3

Quelle: <https://github.com/OpenVPN/easy-rsa>

## CA einrichten

initialisieren

```
./easyrsa init-pki
```

DH erzeugen

```
./easyrsa gen-dh
```

pki bauen (mit PW)

```
./easyrsa build-ca
```

## Signing Request

Signing Request (CSR) für Server oder Client, mit oder ohne Passwort (nopass = Key ohne Passwort)

```
./easyrsa gen-req EntityName nopass
```

## Signieren (client oder server)

```
./easyrsa sign-req client EntityName
```

```
./easyrsa sign-req server EntityName
```

## Zertifikats Inhalt anzeigen

```
./easyrsa show-req EntityName
```

```
./easyrsa show-cert EntityName
```

Alternativ direkt mit openssl

```
# Zertifikat komplett anzeigen  
openssl x509 -text -in <zertifikatsname.crt>
```

```
# Komplette Anzeige ohne Zertifikatstext anzeigen  
openssl x509 -noout -text -in <zertifikatsname.crt>
```

```
# den Herausgeber des Zertifikats anzeigen
openssl x509 -noout -issuer -in <zertifikatsname.crt>

# Für wen wurde das Zertifikat ausgestellt?
openssl x509 -noout -subject -in <zertifikatsname.crt>

# Für welchen Zeitraum ist das Zertifikat gültig?
openssl x509 -noout -dates -in <zertifikatsname.crt>

# das obige kombiniert anzeigen
openssl x509 -noout -issuer -subject -dates -in <zertifikatsname.crt>

# den hash anzeigen
openssl x509 -noout -hash -in <zertifikatsname.crt>

# den MD5-Fingerprint anzeigen
openssl x509 -noout -fingerprint -in <zertifikatsname.crt>
```

Key Passwörter ändern

```
./easyrsa set-rsa-pass EntityName
```

```
./easyrsa set-ec-pass EntityName
```

Mit "nopass" wird ein Passwort entfernt

## CSR

```
openssl req -in www2.netzwissen.de.csr -text -noout
```

## Zertifikat

```
openssl x509 -in certificate.crt -text -noout
```

Achtung bei OpenVPN: der Server prüft den Präfix-String im CN, z.B: bei Devoteam:

```
CN=openvpn_dvsdnet_thomas.rother@devoteam.com
```

## Zertifikat zurückziehen

```
./easyrsa revoke EntityName
```

CRL erzeugen

```
./easyrsa gen-crl
```

# Spezialfälle bei OpenVPN

Validierung über common name (alte Methode): Der Common Name für das Server Zertifikat muss den Präfix aus der Server Config enthalten:

```
-verify-x509-name openvpn name-prefix
```

Mit easysrsa3:

```
locutus:~/easy-rsa/easysrsa3
```

```
./easysrsa sign-req server openvpn.locutus.netzwissen.loc
```

## Neue Methode nach RFC3280: Verifikation nach Zertifikatstyp:

```
remote-cert-tls client
```

```
./easysrsa sign-req client franklin2.netzwissen.loc
```

Im Client wird im Gegenzug geprüft, ob das vom OpenVPN Server präsentierte Zertifikat den "server" Typ hat

# PKITool Befehle

```
pktool -initca → Build root certificate
```

```
pktool -initca -pass → Build root certificate with password-protected key
```

```
pktool -server server1 → Build "server1" certificate/key
```

```
pktool client1 → Build "client1" certificate/key
```

```
pktool -pass client2 → Build password-protected "client2" certificate/key
```

```
pktool -pkcs12 client3 → Build "client3" certificate/key in PKCS#12 format
```

```
pktool -csr client4 → Build "client4" CSR to be signed by another CA
```

```
pktool -sign client4 → Sign "client4" CSR
```

```
pktool -inter interca → Build an intermediate key-signing certificate/key
```

Also see ./inherit-inter script.

```
pktool -pkcs11 /usr/lib/pkcs11/lib1 0 010203 "client5 id" client5 → Build "client5" certificate/key in PKCS#11 token
```

Typical usage for initial PKI setup. Build myserver, client1, and client2 cert/keys. Protect client2 key with a password. Build DH parms. Generated files in ./keys :

```
[edit vars with your site-specific info] source ./vars ./clean-all ./build-dh  
→ takes a long time, consider backgrounding ./pktool -initca ./pktool  
-server myserver ./pktool client1 ./pktool -pass client2
```

Typical usage for adding client cert to existing PKI:

```
source ./vars ./pktool client-new
```

From:

<https://wiki.netzwissen.de/> - **netzwissen.de Wiki**

Permanent link:

<https://wiki.netzwissen.de/doku.php?id=easyrsa&rev=1561982577>

Last update: **05/03/2024 - 10:52**

