TLS Zertifikate und deren Verwendung unter Ubuntu und Gentoo

Zuerst besorgt man sich mal ein kostenloses Zertifikat bei Startssl: https://www.startssl.com/

Pfade

Hier kann man die Pfade hinterlegen:

```
## Gentoo
nano /etc/apache2/vhosts.d/00_default_ssl_vhost.conf

## Ubuntu
nano /etc/apache2/sites-available/default-ssl
```

CSR erstellen

```
openssl req -new -key ssl.key -out cert.csr
```

Apache Zertifikate hinterlegen

In Apache benötigt man bei Startssl 4 Zertifikate.

StartCom Certification Authority (am längsten gültig ca. 2040)

```
## Certificate Authority (CA):
    # Set the CA certificate verification path where to find CA certificates
    # for client authentication or alternatively one huge file containing
all
    # of them (file must be PEM encoded).
    # Note: Inside SSLCACertificatePath you need hash symlinks to point to
the
    # certificate files. Use the provided Makefile to update the hash
symlinks
    # after changes.
    SSLCACertificateFile /etc/ssl/apache2/ca.pem
```

Heist auch oft ca_root_startTLS.pem

StartCom Class 1 Primary Intermediate Server CA

```
## Server Certificate Chain:
    # Point SSLCertificateChainFile at a file containing the concatenation
of
    # PEM encoded CA certificates which form the certificate chain for the
    # server certificate. Alternatively the referenced file can be the same
as
    # SSLCertificateFile when the CA certificates are directly appended to
the
    # server certificate for convinience.
    SSLCertificateChainFile /etc/ssl/apache2/cca.pem
```

Heist auch oft startTLSCAcert.pem

Private Key entschlüsselt

```
## Server Private Key:
    # If the key is not combined with the certificate, use this directive to
    # point at the key file. Keep in mind that if you've both a RSA and a
DSA
    # private key you can configure both in parallel (to also allow the use
of
    # DSA ciphers, etc.)
    SSLCertificateKeyFile /etc/ssl/apache2/server.key
```

Zertifikat

```
## Server Certificate:
    # Point SSLCertificateFile at a PEM encoded certificate. If the
certificate
    # is encrypted, then you will be prompted for a pass phrase. Note that a
    # kill -HUP will prompt again. Keep in mind that if you have both an RSA
    # and a DSA certificate you can configure both in parallel (to also
allow
    # the use of DSA ciphers, etc.)
    SSLCertificateFile /etc/ssl/apache2/server.crt
```

Um nach Einrichtung alles zu überprüfen bedient man sich openssl am localhost:

```
openssl s_client -connect localhost:443 -state  -ssl3
```

Das Ergebnis sollte dann ca. so aussehen:

```
CONNECTED(00000003)
SSL_connect:before/connect initialization
SSL_connect:SSLv3 write client hello A
```

```
SSL3 alert read:fatal:handshake failure
SSL_connect:failed in SSLv3 read server hello A
140679657948816:error:14094410:SSL routines:ssl3_read_bytes:sslv3 alert
handshake failure:s3_pkt.c:1472:SSL alert number 40
140679657948816:error:1409E0E5:SSL routines:ssl3_write_bytes:ssl handshake
failure:s3_pkt.c:656:
---
no peer certificate available
---
No client certificate CA names sent
---
SSL handshake has read 7 bytes and written 0 bytes
---
New, (NONE), Cipher is (NONE)
Secure Renegotiation IS NOT supported
Compression: NONE
Expansion: NONE
No ALPN negotiated
SSL-Session:
    Protocol  : SSLv3
    Cipher    : 0000
    Session-ID:
    Session-ID-ctx:
    Master-Key:
    Key-Arg   : None
    PSK identity: None
    PSK identity hint: None
    SRP username: None
    Start Time: 1453931519
    Timeout   : 7200 (sec)
    Verify return code: 0 (ok)
```