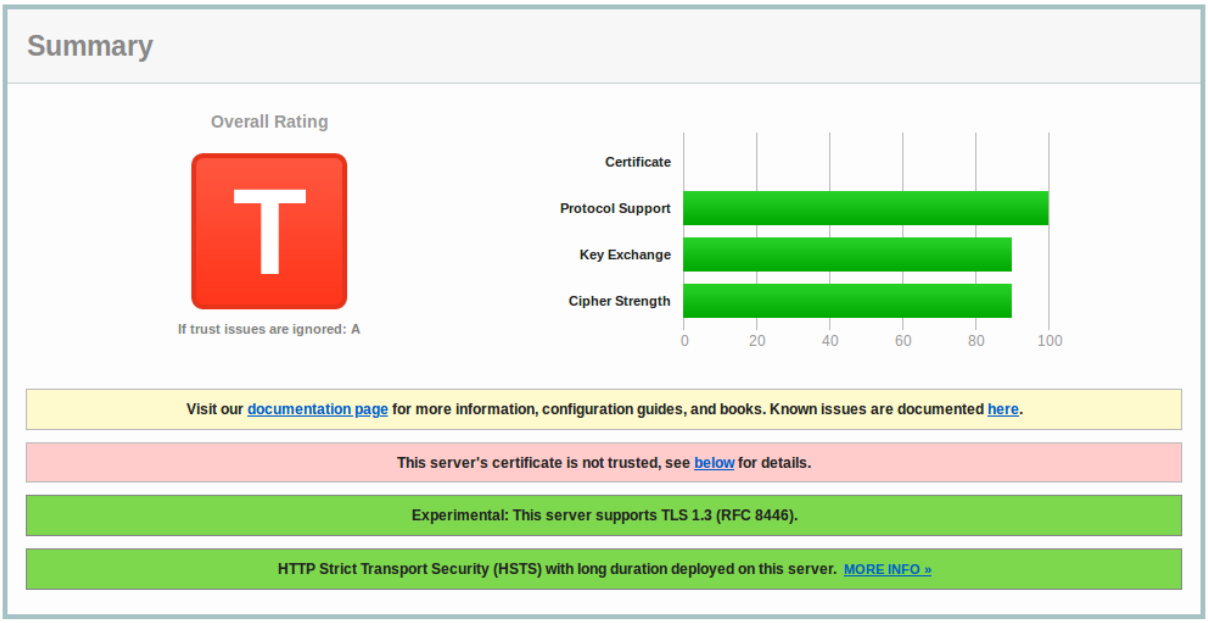
Raspberry Pi – SSL/TLS einrichten (OpenSSL, PKI)

diese Anleitung…

- ist nur Teil der gesamten Anleitung, für eine ordnungsgemäße Verschlüsselung sind alle Informationen auf http://www.cenz.at/raspberry#verschluesselung zu berücksichtigen
- verwendet Beispiel-Eingaben, diese sind durch eigene Werte bzw. Angaben zu ersetzen
- beinhaltet zeilenübergreifende Codezeilen, welche mit einem „\“ enden; so ein Codeblock ist als Ganze zu verwenden
- ist ohne genau Erläuterung der jeweiligen Befehle, diese sind selbstständig zu recherchieren bzw. in folgender Quelle nachzulesen: http://pki-tutorial.readthedocs.org/en/latest/advanced/index.html



Ergebnis einer ausführlichen Analyse der Konfiguration vom SSL-Server nach dieser Anleitung

1. Superuser

```
sudo su
```

2. Grundgerüst

2.1 Grund-Verzeichnisstruktur anlegen

```
cd /etc/ssl/
mkdir -p PKI/CENZ/etc
```

2.2 Konfigurationsdateien anlegen

```
cd PKI/CENZ/etc
wget http://www.cenz.at/various/openssl_root-ca.conf.txt
wget http://www.cenz.at/various/openssl_tls-ca.conf.txt
wget http://www.cenz.at/various/openssl_server.conf.txt

mv openssl_root-ca.conf.txt root-ca.conf
mv openssl_tls-ca.conf.txt tls-ca.conf
mv openssl_server.conf.txt server.conf
```

Die Eingaben in den folgenden Dateien müssen ident sein!

```
nano root-ca.conf
```

```
     [ default ]
     ca                      = root-ca                       # CA name
     dir                     = .                             # Top dir
     base_url                = http://pki.example.at         # CA base URL
     aia_url                 = $base_url/$ca.cer             # CA certificate URL
     crl_url                 = $base_url/$ca.crl             # CRL distribution point
     name_opt                = multiline,-esc_msb,utf8       # Display UTF-8 characters

     [ ca_dn ]
     countryName             = "AT"
     organizationName        = "Example"
     organizationalUnitName  = "Example Certificate Authority"
     commonName              = "Example Root CA"
```

```
nano tls-ca.conf
```

```
     [ default ]
     ca                      = tls-ca                        # CA name
     dir                     = .                             # Top dir
     base_url                = http://pki.example.at         # CA base URL
     aia_url                 = $base_url/$ca.cer             # CA certificate URL
     crl_url                 = $base_url/$ca.crl             # CRL distribution point
     name_opt                = multiline,-esc_msb,utf8       # Display UTF-8 characters

      [ ca_dn ]
     countryName             = "AT"
     organizationName        = "Example"
     organizationalUnitName  = "Example Certificate Authority"
     commonName              = "Example TLS CA"
```

```
cd /etc/ssl/PKI/CENZ
```

3. Root-CA erstellen

http://pki-tutorial.readthedocs.io/en/latest/advanced/index.html#create-root-ca

3.1 Create directories

```
mkdir -p ca/root-ca/private ca/root-ca/db crl certs
chmod 700 ca/root-ca/private
```

3.2 Create database

```
cp /dev/null ca/root-ca/db/root-ca.db
cp /dev/null ca/root-ca/db/root-ca.db.attr
echo 01 > ca/root-ca/db/root-ca.crt.srl
echo 01 > ca/root-ca/db/root-ca.crl.srl
```

3.3 Create CA request

```
openssl req -new \
    -config etc/root-ca.conf \
    -out ca/root-ca.csr \
    -keyout ca/root-ca/private/root-ca.key
```

`Enter PEM pass phrase:`	Passwort min. 4 Stellen, eher 20 bis 30-stellig: Zahlen, Groß- und Kleinbuchstaben, Sonderzeichen „+“ „-„ „+“ „?“ verwenden **(ROOT-CA!)**

3.4 Create CA certificate

```
openssl ca -selfsign \
    -config etc/root-ca.conf \
    -in ca/root-ca.csr \
    -out ca/root-ca.crt \
    -extensions root_ca_ext \
    -enddate 20301231235959Z
```

`Enter pass for ./ca/root-ca/private/root-ca.key:`	[Passwort: ROOT-CA]
`Sign the certificate? [y/n]:`	y [falls Angaben korrekt]
`1 out of 1 certificate requests certified, commit? [y/n]`	y

3.5 Create initial CRL

```
openssl ca -gencrl \
    -config etc/root-ca.conf \
    -out crl/root-ca.crl
```

`Enter passs for ./ca/root-ca/private/root-ca.key:`	[Passwort: ROOT-CA]

4. TLS-CA erstellen

http://pki-tutorial.readthedocs.io/en/latest/advanced/index.html#create-tls-ca

4.1 Create directories

```
mkdir -p ca/tls-ca/private ca/tls-ca/db crl certs
chmod 700 ca/tls-ca/private
```

4.2 Create database

```
cp /dev/null ca/tls-ca/db/tls-ca.db
cp /dev/null ca/tls-ca/db/tls-ca.db.attr
echo 01 > ca/tls-ca/db/tls-ca.crt.srl
echo 01 > ca/tls-ca/db/tls-ca.crl.srl
```

4.3 Create CA request

```
openssl req -new \
    -config etc/tls-ca.conf \
    -out ca/tls-ca.csr \
    -keyout ca/tls-ca/private/tls-ca.key
```

`Enter PEM pass phrase:`	Passwort min. 4 Stellen, eher 20 bis 30-stellig: Zahlen, Groß- und Kleinbuchstaben, Sonderzeichen „+“ „-„ „+“ „?“ verwenden **(TLS-CA!)**

4.4 Create CA certificate

```
openssl ca \
    -config etc/root-ca.conf \
    -in ca/tls-ca.csr \
    -out ca/tls-ca.crt \
    -extensions signing_ca_ext
```

`Enter pass phrase for ./ca/root-ca/private/root-ca.key:`	[Passwort: ROOT-CA]
`Sign the certificate? [y/n]:`	y [falls Angaben korrekt]
`1 out of 1 certificate requests certified, commit? [y/n]`	y

4.5 Create initial CRL

```
openssl ca -gencrl \
    -config etc/tls-ca.conf \
    -out crl/tls-ca.crl
```

`Enter pass phrase for ./ca/tls-ca/private/tls-ca.key:`	[Passwort: TLS-CA]

4.6 Create PEM bundle

```
cat ca/tls-ca.crt ca/root-ca.crt > ca/tls-ca-chain.pem
```

5. TLS CA betreiben

http://pki-tutorial.readthedocs.io/en/latest/advanced/index.html#operate-tls-ca

5.1 Create TLS server request

Anmerkung: falls es nur eine Subdomain ist, Beispiel: `SAN=DNS:cloud.cenz.at \`

```
SAN=DNS:seafile.cenz.at,DNS:baikal.cenz.at,DNS:ttrss.cenz.at \
openssl req -new \
    -config etc/server.conf \
    -out certs/server.cenz.at.csr \
    -keyout certs/server.cenz.at.key
```

```
1. Country Name (2 letters) (eg, US)       []:AT
2. State or Province Name   (eg, region)   []:Austria
3. Locality Name            (eg, city)     []:Vienna
4. Organization Name        (eg, company)  []:cEnz
5. Organizational Unit Name (eg, section)  []:cEnz Certificate Authority
6. Common Name              (eg, FQDN)     []:cEnz Identity CA
```

5.2 Create TLS server certificate¶

```
openssl ca \
    -config etc/tls-ca.conf \
    -in certs/server.cenz.at.csr \
    -out certs/server.cenz.at.crt \
    -extensions server_ext
```

```
Enter pass phrase for ./ca/tls-ca/private/tls-ca.key:
```
[Passwort: TLS-CA]

```
Sign the certificate? [y/n]:
```
y [falls Angaben korrekt]

```
1 out of 1 certificate requests certified, commit? [y/n]
```
y

5.3 Create PKCS#12 bundle

```
openssl pkcs12 -export \
    -name "cEnz.at (Server)" \
    -caname "cEnz TLS CA" \
    -caname "cEnz Root CA" \
    -inkey certs/server.cenz.at.key \
    -in certs/server.cenz.at.crt \
    -certfile ca/tls-ca-chain.pem \
    -out certs/server.cenz.at.p12
```

```
Enter Export Password:
```
Passwort min. 4 Stellen, eher 20 bis 30-stellig: Zahlen, Groß- und Kleinbuchstaben, Sonderzeichen „+“ „-„ „+“ „?“ verwenden **(EXPORT!)**

6. erweiterte Konfiguration

6.1 kompaktes Zertifikat

Das „Bündel“ beinhaltet root- und tls-crts, was nur einen einmaligen Transfer erfordert.

Behebt im SSL-Test: „This server's certificate chain is incomplete.“
Erzeugt im SSL-Test: Chain issues Contains anchor (IST OK!)

```
cd /etc/ssl/PKI/CENZ/

cat certs/server.cenz.at.crt ca/tls-ca.crt \
    ca/root-ca.crt > certs/ssl-bundle.crt
```

6.2 Diffie-Hellman-Schlüsselaustausch

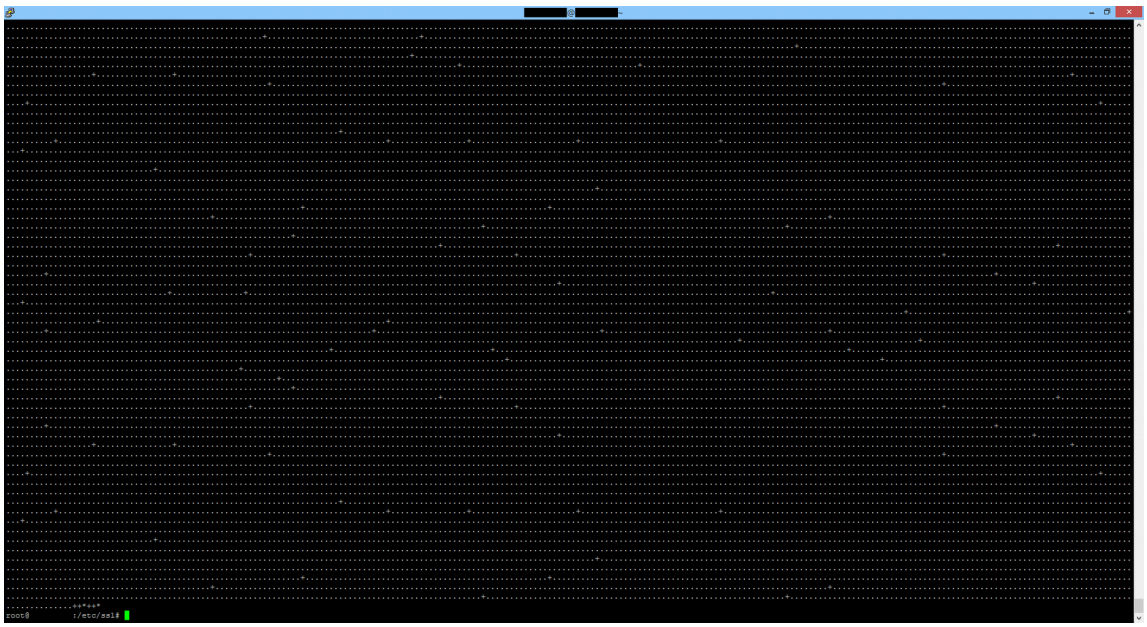
Verbessert den Schlüsselaustauschparameter.
VORSICHT: das dauert etwa 10 Stunden! (bei Raspberry Pi 1 Modell B ca. 13 Stunden)

Behebt im SSL-Test: „This server supports weak Diffie-Hellman (DH) key exchange parameters.“

```
cd /etc/ssl/

openssl dhparam -out private/dh4096.key.pem 4096
```

Das sollte danach so aussehen:



Anmerkung: falls es nicht seitenweise Punkte gibt, und der Vorgang gerade mal 20 Minuten gedauert hat, sollte man die Datei löschen und den Befehl neu ausführen!

6.3 Root-CA exportieren

Das Root-Zertifikat (Punkt 3) kann man manuell am Gerät (PC, Smartphone, usw.) installieren. Danach erkennt das System (Windows, Android, usw.) und die Programme das selbst erstellte Zertifikat an.
Um das Root-Zertifikat ohne große Umstände auf den PC zu bekommen, kopieren wir es in das Webseiten-Verzeichnis und laden es über den Browser herunter.

```
cp /etc/ssl/PKI/CENZ/ca/root-ca.crt
/var/www/html/baikal/html/zertifikat
```

http://[IP des Pis]/zertifikat → auf Festplatte speichern

```
rm /var/www/html/baikal/html/zertifikat
```

Danach löschen wir es wieder aus dem Webseiten-Verzeichnis und benennen es auf der Festplatte mit der Dateiendung .crt, Beispiel: „cEnz Root CA.crt“

6.4 Beispiel einer korrekten NGINX-Konfigurationsdatei (Webseite mit PHP)

weitere Konfigurationsdateien auf http://www.cenz.at/raspberry
NGINX Dokumentation: http://nginx.org/en/docs/

```
# BAIKAL
server {
       listen 443;
       server_name baikal.example.at;

       root /var/www/html/baikal/html;
       index index.php;

       ssl on;
       ssl_certificate /etc/ssl/PKI/example/certs/ssl-bundle.crt;
       ssl_certificate_key /etc/ssl/PKI/example/certs/server.cenz.at.key;

       # Session-Wiederaufnahme aktivieren um https Leistung zu verbessern
       ssl_session_cache   shared:SSL:10m;
       ssl_session_timeout 10m;

       # TLS-Versionen: 1.2 und 1.3
       ssl_protocols TLSv1.2 TLSv1.3;
       # Diffie-Hellman (DH) http://de.wikipedia.org/wiki/Diffie-Hellman-Schl
%C3%BCsselaustausch
       ssl_dhparam /etc/ssl/private/dh4096.key.pem;
       # Schutz gegen Ausspaehung
       ssl_ciphers ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-
SHA384:DHE-RSA-AES256-SHA256:ECDHE-RSA-AES256-SHA:DHE-RSA-AES256-SHA:!aNULL:!eNULL:!EXPORT:!
LOW:!MEDIUM:!DES:!3DES:!RC4:!SEED:!CAMELLIA:!MD5:!PSK:!DSS;
       ssl_prefer_server_ciphers on;
       # Schutz gegen man-in-the-middle-attacks,
https://de.wikipedia.org/wiki/Hypertext_Transfer_Protocol_Secure#HSTS
       add_header Strict-Transport-Security "max-age=31536000; includeSubdomains";
       # Nginx-Version nicht anzeigen, potenzielle Angreifer müssen sich ein wenig mehr Mühe
machen als nur den Header abzufragen
       server_tokens off;


       rewrite ^/.well-known/caldav /dav.php redirect;
       rewrite ^/.well-known/carddav /dav.php redirect;

       location ~ ^(.+\.php)(.*)$ {
          try_files $fastcgi_script_name =404;
          include        /etc/nginx/fastcgi_params;
          fastcgi_split_path_info  ^(.+\.php)(.*)$;
          fastcgi_pass   unix:/run/php/php7.3-fpm.sock;
          fastcgi_param  SCRIPT_FILENAME  $document_root$fastcgi_script_name;
          fastcgi_param  PATH_INFO        $fastcgi_path_info;
          access_log   /var/log/nginx/baikal.access.log;
          error_log    /var/log/nginx/baikal.error.log;
       }

       # deny access to file and directories
       location ~ /(\.ht|Core|Specific) {
          deny all;
          return 404;
       }

}

# TINY TINY RSS
server {
```

```
        listen 443;
        server_name ttrss.example.at;

        root /var/www/html/ttrss;
        index index.php;

        ssl on;
        ssl_certificate /etc/ssl/PKI/example/certs/ssl-bundle.crt;
        ssl_certificate_key /etc/ssl/PKI/example/certs/server.cenz.at.key;

        # Session-Wiederaufnahme aktivieren um https Leistung zu verbessern
        ssl_session_cache   shared:SSL:10m;
        ssl_session_timeout 10m;

        # TLS-Versionen: 1.2 und 1.3
        ssl_protocols TLSv1.2 TLSv1.3;
        # Diffie-Hellman (DH) http://de.wikipedia.org/wiki/Diffie-Hellman-Schl
%C3%BCsselaustausch
        ssl_dhparam /etc/ssl/private/dh4096.key.pem;
        # Schutz gegen Ausspaehung
        ssl_ciphers ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-
SHA384:DHE-RSA-AES256-SHA256:ECDHE-RSA-AES256-SHA:DHE-RSA-AES256-SHA:!aNULL:!eNULL:!EXPORT:!
LOW:!MEDIUM:!DES:!3DES:!RC4:!SEED:!CAMELLIA:!MD5:!PSK:!DSS;
        ssl_prefer_server_ciphers on;
        # Schutz gegen man-in-the-middle-attacks,
https://de.wikipedia.org/wiki/Hypertext_Transfer_Protocol_Secure#HSTS
        add_header Strict-Transport-Security "max-age=31536000; includeSubdomains";
        # Nginx-Version nicht anzeigen, potenzielle Angreifer müssen sich ein wenig mehr Mühe
machen als nur den Header abzufragen
        server_tokens off;


        location / {
          try_files $uri $uri/ =404;
          access_log   /var/log/nginx/ttrss.access.log;
          error_log    /var/log/nginx/ttrss.error.log;
        }

        location ~ \.php$ {
          include snippets/fastcgi-php.conf;
          fastcgi_pass unix:/run/php/php7.3-fpm.sock;
        }

        location ~ /(\.ht) {
          deny all;
          return 404;
        }

}

# SEAFILE
server {
        listen 443;
        server_name seafile.example.at;

        proxy_set_header X-Forwarded-For $remote_addr;

        ssl on;
        ssl_certificate /etc/ssl/PKI/example/certs/ssl-bundle.crt;
        ssl_certificate_key /etc/ssl/PKI/example/certs/server.cenz.at.key;

        # Session-Wiederaufnahme aktivieren um https Leistung zu verbessern
        ssl_session_cache   shared:SSL:10m;
        ssl_session_timeout 10m;

        # TLS-Versionen: 1.2 und 1.3
```

```
        ssl_protocols TLSv1.2 TLSv1.3;
        # Diffie-Hellman (DH) http://de.wikipedia.org/wiki/Diffie-Hellman-Schl
%C3%BCsselaustausch
        ssl_dhparam /etc/ssl/private/dh4096.key.pem;
        # Schutz gegen Ausspaehung
        ssl_ciphers ECDHE-RSA-AES256-GCM-SHA384:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-
SHA384:DHE-RSA-AES256-SHA256:ECDHE-RSA-AES256-SHA:DHE-RSA-AES256-SHA:!aNULL:!eNULL:!EXPORT:!
LOW:!MEDIUM:!DES:!3DES:!RC4:!SEED:!CAMELLIA:!MD5:!PSK:!DSS;
        ssl_prefer_server_ciphers on;
        # Schutz gegen man-in-the-middle-attacks,
https://de.wikipedia.org/wiki/Hypertext_Transfer_Protocol_Secure#HSTS
        add_header Strict-Transport-Security "max-age=31536000; includeSubdomains";
        # Nginx-Version nicht anzeigen, potenzielle Angreifer müssen sich ein wenig mehr Mühe
machen als nur den Header abzufragen
        server_tokens off;


        # SEAFILE
        location / {
            proxy_pass         http://127.0.0.1:8000;
            proxy_set_header   Host $host;
            proxy_set_header   X-Real-IP $remote_addr;
            proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header   X-Forwarded-Host $server_name;
            proxy_set_header   X-Forwarded-Proto https;
            proxy_read_timeout  1200s;
            client_max_body_size 0;

            access_log      /var/log/nginx/seahub.access.log;
            error_log       /var/log/nginx/seahub.error.log;
        }
        location /seafhttp {
            rewrite ^/seafhttp(.*)$ $1 break;
            proxy_pass http://127.0.0.1:8082;
            client_max_body_size 0;
            proxy_set_header   X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_connect_timeout  36000s;
            proxy_read_timeout  36000s;
            proxy_send_timeout  36000s;
            send_timeout  36000s;

            access_log      /var/log/nginx/seahub.access.log;
            error_log       /var/log/nginx/seahub.error.log;
        }
        location /media {
            root /home/seafile/seafile-server-latest/seahub;
        }
}
```

7. FERTIG! - Ergebnis testen

https://www.ssllabs.com/ssltest/