Zertifikat für admin.ekey Zugang

Dieser Leitfaden beschreibt, wie mit OpenSSL eine eigene Certificate Authority (CA) betrieben und daraus **Client-Zertifikate für Benutzer** erstellt werden, um z. B. eine Webseite per mTLS (Client-Zertifikatsauthentifizierung) zugänglich zu machen.

Übersicht

1. Einmalig: **CA-Schlüssel** und **CA-Zertifikat** erstellen
2. Für **jeden neuen Nutzer**:
   - privaten Schlüssel erzeugen
   - CSR (Certificate Signing Request) erzeugen
   - CSR mit der CA signieren → Benutzerzertifikat
   - Benutzerzertifikat + Key + CA-Zertifikat als `.pfx` exportieren
3. Nutzer importiert `.pfx` im Betriebssystem / Browser
4. Webserver ist so konfiguriert, dass er:
   - Client-Zertifikate verlangt
   - der eigenen CA vertraut
   - (optional) bestimmte Nutzer anhand des Zertifikats freischaltet

---

Voraussetzungen

- OpenSSL ist installiert (`openssl` auf der Kommandozeile verfügbar).
- Ein Verzeichnis für die CA, z. B.:

```
cd /CertificateAuthCA
```

- Basiskonzept:
  - CA-Dateien: `fb4ca.key`, `fb4ca.crt`, `fb4ca.srl`
  - Benutzer-Dateien: `<user>.key`, `<user>.csr`, `<user>.crt`, `<user>.pfx`

---

1. Einmalig: eigene CA erstellen

> Um einen neuen Nutzer zu Berechtigen wird dieser Schritt NICHT benötigt

Diesen Schritt führst du **nur einmal** aus.
Er erzeugt den privaten CA-Schlüssel und ein selbstsigniertes Root-Zertifikat.

```
# 1.1 CA-Privatschlüssel erzeugen
openssl genrsa -des3 -out fb4ca.key 4096
```

- Es wird eine **Passphrase für den CA-Key** abgefragt (merken und sicher aufbewahren).

```
# 1.2 Self-signed Root-CA-Zertifikat erzeugen
openssl req -new -x509 -days 3650 -key fb4ca.key -out fb4ca.crt
```

- Wichtige Felder:
  - `Common Name (CN)` → sinnvoller Name der CA, z. B. `Example Internal Root CA`
  - Rest nach Bedarf (`O`, `OU`, `C`, …)

Die Dateien:

- `fb4ca.key` → **geheimer** privater Schlüssel der CA
- `fb4ca.crt` → öffentliches CA-Zertifikat (darf verteilt werden)

---

2. Für jeden neuen Nutzer: Zertifikat ausstellen

Im Folgenden wird `<user>` als Platzhalter verwendet, z. B. `tobiasB` oder `janS`.

2.1 Privaten Schlüssel für den Nutzer erzeugen

```
openssl genrsa -des3 -out <user>.key 2048
```

- Es wird eine **PEM-Passphrase für den Nutzer-Key** abgefragt.
- Diese schützt den privaten Schlüssel des Nutzers (z. B. „PEMPASSPHRASE TOBI“).

Beispiel:

```
openssl genrsa -des3 -out tobiasB.key 2048
```

2.2 CSR (Certificate Signing Request) für den Nutzer erzeugen

```
openssl req -new -key <user>.key -out <user>.csr
```

- Du wirst nach diversen Angaben gefragt:
  - `Common Name (CN)` → z. B. `tobias.blaeser@example.com` oder `Tobias Blaeser`
  - `O`, `OU`, `C` etc. nach Bedarf
- Das **Challenge Password** kann in der Regel leer gelassen werden (einfach Enter).

Beispiel:

```
openssl req -new -key tobiasB.key -out tobiasB.csr
```

2.3 CSR mit der CA signieren ? Nutzerzertifikat

Dieser Schritt wird auf dem System ausgeführt, auf dem die CA liegt (`fb4ca.key` und `fb4ca.crt`).

```
openssl x509 -req -days 365 \
  -in <user>.csr \
  -CA fb4ca.crt \
  -CAkey fb4ca.key \
  -CAcreateserial \
  -out <user>.crt \
  -sha256
```

- `-days 365` → Gültigkeit des Nutzerzertifikats (hier: 1 Jahr)
- `-CAcreateserial` → legt einmalig `fb4ca.srl` an und vergibt Seriennummern automatisch

Wichtig:

- Für das **erste** Nutzerzertifikat: `-CAcreateserial` verwenden.
- Für weitere Zertifikate kann (optional) `-CAserial fb4ca.srl` genutzt werden, wenn die Datei bereits existiert.

Beispiel für erstes Nutzerzertifikat:

```
openssl x509 -req -days 365 \
  -in tobiasB.csr \
  -CA fb4ca.crt \
```

```
  -CAkey fb4ca.key \
  -CAcreateserial \
  -out tobiasB.crt \
  -sha256
```

Beispiel für weiteres Nutzerzertifikat (wenn `fb4ca.srl` existiert):

```
openssl x509 -req -days 365 \
  -in janS.csr \
  -CA fb4ca.crt \
  -CAkey fb4ca.key \
  -CAserial fb4ca.srl \
  -out janS.crt \
  -sha256
```

2.4 PKCS#12-Datei (.pfx) für den Nutzer erzeugen

Um dem Benutzer ein **importierbares Paket** aus Key + Zertifikat + CA zu liefern, wird eine `.pfx` (PKCS#12) erstellt:

```
openssl pkcs12 -export \
  -out <user>.pfx \
  -inkey <user>.key \
  -in <user>.crt \
  -certfile fb4ca.crt
```

- Es wird das **Export-Passwort** für die `.pfx` abgefragt.
  - Dieses Passwort ist notwendig, um die `.pfx` auf Clientseite zu importieren (z. B. „EXPORT PW TOBI“).

Beispiel:

```
openssl pkcs12 -export \
  -out tobiasB.pfx \
  -inkey tobiasB.key \
  -in tobiasB.crt \
  -certfile fb4ca.crt
```

Die `.pfx`-Datei enthält:

- privaten Schlüssel des Nutzers (`<user>.key`)
- Nutzerzertifikat (`<user>.crt`)
- CA-Zertifikat (`fb4ca.crt`)

---

3. Nutzung durch den Nutzer

3.1 Dateien und Passwörter an den Nutzer

Der Nutzer erhält:

- Datei: `<user>.pfx`
- Zugehöriges **Export-Passwort** (für den Import)
- Optional: CA-Zertifikat `fb4ca.crt`, falls es auf seinem System noch nicht vertrauenswürdig ist

3.2 Import der `.pfx` beim Nutzer

Je nach Umgebung (Beispiele):

- **Windows (Zertifikatsspeicher)**
  - Doppelklick auf `<user>.pfx`
  - Zertifikatimport-Assistent durchlaufen
  - Persönlichen Zertifikatsspeicher auswählen („Eigene Zertifikate“ / „Personal“)
  - Beim Import das Export-Passwort eingeben
- **Firefox**
  - Einstellungen → „Datenschutz & Sicherheit“ → Abschnitt „Zertifikate“
  - „Zertifikate anzeigen…“ → Reiter „Ihre Zertifikate“ → „Importieren…“
  - `<user>.pfx` wählen, Export-Passwort eingeben
- **Chrome / Edge (Windows)**
  - Nutzen den Windows-Zertifikatsspeicher; Import wie unter Windows beschrieben

3.3 CA-Zertifikat vertrauen

Damit die Client-Zertifikate als gültig erkannt werden, muss die CA (`fb4ca.crt`) als vertrauenswürdige Stammzertifizierungsstelle importiert sein (falls nicht bereits geschehen):

- Unter Windows: in den Zertifikatsspeicher „Vertrauenswürdige Stammzertifizierungsstellen“
- Unter Linux/macOS/Browser: je nach System in den entsprechenden Trust Store / Schlüsselbund importieren

4. Berechtigung auf der Webseite / im Webserver

Damit ein Nutzer tatsächlich auf die geschützte Webseite zugreifen kann, muss der Webserver:

1. **Client-Zertifikate verlangen** (mTLS-Authentifizierung):
   - z. B. in Apache `SSLVerifyClient require`
   - Nginx `ssl_verify_client on;`
2. Der Webserver muss die eigene CA (`fb4ca.crt`) als **trusted client CA** kennen:
   - z. B. Apache `SSLCACertificateFile fb4ca.crt`
   - Nginx `ssl_client_certificate fb4ca.crt;`
3. Optional: Zugriffe auf bestimmte Zertifikate einschränken, z. B.:
   - anhand des `CN` im Zertifikat
   - anhand von Subject Alternative Names (SAN)
   - anhand des Zertifikat-Fingerprints

Die konkrete Konfiguration hängt vom eingesetzten Webserver (Apache, Nginx, IIS, …) bzw. der Anwendung ab.

---

5. Kurzablauf für einen neuen Nutzer (Cheat Sheet)

```
# 1) Nutzer-Key erzeugen
openssl genrsa -des3 -out <user>.key 2048

# 2) CSR erzeugen
openssl req -new -key <user>.key -out <user>.csr

# 3) CSR mit CA signieren (erstes Zertifikat: -CAcreateserial)
openssl x509 -req -days 365 \
  -in <user>.csr \
  -CA fb4ca.crt \
  -CAkey fb4ca.key \
  -CAcreateserial \
  -out <user>.crt \
  -sha256
```

```
# (weitere Nutzer optional mit -CAserial fb4ca.srl)

# 4) PFX für Nutzer erzeugen
openssl pkcs12 -export \
  -out <user>.pfx \
  -inkey <user>.key \
  -in <user>.crt \
  -certfile fb4ca.crt
```

Danach:

- `<user>.pfx` + Export-Passwort an den Nutzer geben
- Nutzer importiert die `.pfx`
- Webserver vertraut der CA `fb4ca.crt` und verlangt Client-Zertifikate
- (optional) Nutzer im Webserver/Applikation explizit freischalten.

Revision #4
Created 22 April 2026 09:28:53 by Tobi
Updated 22 April 2026 09:48:29 by Tobi