



LABORPROTOKOLL

NTSI

PKI-Aufbau mit XCA und TLS-Bereitstellung auf Nginx

Aufbau einer PKI mit Root-, Intermediate- und TLS-Server-Zertifikat inklusive Revokationstest sowie Prüfung der TLS-Verbindung

ÜBUNG 01

LAB REPORT

NTSI

Building a PKI with XCA and TLS deployment on Nginx

Building a PKI with root, intermediate, and TLS server certificates including revocation testing and TLS connection verification

EXERCISE 01

Ausgeführt im Schuljahr 2025/26 von:

Haselsteiner Matteo

5AHITN

Klasse/Gruppe

5AHITN

Gruppe 2

Übungsdatum

24.09.2025

Inhaltsverzeichnis

1 Einleitung	3
2 Theoretischer Hintergrund	4
3 Konfiguration einer PKI	5
3.1 Konfiguration einer PKI in XCA	5
3.1.1 Installation von XCA	5
3.1.2 Erstellen der Zertifikatsdatenbank	5
3.1.3 Erstellung der benötigten Zertifikate	6
3.1.3.1 Root CA Certificate (Certificate Authority)	7
3.1.3.2 Intermediate Certificate	10
3.1.3.3 TLS Server Certificate	12
3.2 Konfiguration einer PKI in OpenSSL	15
4 Konfiguration & Installation eines Webservers	16
4.1 Übersicht	16
4.2 Docker Compose	17
4.3 Nginx	18
4.3.1 Export der Zertifikate und Schlüssel	18
4.3.2 Speichern der Zertifikate	19
4.3.3 Kombinieren der Zertifikatskette	20
4.3.4 Anpassen der Nginx-Konfiguration	20
4.3.5 Testen und Starten von Nginx	20
4.4 Apache	22
5 Browser	23
5.1 Übersicht	23
5.2 Microsoft Edge	24
5.3 Chrome	25
5.4 Firefox	26
5.5 Safari	27
5.5.1 Testumgebung	27
5.5.2 Import der Root-CA und des Intermediate-Zertifikats	27
5.5.3 Verbindungstest	29

Kapitel 1

Einleitung

Ziel dieser Laborübung ist die praktische Umsetzung einer **Public Key Infrastructure (PKI)** mithilfe der Software **XCA** (*X Certificate and Key Management*). Dabei sollen die grundlegenden Schritte zur **Erstellung, Verwaltung und Überprüfung digitaler Zertifikate** nachvollzogen und in einer realen Umgebung getestet werden.

Im Rahmen der Übung wird zunächst eine einfache PKI-Struktur aufgebaut, bestehend aus einem Root-, Intermediate- und Serverzertifikat. Anschließend wird die Funktionsweise der *Certificate Chain*, der Widerrufsmechanismen und der Zertifikatsprüfung demonstriert.

In einer virtuellen Linux-Umgebung wird ein **Webserver** (z. B. *Nginx*) installiert und mit dem erzeugten Serverzertifikat konfiguriert. Auf dem Client-Rechner werden anschließend die notwendigen Zertifikate importiert, um die **Client-Server-Kommunikation über TLS** zu testen.

Erweiterte Aufgabe im Klassenverbund

Im weiteren Verlauf der Übung wird das Verhalten verschiedener Webbrowser bei widerrufenen Zertifikaten untersucht:

- Microsoft Edge
- Mozilla Firefox
- Google Chrome
- Apple Safari

Das Ziel besteht darin, den Zertifikatsfehler zu erkennen und durch das Erstellen sowie Importieren neuer Zertifikate zu beheben.

Die Vorgehensweise, die Ergebnisse und die gewonnenen Erkenntnisse im Laborbericht dokumentiert.

Kapitel 2

Theoretischer Hintergrund

Kapitel 3

Konfiguration einer PKI

3.1 Konfiguration einer PKI in XCA

3.1.1 Installation von XCA

Für die Konfiguration der Public Key Infrastructure (PKI) wird eine virtuelle Maschine mit **Kali Linux** verwendet. In der Regel ist **XCA** (*X Certificate and Key Management*) bereits in der Standardinstallation von Kali Linux enthalten. Sollte dies nicht der Fall sein, kann die Installation über die Paketverwaltung erfolgen:

```
sudo apt update  
sudo apt install xca
```

Nach der Installation kann XCA über das Anwendungsmenü oder den Terminalbefehl `xca` gestartet werden.

3.1.2 Erstellen der Zertifikatsdatenbank

Um Zertifikate verwalten und signieren zu können, muss zunächst eine neue **Zertifikatsdatenbank** erstellt werden. Dies erfolgt in XCA über das Menü:

File → New Database

Im folgenden Dialog wird der Datenbank ein aussagekräftiger Name gegeben sowie ein Speicherort festgelegt. Das Passwortfeld bleibt leer.

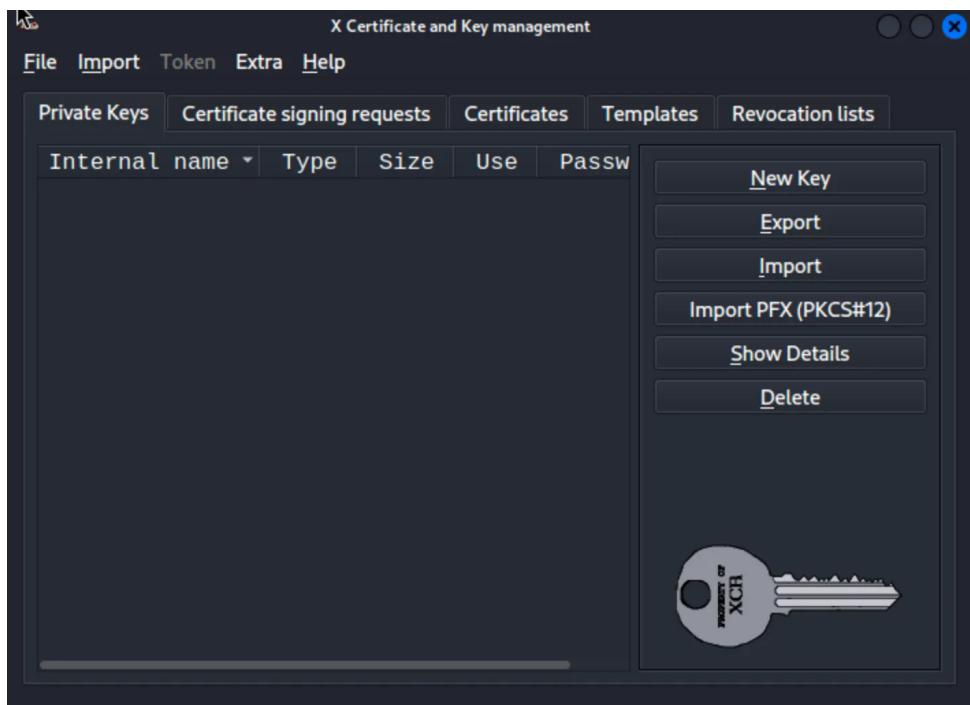


Abb. 3.1: Certificate Database created

Sobald die Datenbank angelegt wurde, kann mit der Erstellung der obersten Instanz begonnen werden: dem **Root-Zertifikat**.

3.1.3 Erstellung der benötigten Zertifikate

Im nächsten Schritt werden die für die PKI-Struktur erforderlichen Zertifikate erzeugt. Dazu gehören:

- a) ein **Root-Zertifikat** (selbstsigniert),
- b) ein **Intermediate-Zertifikat**, das vom Root-Zertifikat signiert wird,
- c) ein **Server-Zertifikat (TLS)**, das vom Intermediate-Zertifikat signiert wird.

Alle Zertifikate werden mit modernen **RSA**-Schlüsseln erstellt (nicht mit einem ED25519-Schlüssel)

Die detaillierte Vorgehensweise zur Erstellung der einzelnen Zertifikate wird in den folgenden Abschnitten beschrieben:

3.1.3.1 Root CA Certificate (Certificate Authority)

Um ein Root-CA-Zertifikat zu erstellen, wechselt man zunächst in den *Certificates*-Tab und klickt auf:

Certificates → New Certificate

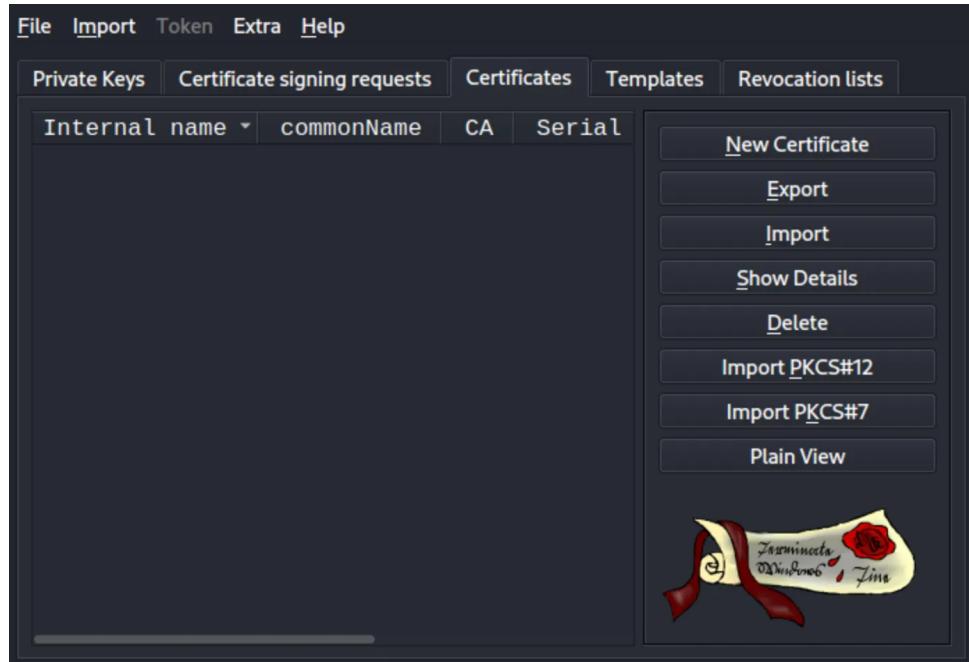


Abb. 3.2: Creating a Root CA Certificate

Daraufhin öffnet sich das Fenster zur Konfiguration des neuen Zertifikats. Im *Source*-Tab muss unter *Signing* die Option "Create a self-signed certificate" aktiviert werden.

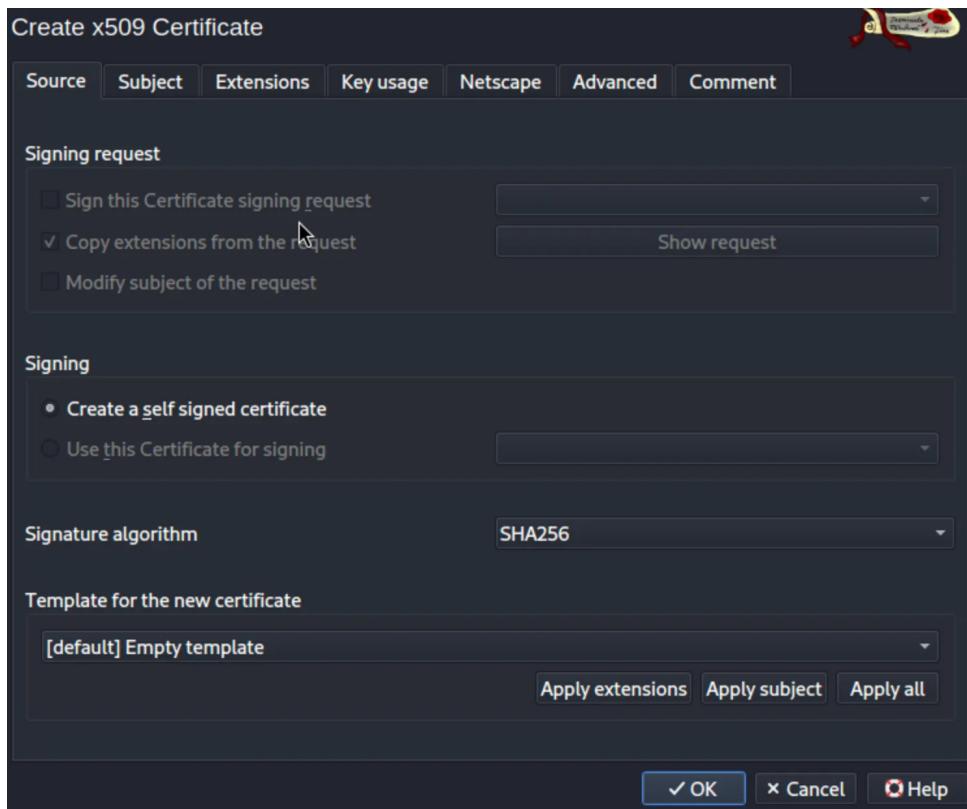


Abb. 3.3: Creating a Root CA Certificate – *Source*-Tab

Anschließend wechselt man in den *Subject*-Tab, um die Informationen für das Zertifikat einzugeben. In diesem Schritt wird außerdem der zugehörige Private Key erstellt. Im unteren Bereich des *Subject*-Tabs befindet sich die Sektion *Private Key*. Hier wird ein neuer Schlüssel mit den unten dargestellten Einstellungen generiert.

(a) Information of the CA Certificate (b) Private Key for CA Certificate

Abb. 3.4: Creating a Root CA Certificate – *Subject*-Tab

Im nächsten Schritt wird im *Extensions*-Tab der Zertifikatstyp festgelegt. Zusätzlich muss der *x509v3 CRL Distribution Point* eingetragen werden.

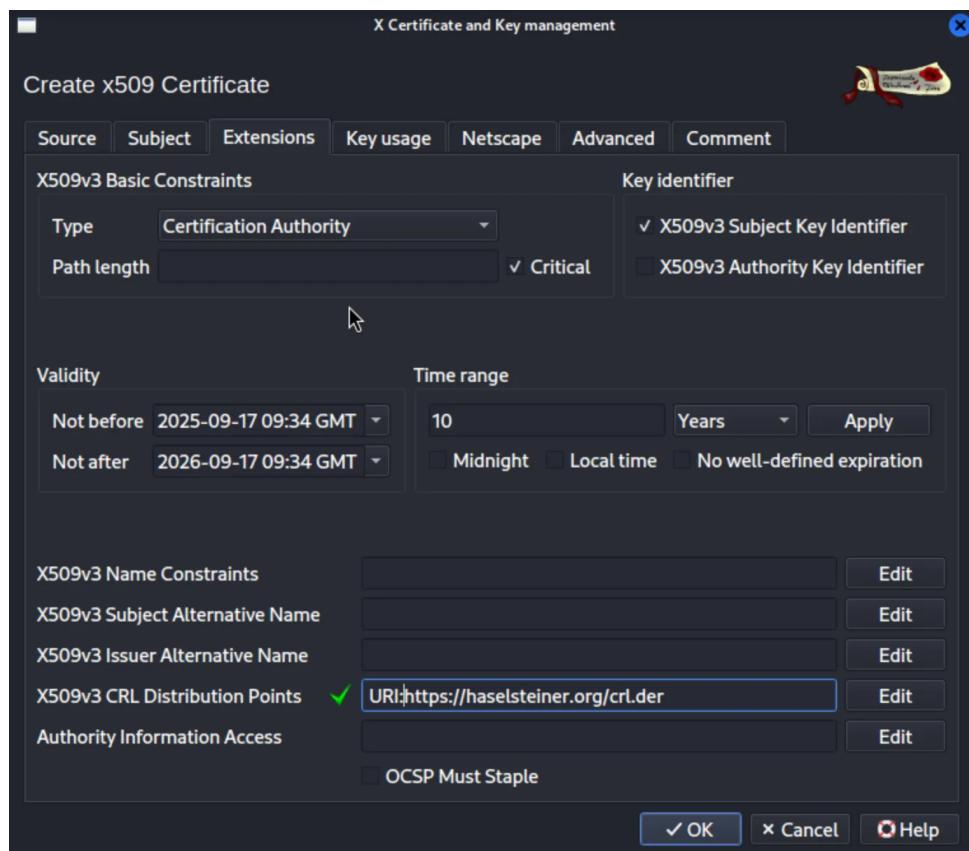


Abb. 3.5: Creating a Root CA Certificate – *Extension*-Tab

Nach erfolgreicher Erstellung erscheint das neue Root-Zertifikat in der Datenbank unter *Certificates*.

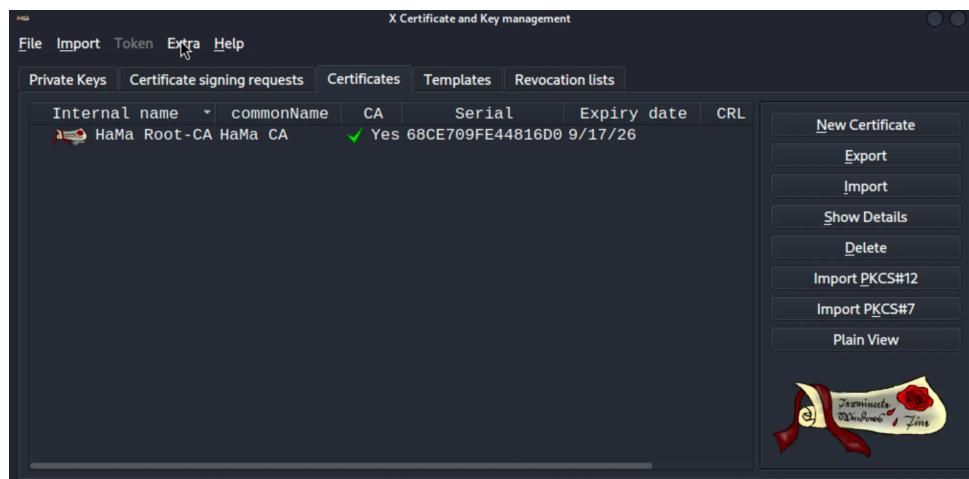


Abb. 3.6: Root-CA Certificate in Database

3.1.3.2 Intermediate Certificate

Das **Intermediate-Zertifikat** wird analog zum Root-Zertifikat erstellt. Der entscheidende Unterschied besteht darin, dass es *nicht selbstsigniert* ist, sondern vom Root-Zertifikat signiert wird.

Im Dialogfeld für die Signatur wird unter *Signing* die Option *Use this Certificate for signing* ausgewählt und anschließend das zuvor erstellte Root-Zertifikat als Aussteller angegeben. Als Vorlage kann unter *Template for the new certificate* das CA-Template verwendet werden, da das Intermediate-Zertifikat ebenfalls als Zertifizierungsstelle fungiert.

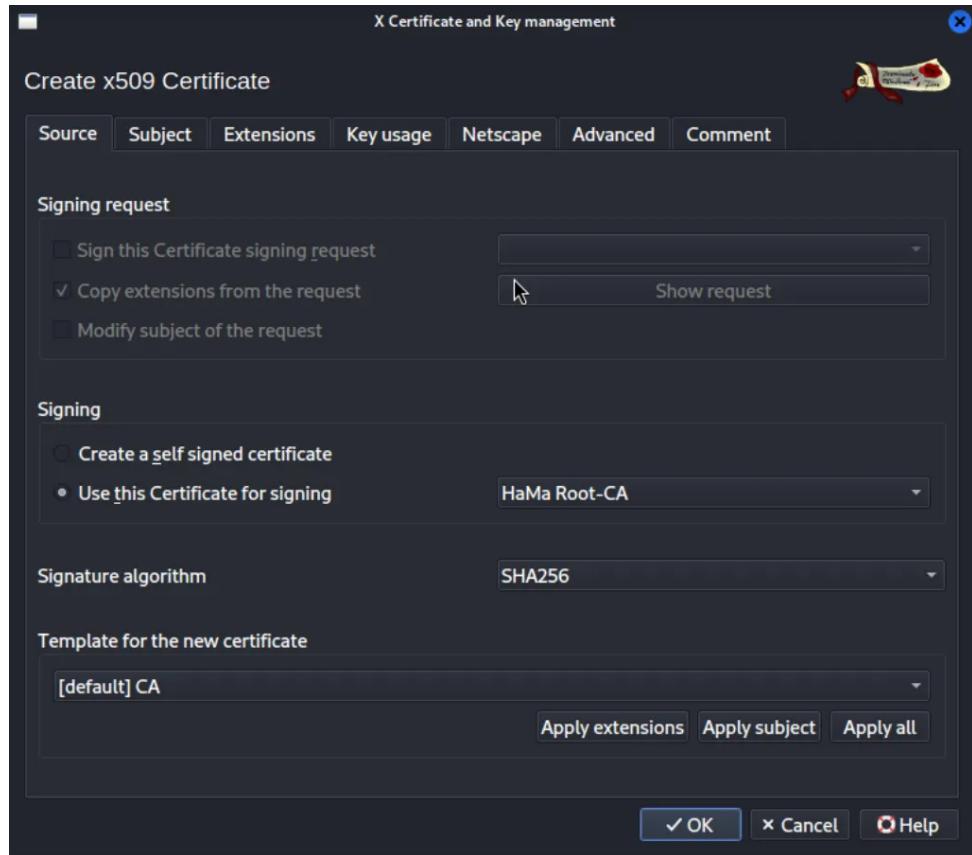


Abb. 3.7: Intermediate Certificate – *Source*-Tab

Im Tab *Subject* werden die Identitätsinformationen für das Intermediate-Zertifikat eingetragen. Hier wird ein neuer privater Schlüssel erzeugt — wichtig ist, dass dieser als **RSA-Schlüssel** erstellt wird (kein ED25519).

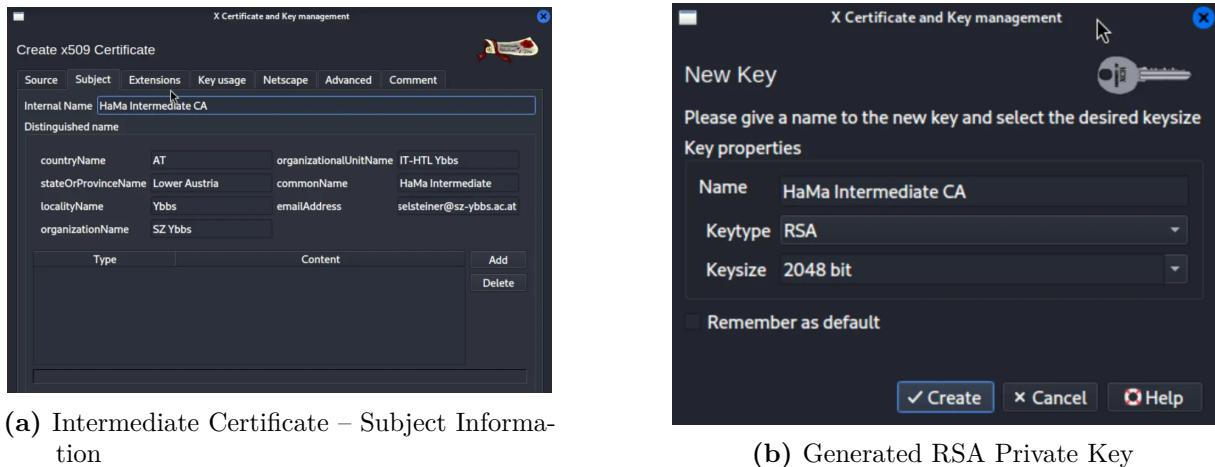


Abb. 3.8: Configuration of the Intermediate Certificate – *Subject*-Tab

Anschließend wird – wie beim Root-Zertifikat – in den Tab *Extensions* gewechselt, um den Zertifikatstyp festzulegen. Da das Intermediate-Zertifikat ebenfalls eine Zertifizierungsstelle darstellt, wird hier die Option *Certification Authority* ausgewählt.

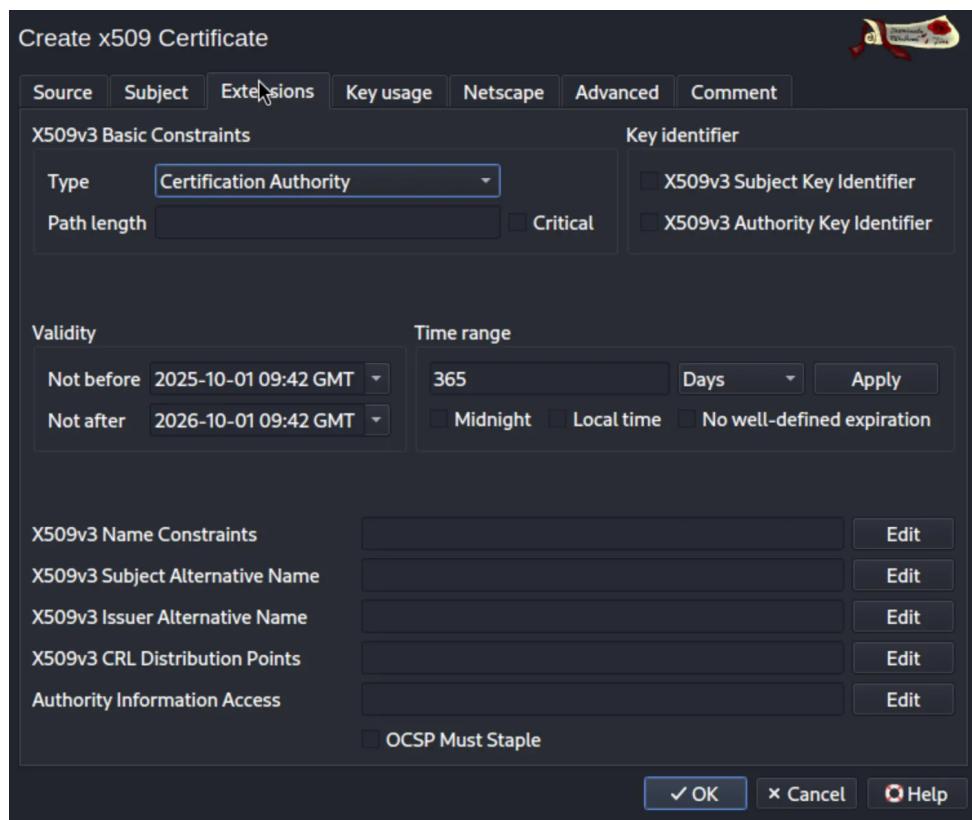


Abb. 3.9: Intermediate Certificate – *Extensions*-Tab

3.1.3.3 TLS Server Certificate

Im letzten Schritt wird das **TLS-Server-Zertifikat** erstellt. Dieses Zertifikat wird später auf dem Webserver eingebunden und dient der verschlüsselten Kommunikation mit den Clients.

Die Erstellung erfolgt analog zu den vorherigen Zertifikaten. Als Signaturzertifikat wird diesmal jedoch das zuvor erzeugte **Intermediate-Zertifikat** verwendet. Unter *Template for the new certificate* kann das vordefinierte *TLS Server*-Template ausgewählt werden.

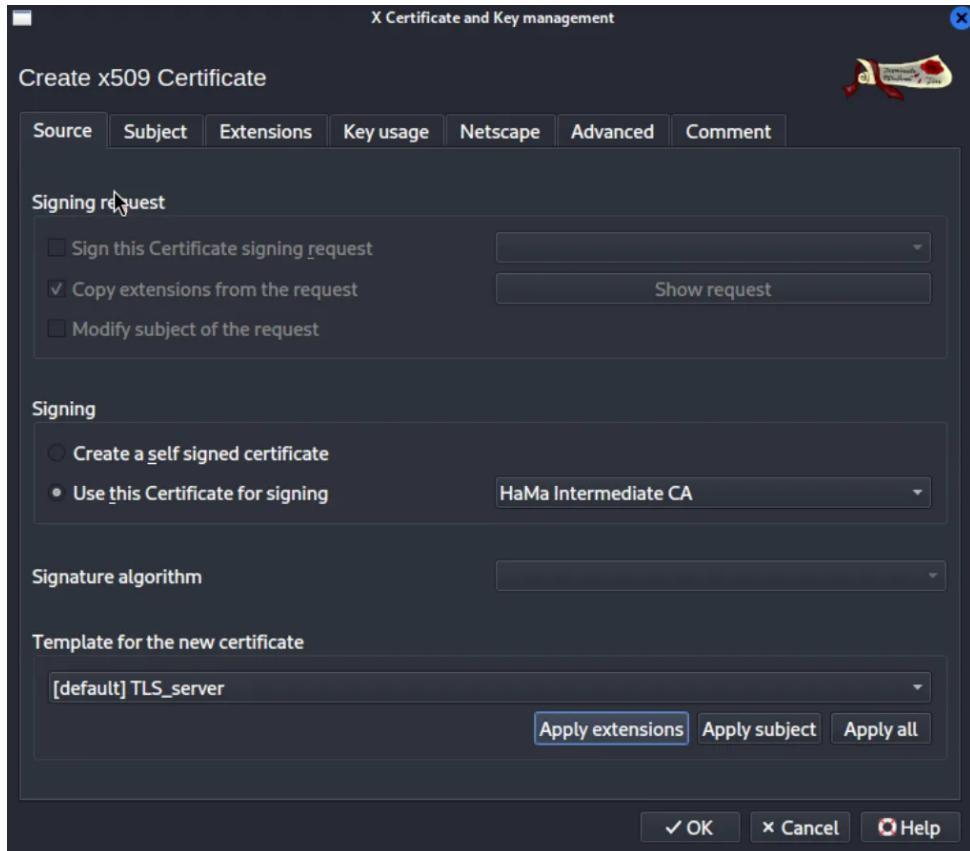
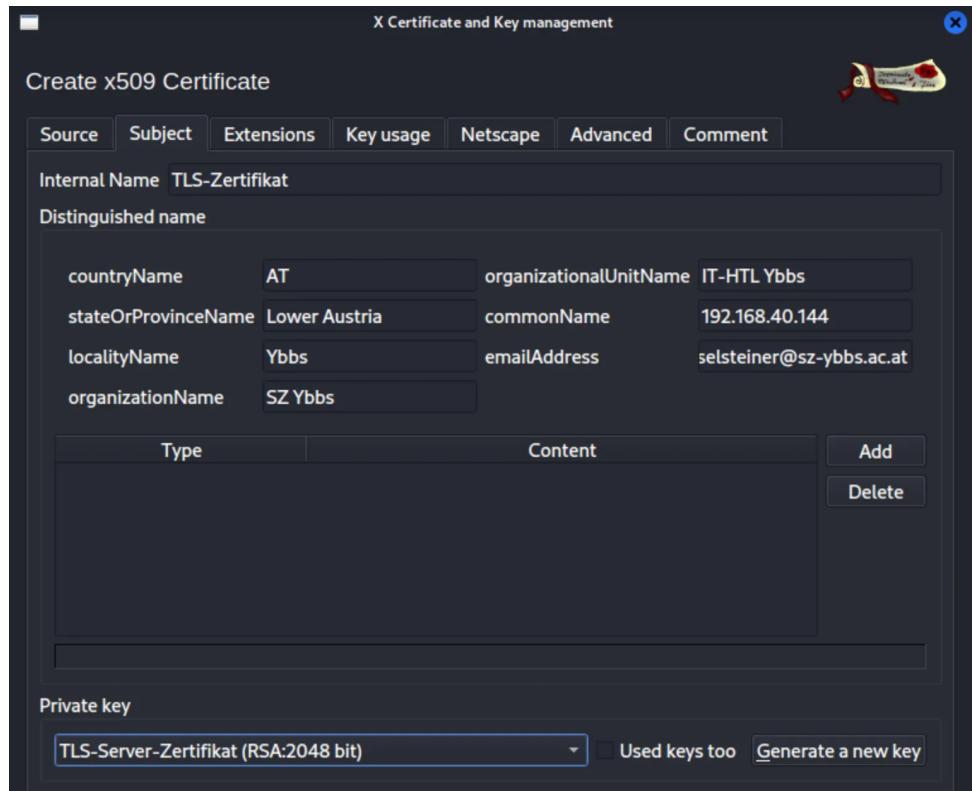


Abb. 3.10: TLS Certificate – *Source*-Tab

Im Tab *Subject* werden die Identitätsdaten des Server-Zertifikats eingetragen. Als *Common Name (CN)* muss hier die **IP-Adresse der virtuellen Maschine** angegeben werden, auf der der Webserver (*nginx*) betrieben wird. Auch dieses Zertifikat erhält einen **RSA-Schlüssel** – es soll kein ED25519-Schlüssel verwendet werden.

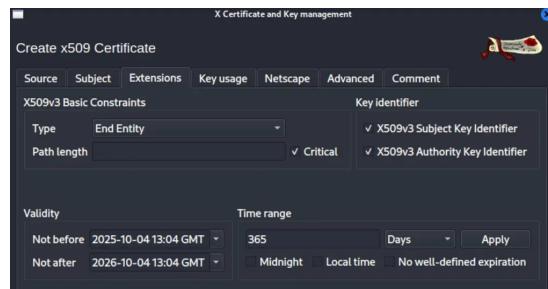
Abb. 3.11: TLS Certificate – *Subject*-Tab

Anschließend wird in den Tab *Extensions* gewechselt, um die Zertifikatseigenschaften festzulegen. Unter *X.509v3 Basic Constraints* wird der Typ **End Entity** ausgewählt, da dieses Zertifikat keine weiteren Zertifikate signiert.

Zusätzlich werden im unteren Bereich der Maske der *Subject Alternative Name (SAN)* sowie der *CRL Distribution Point* eingetragen. Wenn im SAN eine IP-Adresse verwendet wird – was hier der Fall ist – muss diese korrekt als IP angegeben werden, also in der Form:

IP:192.168.40.144

und nicht als URI, also URI:192.168.40.144.

(a) TLS Certificate – *Extensions*-Tab

X509v3 Name Constraints		Edit
X509v3 Subject Alternative Name	✓ IP:192.168.40.144	Edit
X509v3 Issuer Alternative Name		Edit
X509v3 CRL Distribution Points	✓ URI:http://hama.org/crl.der	Edit
Authority Information Access		Edit
OCSP Must Staple		

(b) Subject Alternative Name (IP-Adresse)

Abb. 3.12: TLS Certificate – Einstellungen im *Extensions*-Tab

Nachdem alle drei Zertifikate erstellt wurden, sollte die Zertifikatsdatenbank in XCA wie in Abbildung 3.13 aussehen.

Internal name	commonName	CA	Serial
HaMa Root-CA		✓	
HaMa Intermediate CA		✓	
TLS-Zertifikat	192.168.40.144	No	031CBEDFED49C40

Abb. 3.13: All Certificates in Certificate-Database

3.2 Konfiguration einer PKI in OpenSSL

Kapitel 4

Konfiguration & Installation eines Webservers

4.1 Übersicht

Als Webserver wird in einer separaten virtuellen Maschine **Nginx** verwendet. Ziel ist es, den zuvor erstellten Server Private Key, das Server-Zertifikat sowie das Intermediate-Zertifikat in die Nginx-Konfiguration zu integrieren, um einen sicheren Zugriff über **HTTPS** zu ermöglichen.

In den folgenden Abschnitten wird beschrieben:

- wie die Zertifikate und Schlüssel aus XCA exportiert werden,
- wie diese im System abgelegt und kombiniert werden,
- und wie Nginx für den Betrieb mit TLS/SSL konfiguriert wird.

4.2 Docker Compose

4.3 Nginx

4.3.1 Export der Zertifikate und Schlüssel

Aus der XCA-Datenbank werden folgende Dateien exportiert:

- a) **Server-Zertifikat** (`server.crt`)
- b) **Server Private Key** (`server.key`)
- c) **Intermediate-Zertifikat** (`HaMaIntermediateCA.crt`)

Beim Export ist darauf zu achten, dass:

- das Zertifikat im Format PEM (`*.crt`) gespeichert wird,
- der Private Key im unverschlüsselten Format PEM (`*.key`) exportiert wird,
- und die Dateinamen klar und konsistent benannt sind.

Zertifikate exportiert man in XCA folgendermaßen:

Click on Certificate → Export

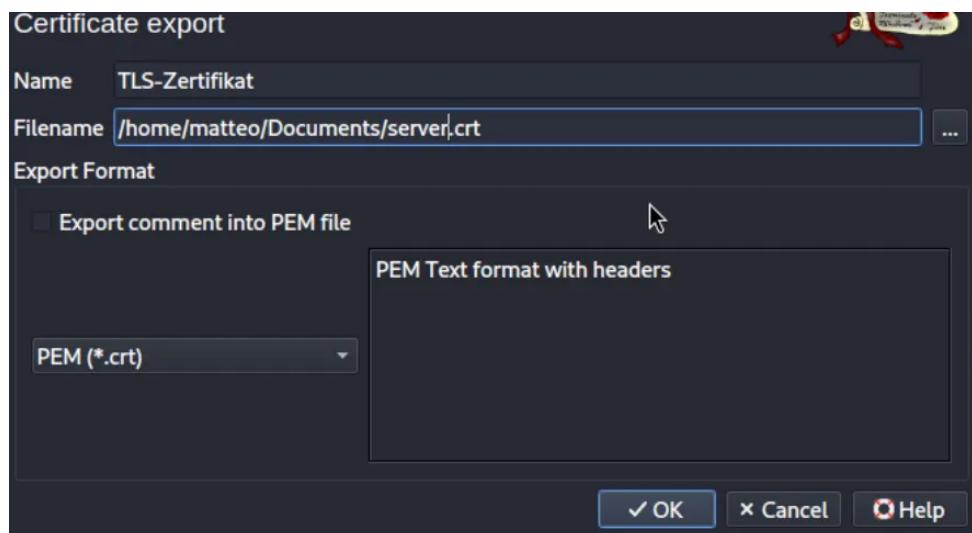


Abb. 4.1: Export Server Certificate

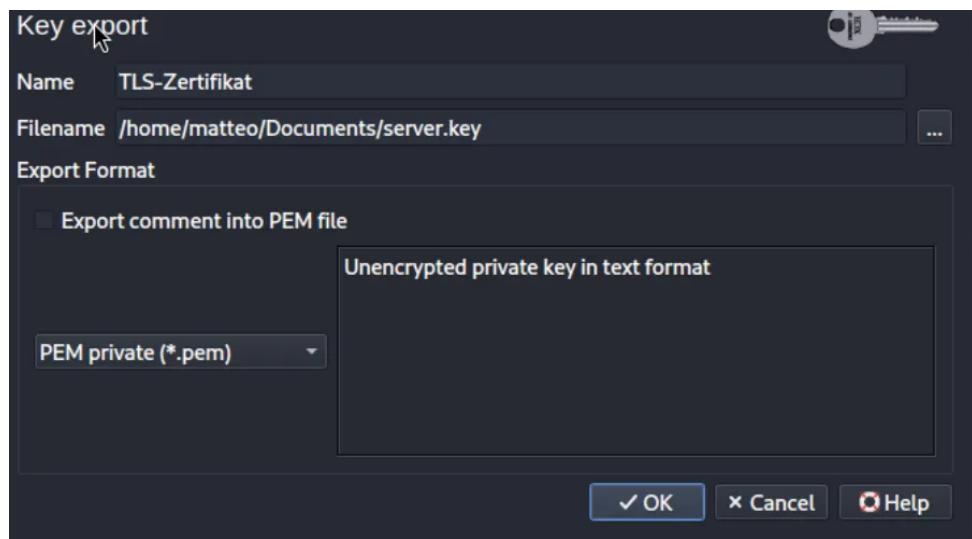


Abb. 4.2: Export Server Private Key

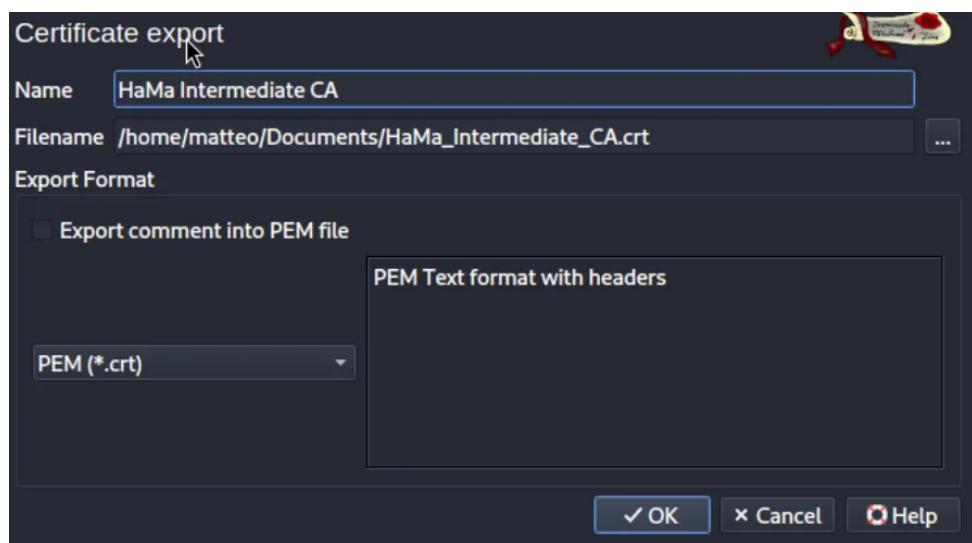


Abb. 4.3: Export Intermediate Certificate

4.3.2 Speichern der Zertifikate

Nachdem die Zertifikate und der Private Key aus XCA exportiert wurden, müssen diese auf den Webserver übertragen und an einem geeigneten Ort gespeichert werden. Hierzu wird ein Verzeichnis *ssl* im Nginx-Konfigurationsverzeichnis erstellt:

```
matteo@kali-03:/etc/nginx$ sudo mkdir ssl
```

Die Dateien können anschließend überprüft werden:

```
matteo@kali-03:/etc/nginx/ssl$ ls
HaMa_Intermediate_CA.crt  server.crt  server.key
```

4.3.3 Kombinieren der Zertifikatskette

Damit Nginx die vollständige Zertifikatskette (Server- + Intermediate-Zertifikat) bereitstellen kann, werden die beiden Dateien zu einem sogenannten *Full Chain Certificate* zusammengeführt:

```
cat /etc/nginx/ssl/server.crt
/etc/nginx/ssl/HaMa_Intermediate_CA.crt >
/etc/nginx/ssl/server_fullchain.crt
```

Die Datei `server_fullchain.crt` enthält nun die gesamte Zertifikatskette, wie sie später im Nginx-Konfigurationsfile angegeben wird.

4.3.4 Anpassen der Nginx-Konfiguration

Die Standardkonfiguration von Nginx befindet sich in:

```
/etc/nginx/sites-available/default
```

Diese Datei wird mit Root-Rechten geöffnet und um die SSL-Parameter ergänzt:

```
server {
    listen 80 default_server;
    listen [::]:80 default_server;

    # SSL configuration
    listen 443 ssl default_server;
    listen [::]:443 ssl default_server;

    ssl_protocols TLSV1.2 TLSv1.3;
    ssl_ciphers HIGH:! aNULL: !MD5;
    ssl_prefer_server_ciphers on;
    ssl_certificate /etc/nginx/ssl/server_fullchain.crt;
    ssl_certificate_key /etc/nginx/ssl/tls-server.key;
    ...
}
```

Achtung: Jeder Block und jede Direktive muss mit einem **Strichpunkt** (;) abgeschlossen werden, da sonst ein Syntaxfehler auftritt.

4.3.5 Testen und Starten von Nginx

Nach dem Bearbeiten der Konfiguration sollte die Syntax geprüft werden:

```
matteo@kali-03:/etc/nginx/ssl$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
```

Anschließend kann der Webserver gestartet werden:

```
matteo@kali-03:/etc/nginx/ssl$ sudo systemctl start nginx
```

4.4 Apache

Kapitel 5

Browser

5.1 Übersicht

In diesem Kapitel wird die Verbindung zwischen dem Client und dem zuvor konfigurierten Nginx-Webserver getestet. Dabei wird überprüft, ob die vom eigenen *Public Key Infrastructure (PKI)*-System ausgestellten Zertifikate vom Client als vertrauenswürdig erkannt werden und ob eine verschlüsselte Verbindung über **HTTPS** erfolgreich aufgebaut werden kann.

5.2 Microsoft Edge

5.3 Chrome

5.4 Firefox

5.5 Safari

5.5.1 Testumgebung

Als Client-System wird ein **macOS**-Rechner verwendet, auf dem die Verbindung zum Webserver hergestellt wird. Die Server-VM basiert auf Kali Linux mit einem installierten Nginx-Webserver, der die zuvor erstellten Zertifikate nutzt.

- **Server:** Kali Linux (Nginx, IP: 192.168.40.144)
- **Client:** macOS Sonoma (Safari, Chrome)
- **Protokoll:** HTTPS (Port 443)
- **Zertifikate:**
 - Root CA: `HaMa_Root_CA.crt`
 - Intermediate CA: `HaMa_Intermediate_CA.crt`
 - Server-Zertifikat: `server.crt`

5.5.2 Import der Root-CA und des Intermediate-Zertifikats

Damit der Client dem Server-Zertifikat vertraut, muss die Root-Zertifizierungsstelle (Root CA) lokal importiert werden.

1. Öffnen der Anwendung **Keychain Access** (Schlüsselbundverwaltung).
2. Wechsel in den Schlüsselbund **System**.
3. Import der Datei `HaMa_Root_CA.crt`.

`File → Import Items → Root CA & Intermediate CA auswählen`

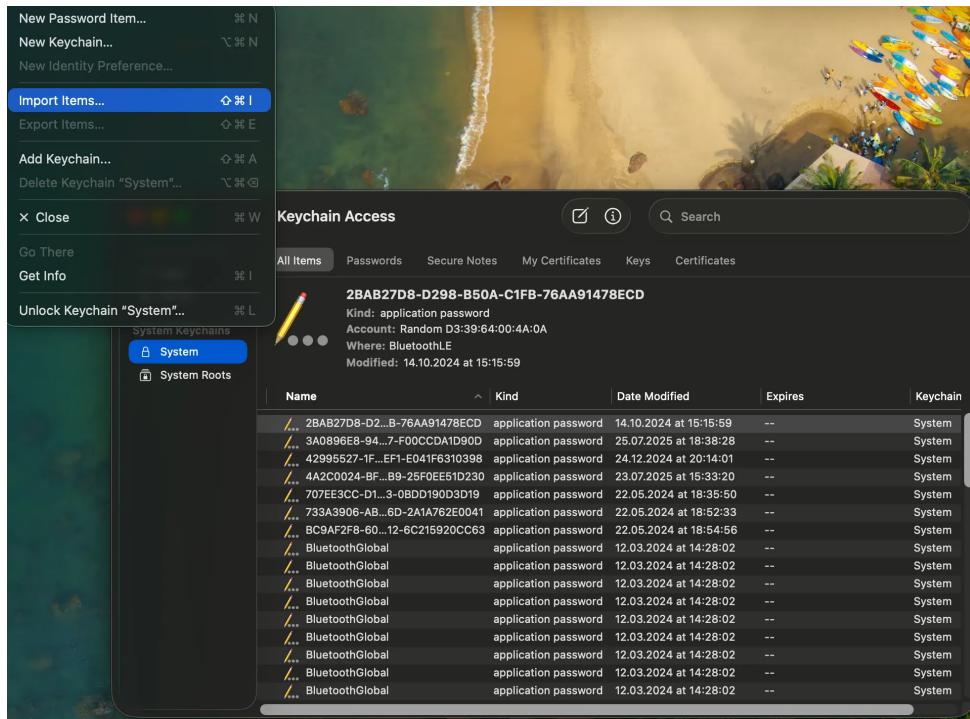


Abb. 5.1: Import Root CA into Keychain Access

Anschließend muss das Zertifikat als vertrauenswürdig markiert werden.

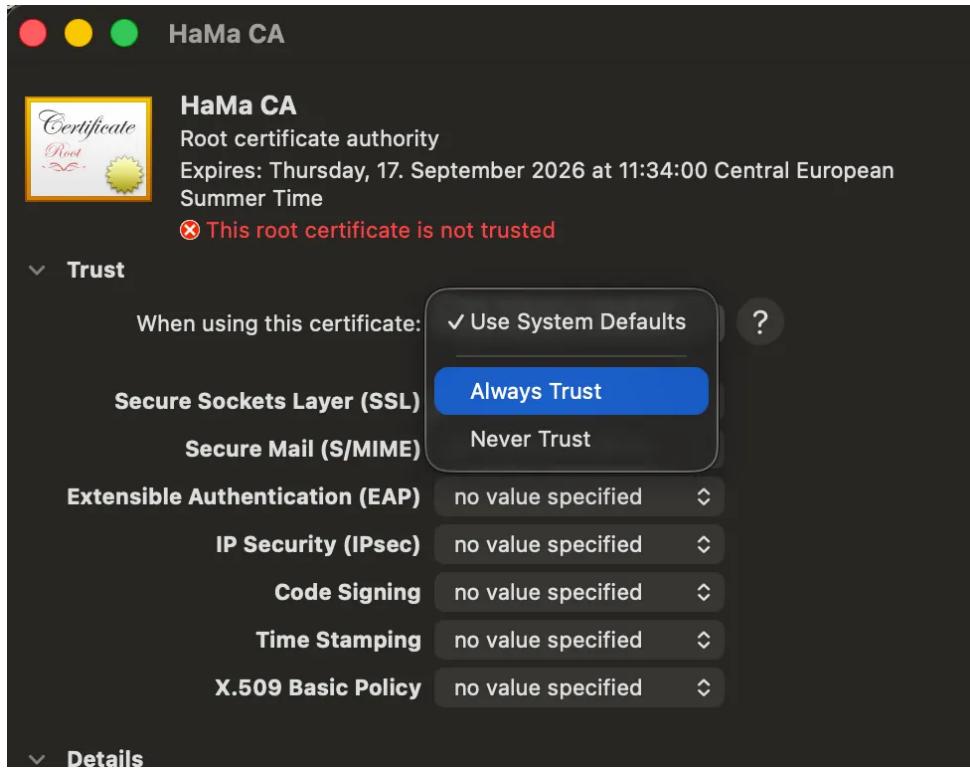


Abb. 5.2: Set Trust Settings for Root CA

Hinweis

Der Import des Intermediate-Zertifikats ist optional, da es beim Verbindungsauflauf vom Server mitgeliefert wird. Dennoch empfiehlt sich der Import, um Zertifikatswarnungen im Browser zu vermeiden.

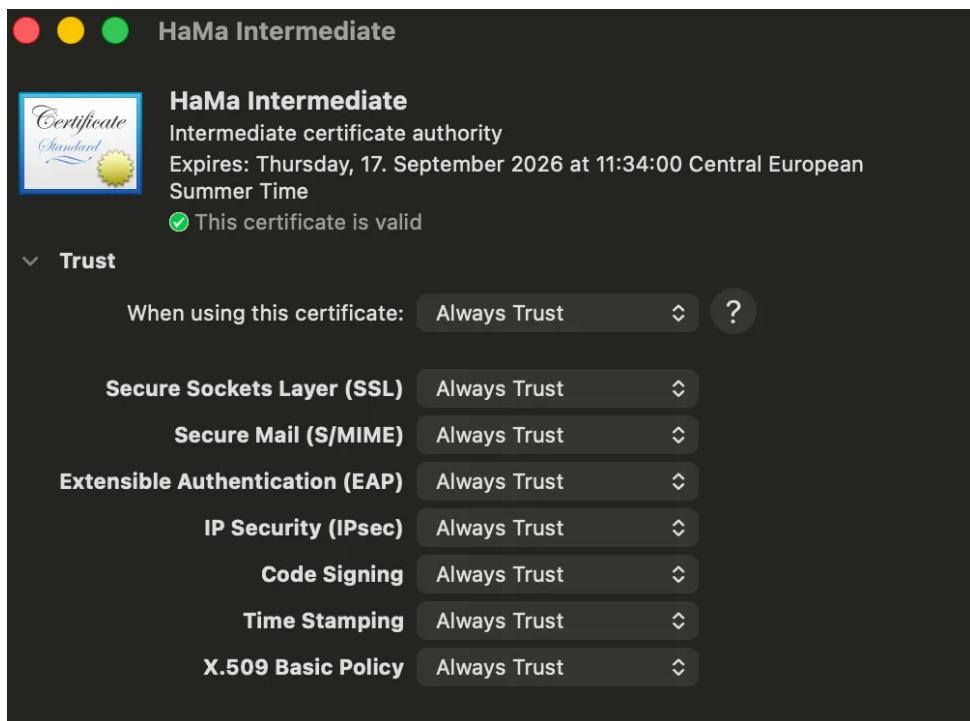


Abb. 5.3: Set Trust Settings for Intermediate Zertificate

5.5.3 Verbindungstest

Nach erfolgreichem Import wird im Browser folgende URL aufgerufen:

<https://192.168.40.144>

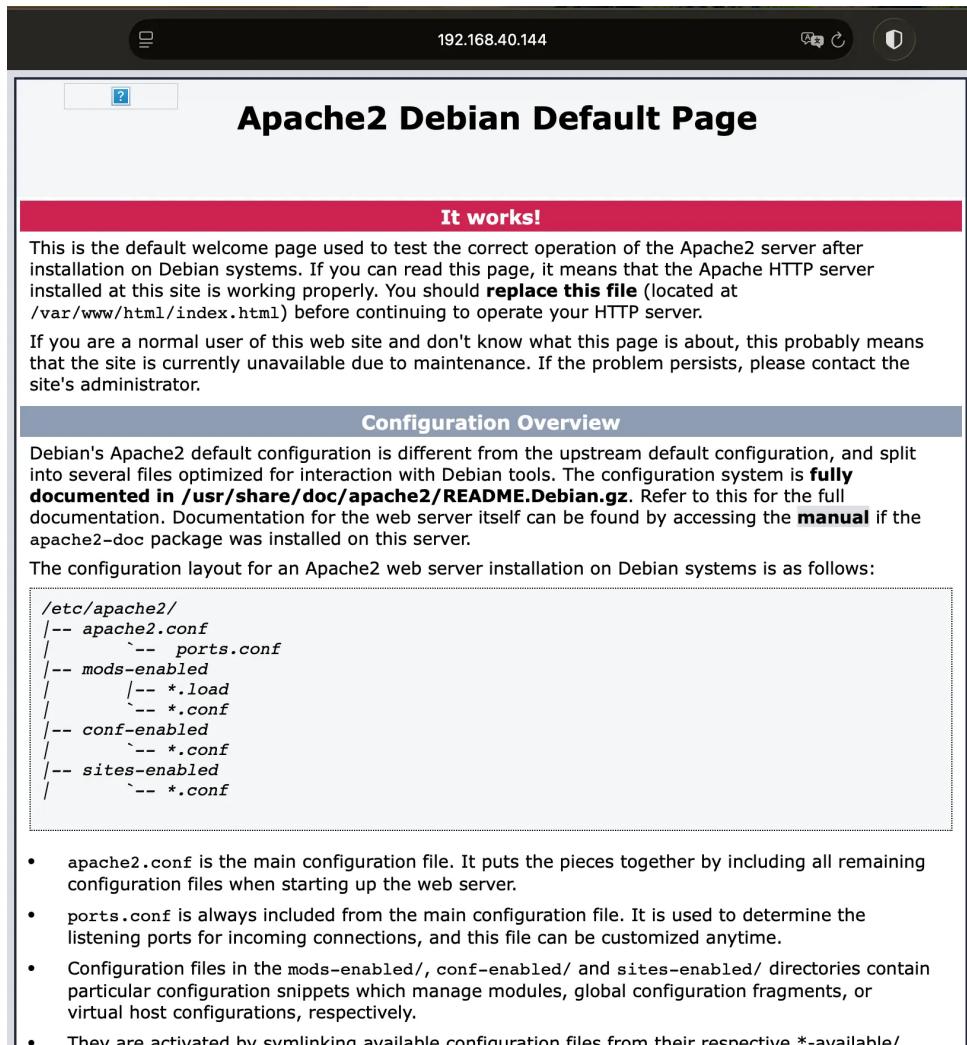


Abb. 5.4: Successful TLS Connection in Safari