Dokumentation EJBCA

0. Einführung

Gruppenmitglieder: Astrid Kuzma-Kuzniarski, Philip Magnus

0.1 Was ist EJBCA?

Enterprise JavaBeans Certificate Authority, kurz EJBCA, ist eine plattformunabhängige CA-Software, in einer Public Key Infrastructure (PKI), geschrieben in Java und kompatibel mit Jakarta-EE-Server.

Die folgende Dokumentation soll zeigen, wie eine EJBCA Instanz aufgesetzt werden kann und welche grundlegenden Features in dieser Instanz genutzt werden kann. Der einfachheit halber wurde das von Keyfactor bereitgestellte und gemanagete `ejbca-ce` Docker Image genutzt.

1. Aufsetzen einer EJBCA-Instanz

1.1 Installation des Docker Images

Mit der folgenden `docker-compose.yml` Datei werden sowohl die EJBCA-Instanz selbst als auch die benötigte DB-Instanz, in diesem Beispiel MariaDB, gestartet.

```
networks:
  access-bridge:
    driver: bridge
  application-bridge:
    driver: bridge
services:
  ejbca-database:
    container_name: ejbca-database
    image: "library/mariadb:latest"
    networks:
      - application-bridge
    environment:
      - MYSQL_ROOT_PASSWORD=foo123
      - MYSQL_DATABASE=ejbca
      - MYSQL_USER=ejbca
      - MYSQL_PASSWORD=ejbca
    volumes:
      - ./datadbdir:/var/lib/mysql:rw
  ejbca-node1:
    hostname: ejbca-node1
    container_name: ejbca
    image: keyfactor/ejbca-ce:latest
    depends_on:
      - ejbca-database
```

```
networks:
  - access-bridge
  - application-bridge
environment:
  - DATABASE_JDBC_URL=jdbc:mariadb://ejbca-database:3306/ejbca?characterEncoding=UTF-8
  - LOG_LEVEL_APP=INFO
  - LOG_LEVEL_SERVER=INFO
  - TLS_SETUP_ENABLED=simple
ports:
  - "80:8080"
  - "443:8443"
```

Um die Container zu starten wird der folgende Befehl in der Konsole ausgeführt:

```
docker compose up -d
```

Die Container werden mit `-d` im detached Modus gestartet. Anschließend kann der ordnungsgemäße Start mit dem Befehl:

```
docker compose logs -f
```

überprüft werden. Eine korrekte Ausgabe sollte ungefähr wie in der folgenden Darstellung aussehen.

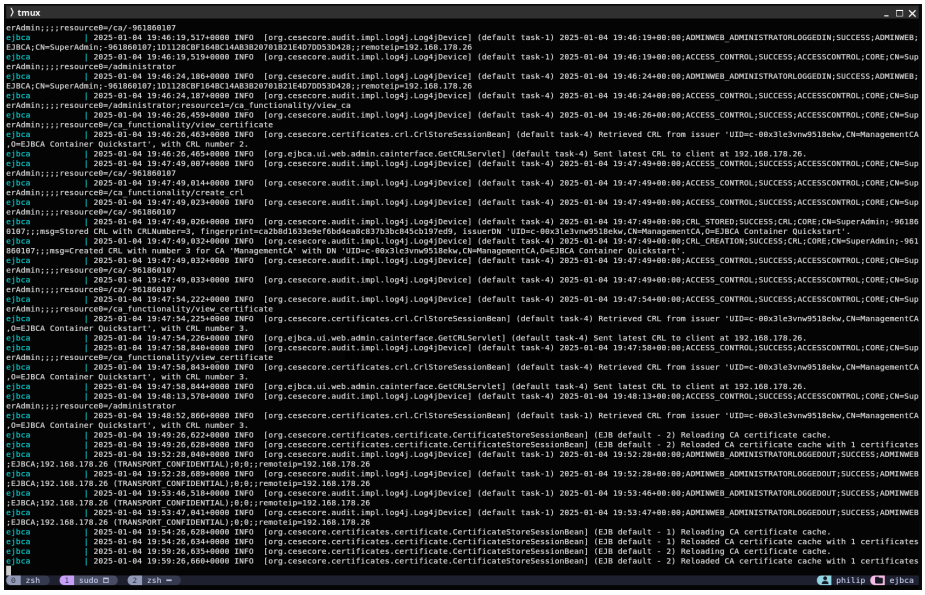

Figure 1: EJBCA-CE Beispiel Log-Ausgabe

Nach dem aufrufen des Web-UI gelangt man auf eine Seite die ähnlich zu folgender Abbildung gestaltet ist.

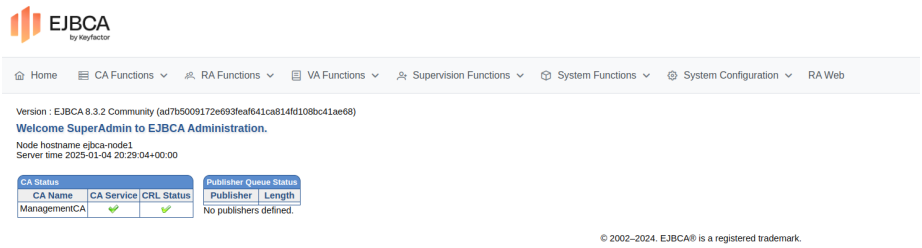

Figure 2: Dashboard EJBCA

2. Features von EJBCA

2.1 Erstellen einer eigenen Root-CA

EJBCA kann verwendet werden um eigenen CAs zu erstellen. Als ersten Schritt kann eine eigene Root-CA erstellet werden.

2.1.1 Anlegen eines Certificate Profile Zum erstellen einer eigenen Root-CA muss zunächst ein eigenes **Certificate Profile** angelegt werden. Unter CA Functions kann der Punkt Certificate Profiles gewählt werden.

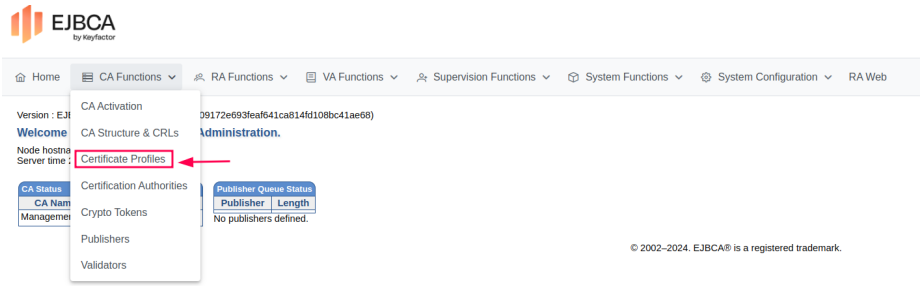

Figure 3: Certificate Profiles Menu

Es sollte sich ein Fenster öffnen in dem eine Übersicht an bereits vorhandenen Certificate Profiles befindet. Hier kann das standard Root-CA Profil der einfachheit halber geklont werden.

Nachdem ein Name für dieses neue Profil vergeben wurde und mit einem Klick auf `create from template` das Profil geklont wurde, kann das neue Profil nun bearbeitet werden.

Ein Einstellungsfenster öffnet sich, in dem die Einstellungen für das neue Profil vorgenommen werden können. Das Fenster sollte ähnlich der Abbildung aufgebaut sein.

Figure 4: Cert Profiles

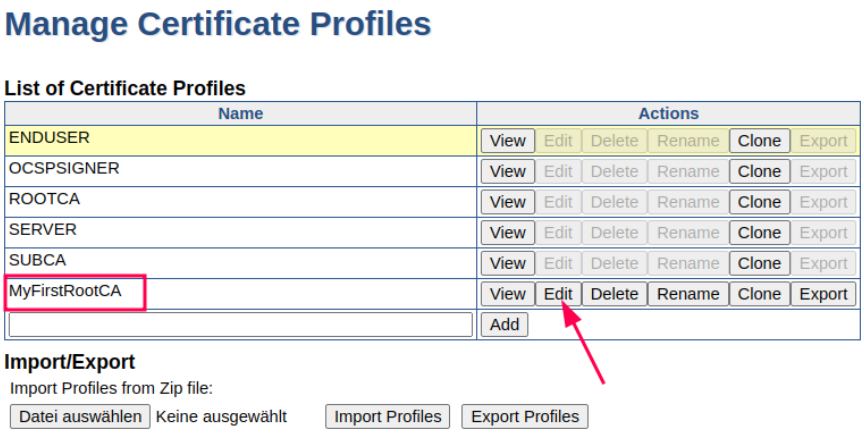

Figure 5: Edit Profile

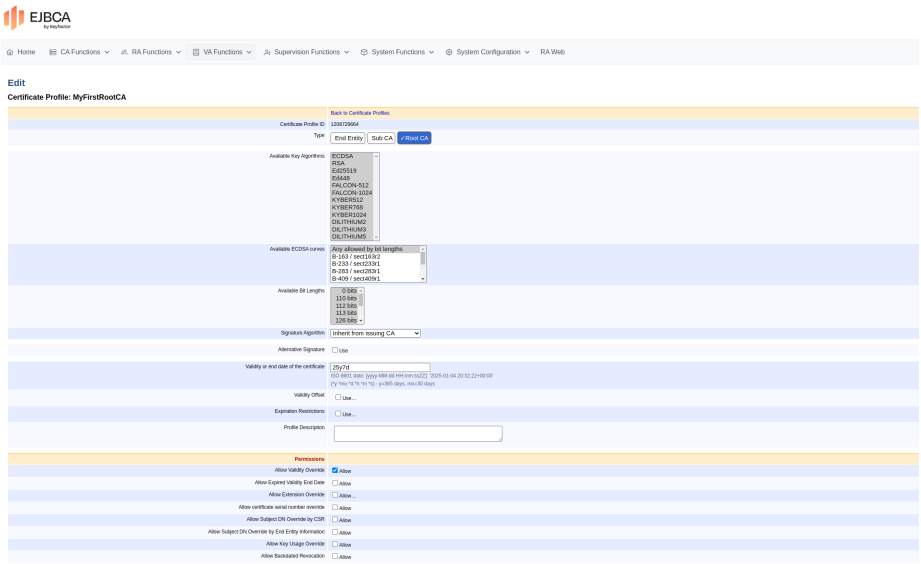

Figure 6: Profile Settings

Die folgenden Einstellungen sollten für eine erste Root-CA wie folgt übernommen werden:

1. Auf der Edit-Seite:
   - **Available Key Algorithms** sollte **RSA** ausgewählt werden
   - **Available Bit Length** sollte **40096 bits** ausgewählt werden
   - **Validity or End Date of the certificate** kann für das Beispiel auf 30y gesetzt werden
2. Unter **X.509v3 Extensions** sollten die folgenden Haken entfernt werden
   - Authority Key ID
   - Subject Alternative Name
   - Issuer Alternative Name
3. Unter **Other Data** sollte der folgende Haken entfernt werden:
   - LDAP DN Order

Anschließend kann das Certificate Profil mit `save` gespeichert werden.

2.1.2 Erstellen eines Crypto Token In einem Crypto Token werden die entsprechenden Keys für Signing und Encryption der Root-CA gespeichert. Für dieses Beispiel wird ein sogenanntes Soft Token erstellt, d.h. die Keys werden in der Datenbank gespeichert. In einer Produktions-Umgebung sollten ein HSM Token (Hardware Security Module) genutzt werden.

Im Crypto Token werden drei Keys gespeichert:

- **SignKey** wird von der CA für Signaturen genutzt

- **EncryptionKey** wird von der CA für alle nötigen Verschlüsselungen genutzt
- **TestKey** wird für Health-Checks der CA genutzt

Um ein Crypto Token zu erstellen muss zunächst unter dem Menüpunkt `CA Functions` der Punkt Crypto Tokens gewählt werden. Hier können vorhandene Crypto Tokens gemanaged und neue Tokens erstellt werden. Unter dem Punkt `Create new...` kann schließlich das neue Crypto Token angelegt werden, Hierfür werden folgende Einstellungen eingetragen:

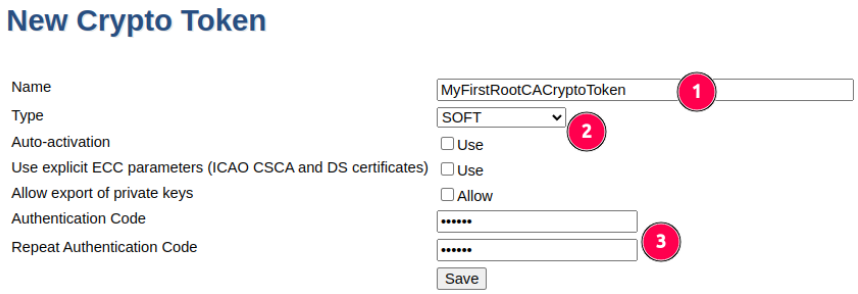

Figure 7: Crypto Token

Es ist hierbei wichtig sich den **Authentication Code** zu merken.

Als nächstes können die jeweiligen Schlüsselpaare generiert werden, im Beispiel werden hierfür die folgenden Einstellungen genutzt:

- **Alias**: myFirstRootCaSignKey0001, myFirstRootCaEncryptKey0001, testKey
- **Key Algorithm**: RSA
- **Key Specification**: 4096

Es gilt als Best-Practice die Sign und Encrypt Keys zu nummerieren um diese durch die Lifetime der CA besser verfolgen zu können.

2.1.3 Anlegen der Root-CA Im folgenden Schritt wird die eigentliche Root-CA angelegt. Unter dem Menüpunkt `CA Functions` muss der Punkt Certificate Authorities gewählt werden, auf der folgenden Seite kann dann unter dem Punkt **Add CA** die neue CA eingetragen werden. Für dieses Beispiel wurde der Name *MyFirstRootCA* gewählt und mit `create...` bestätigt.

Auf der folgenden Edit-Seite wurden die folgenden Einstellungen vorgenommen:

- Als Crypto Token wurde das im vorherigen Schritt erstellte Token gewählt
- Für den defaultKey wurde der myFirstRootCaEncryptKey0001 ausgewählt
- Für den signKey wurde der myFirstRootCaSignKey0001 ausgewählt
- Anschließend wurden noch die folgenden Punkte bearbeitet:

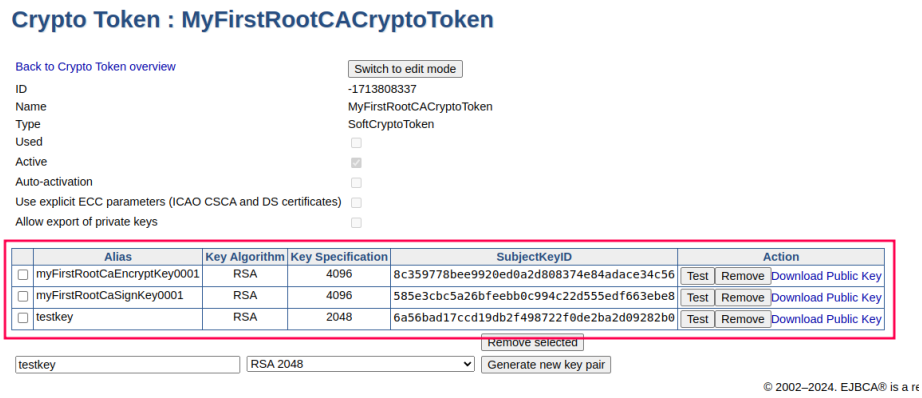

Figure 8: Crypto Keys

- Subject DN: CN = MyFirstRootCA, O = FH Campus, C = AT
- Validity: 30y
- LDAP DN Order: Abgewählt
- CRL Expire Period: 3mo

Mit `Create`wurde die Root-CA abschließend erstellt. In der Liste der CA's taucht diese nun auch auf.

Analog zu diesen Schritten können mehrere Root-CAs aber auch Sub-CAs erstellt werden.

2.2 Aktivieren von CRL und OCSP

2.2.1 Überprüfen ob CRL und OCSP Protokoll aktiviert sind Um in einer CA Certificate Revocation Lists (CRL) und das Online Certificate Status Protocol (OCSP) nutzen zu können, muss zuerst überprüft werden ob die entsprechenden Protokolle in den Systemeinstellungen der EJBCA Instanz aktiviert sind.

Hierfür muss unter dem Reiter System Configuration der Menüpunkt System Configuration ausgewählt werden. In den Einstellungen muss unter dem Reiter Protocols überprüft werden ob die enstprechenden Protokolle aktiviert sind.

Sollten die beiden Protokolle nicht aktiviert sein, können diese einfach mit dem Button `Enable` aktiviert werden.

2.2.2 Aktivieren von CRL und OCSP in CA Profiles Im nächsten Schritt müssen die Protokolle im Certificate Profile aktiviert werden. Hierfür muss unter dem Menüpunkt Certificate Profiles das entsprechende Profil bearbeitet werden.

Unter dem Punkt **X.509v3 Extensions** Müssen die folgenden Haken gesetzt werden.

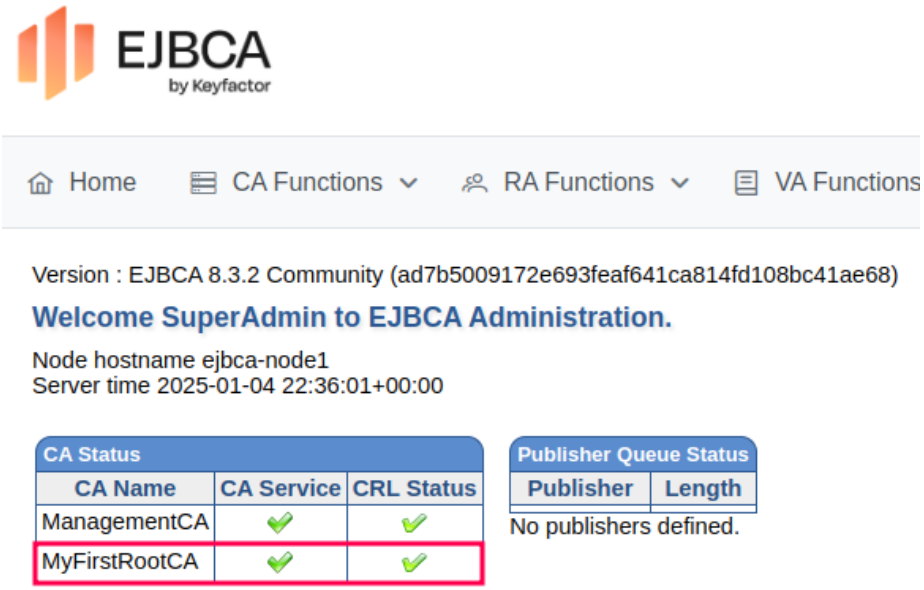

Figure 9: CA List

- **CRL Distribution Points** Use
  - Use CA defined CRL Distribution point (nicht zwingend notwendig, es kann auch im Profil ein Endpunkt gesetzt werden)
- **Authority Information Access**
  - Use CA defined OCSP locator (auch dieser kann im Profil gesetzt werden)

Um das Beispiel einfacher zu gestalten nutzen wir die Locator aus der CA selbst und nicht eigens gesetzte aus dem Certificate Profil

2.2.3 Generieren der Endpunkte in der CA Nun müssen nur noch die Endpunkte an der CA selbst gesetzte werden. Hierfür wird die bereits vorhanden CA, welche das bearbeitete Profil verwendet, bearbeitet. Unter dem Punkt **Default CA defined validation Data** können entweder eigene Endpunkte gesetzt werden oder von EJBCA Endpunkte automatisch generiert werden.

Um das Beispiel einfach zu gestalten, werden die Endpunkte automatisch generiert. Hierzu kann einfach der Button `Generate` genutzt werden.

Zuer veranschaulichung sind die beiden Endpunkte auch noch einmal als ganzer Text angehangen:

CRL Endpoint:

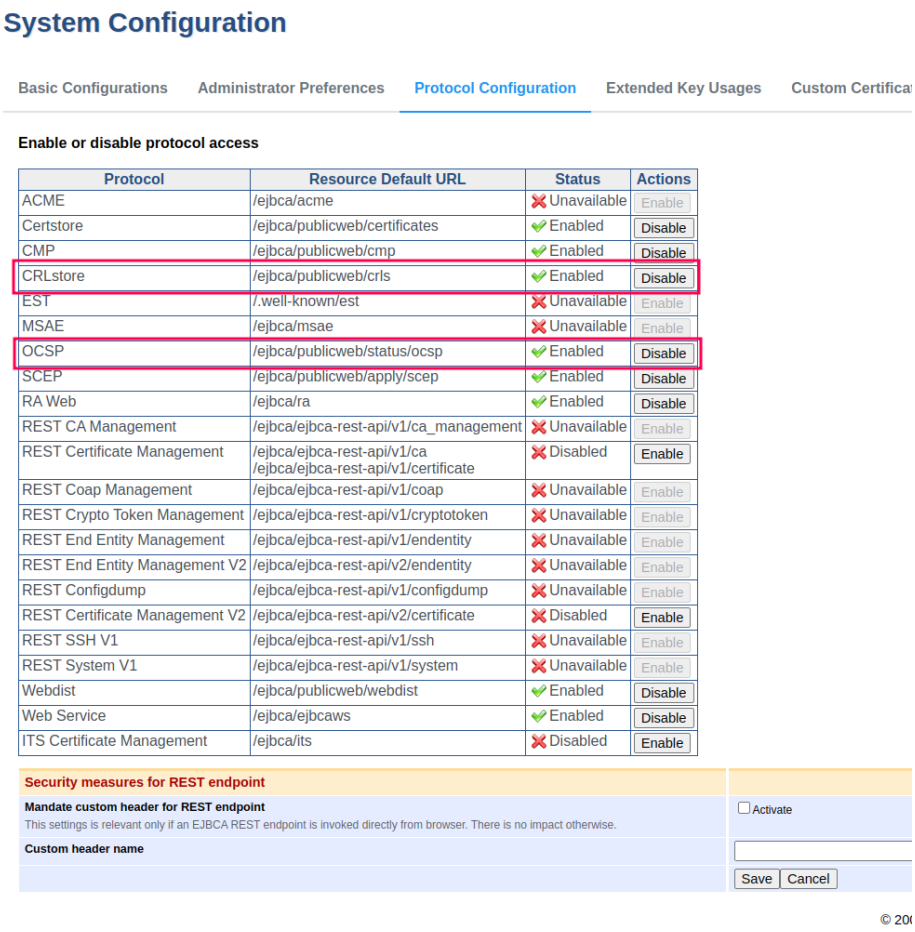

Figure 10: System Configuration > Protocols

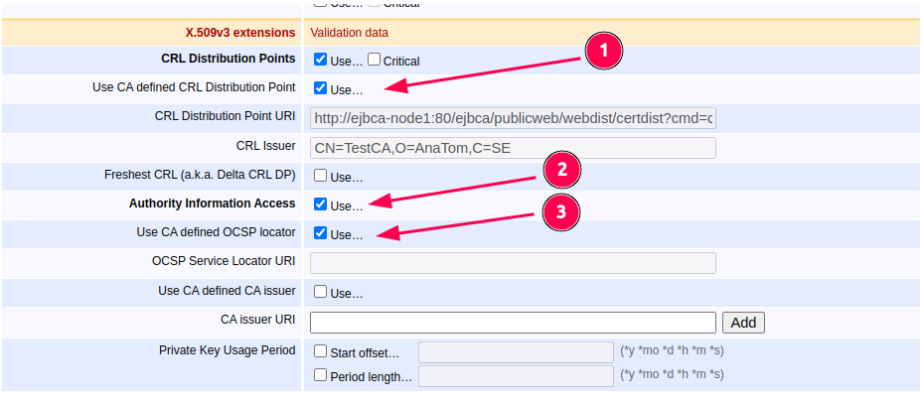

Figure 11: Cert profile

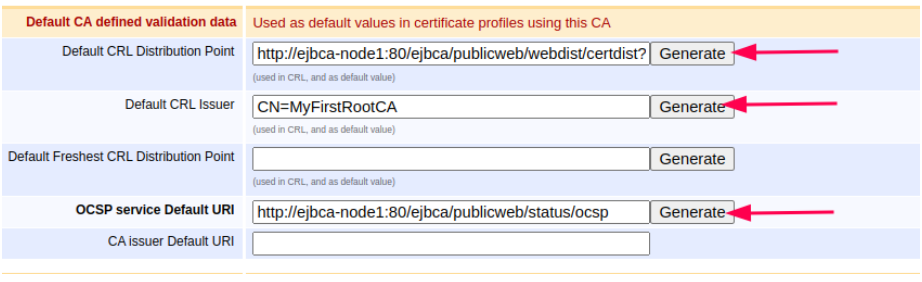

Figure 12: CRL u OCSP Endpoint

`http://ejbca-node1:80/ejbca/publicweb/webdist/certdist?cmd=crl&issuer=CN%3DMyFirstRootCA`

OCSP Endpoint:

`http://ejbca-node1:80/ejbca/publicweb/status/ocsp`

2.2.4 Überprüfen der Funktionalität Über den CRL Endpunkt kann einfach die Revocation List heruntergeladen werden. Dies sollte in etwa wie folgt aussehen.

Download der CRL:

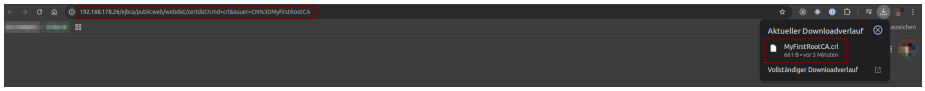

Figure 13: CRL Download

Die CRL kann nun mit `OpenSSL` überprüft werden.

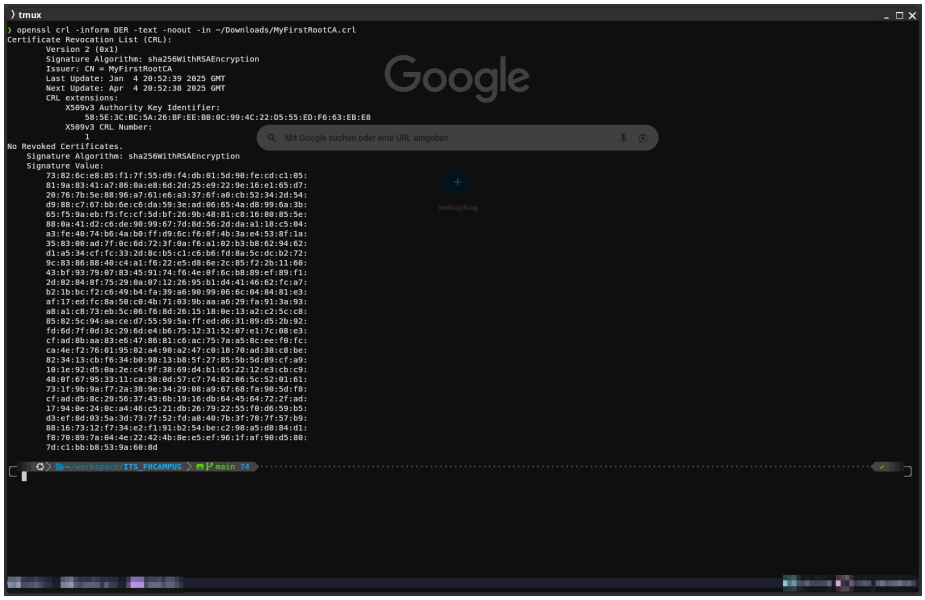

Figure 14: OpenSSL CRL

Wie in der Konsolenausgabe von OpenSSL zu sehen, existieren aktuell keine widerrukfenen Zertifikate.

Mit einem vorhandenen von der CA ausgestellten Zertifikat und dem folgenden `OpenSSL` Befehl lässt sich die funktion des OCSP Endpunktes testen:

`openssl ocsp -issuer MyFirstRootCA.cacert.pem -cert MyWebApp.pem -text -url http://192.168.1`

Sollte das Zertifikat gültig sein sollte die Ausgabe analog zu folgender Ausgabe ausfallen:

```
OCSP Request Data:
    Version: 1 (0x0)
    Requestor List:
        Certificate ID:
          Hash Algorithm: sha1
          Issuer Name Hash: B71FF4FA62D45B811B06E47C1EB4A643D765E332
          Issuer Key Hash: 585E3CBC5A26BFEEBB0C994C22D555EDF663EBE8
          Serial Number: 4B4642020E10A24F257C8FAE1E20FCA265F841CD
    Request Extensions:
        OCSP Nonce:
            0410A2613FA376EEFA11857883CB685EFA5A
OCSP Response Data:
    OCSP Response Status: successful (0x0)
    Response Type: Basic OCSP Response
    Version: 1 (0x0)
    Responder Id: 585E3CBC5A26BFEEBB0C994C22D555EDF663EBE8
    Produced At: Jan  5 20:42:40 2025 GMT
    Responses:
    Certificate ID:
      Hash Algorithm: sha1
      Issuer Name Hash: B71FF4FA62D45B811B06E47C1EB4A643D765E332
      Issuer Key Hash: 585E3CBC5A26BFEEBB0C994C22D555EDF663EBE8
      Serial Number: 4B4642020E10A24F257C8FAE1E20FCA265F841CD
    Cert Status: good
    This Update: Jan  5 20:42:40 2025 GMT

    Response Extensions:
        OCSP Nonce:
            0410A2613FA376EEFA11857883CB685EFA5A
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        ab:21:c2:fa:ea:19:f8:3f:78:1e:1e:0e:f8:3c:2b:0d:8e:d4:
        e4:dc:cc:c6:53:79:ad:c7:c7:f1:f6:00:44:9b:1e:4b:bf:8c:
        a5:6a:22:6c:cd:ab:ef:ac:f0:02:7f:7d:95:b4:32:15:e2:fd:
        90:9b:d4:06:c2:cf:be:a6:0f:ef:70:43:fd:72:e9:67:d1:e1:
        e1:d0:41:43:81:d0:6e:5d:00:46:d9:06:b6:6c:b8:b1:15:03:
        57:62:6e:dd:c8:83:31:5d:d0:20:4a:ce:fe:d1:f8:16:3d:7a:
        d2:3d:bd:65:e5:bf:9f:be:b0:2f:1b:5d:66:aa:db:43:69:be:
        bf:92:6f:d8:86:8c:37:8a:b3:31:c6:4b:c9:7e:2d:8b:1b:37:
        5c:6f:35:60:78:2a:44:39:d0:e2:43:04:64:ec:ad:d6:24:12:
        b6:7d:65:b7:fa:89:f6:3f:e6:b5:06:3f:17:7c:ee:04:59:49:
        78:80:23:cf:58:a8:85:eb:fd:e6:d7:86:d3:93:ca:25:90:9f:
        d8:e2:f8:32:d0:2f:a9:55:50:11:50:12:15:f1:ec:32:7d:bf:
```

```
        ff:07:11:9d:51:d3:87:6c:69:64:03:72:0b:d1:c6:79:0f:27:
        06:64:c2:31:9e:e0:c9:93:02:81:67:47:bd:a2:e8:fc:cc:53:
        04:b2:88:48:e7:c3:b1:46:e1:c8:46:42:fb:8f:c7:af:56:8b:
        f6:0a:db:f6:a1:5a:1f:d8:0a:48:20:5d:aa:ec:48:c0:0e:0e:
        8a:8e:39:0b:1e:09:8b:59:64:87:d3:80:ac:29:14:da:dd:3a:
        2f:92:06:38:0c:79:da:e3:46:a0:be:04:82:d2:e5:49:51:3f:
        48:90:6d:d1:70:c4:fe:af:75:00:5d:0d:a1:bf:85:4d:a0:d2:
        9a:6e:99:e6:5e:32:7c:b7:21:21:35:96:3b:49:2a:7c:8c:95:
        e7:f4:fe:6d:88:68:d4:f9:cc:77:0b:5c:0c:ab:a3:c0:a5:f0:
        25:bd:3a:b0:94:75:9d:f0:6b:4b:17:d5:18:b2:b4:45:90:e3:
        6d:0e:5f:05:7d:7b:43:d1:7f:e0:c9:30:f2:57:58:c1:b2:12:
        df:95:a3:5d:d5:4e:6d:e7:66:3d:c8:b1:c4:43:68:85:d7:51:
        9f:84:5a:c7:e7:63:f8:02:fd:fb:15:dc:75:aa:78:5d:e2:8e:
        29:6a:50:b5:fa:69:7f:dc:c1:d9:ad:46:17:2d:f0:b3:01:3e:
        45:0a:51:f8:c9:a0:eb:46:6b:d2:c2:fe:4b:8f:ce:35:e9:25:
        a3:a4:98:1e:a2:bf:1f:51:b9:9c:78:bb:0f:14:11:5a:3b:76:
        99:e1:29:aa:60:2a:a9:2b
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            65:69:d8:c6:cd:d5:fa:dd:79:80:a9:e0:4d:42:e6:cd:f1:72:ae:a1
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: CN=MyFirstRootCA
        Validity
            Not Before: Jan  4 20:52:39 2025 GMT
            Not After : Dec 28 20:52:38 2054 GMT
        Subject: CN=MyFirstRootCA
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (4096 bit)
                Modulus:
                    00:c0:8e:5c:c5:94:e8:fb:de:7e:6a:c4:25:0c:fe:
                    0e:a7:a9:15:aa:51:98:4f:8a:48:bb:31:ac:bd:91:
                    44:05:a5:2f:e0:60:54:37:69:3c:f7:21:42:1c:f3:
                    bf:b9:fe:cf:05:dd:c3:24:17:4c:04:60:c5:2c:d5:
                    86:60:3c:ec:5b:89:62:f1:2d:66:bc:e5:b6:ab:9c:
                    7d:bd:2f:22:32:06:48:27:17:05:0a:5f:b1:fc:62:
                    ec:dc:65:df:d5:18:3f:51:49:6f:fb:d9:e6:b2:12:
                    d6:ab:d7:56:36:e5:2c:6a:35:61:b1:8e:b0:74:bc:
                    6c:f2:f1:1f:9c:6c:7a:16:61:ad:7e:5c:71:ed:66:
                    c9:f8:14:5c:c9:e0:b7:b4:d9:22:17:d9:f4:fa:f1:
                    b3:dc:c1:69:8e:4f:90:c7:bd:04:07:ec:90:c3:a1:
                    d8:e7:94:ed:d6:b6:90:bd:db:99:a7:55:0f:9c:43:
                    c9:e5:04:9a:bc:77:5b:ee:98:99:f4:0c:0b:1b:98:
                    bb:4f:54:c2:0a:5f:7b:9d:15:24:29:fc:d4:c7:00:
```

```
                ff:5d:0d:85:a2:99:8a:cf:0e:28:a0:0a:61:95:bb:
                20:e8:d2:4a:e7:1d:4e:1f:07:5c:d9:c7:84:bd:6a:
                ac:da:f1:57:7b:0b:9d:58:59:e9:48:cd:85:a5:b2:
                90:8f:8b:da:21:c9:44:04:70:89:a0:ef:92:47:b7:
                af:be:94:70:18:19:c0:88:6f:e5:3e:95:c8:65:c6:
                40:db:dd:c5:76:36:c2:be:a2:f6:05:cd:4d:20:84:
                81:91:0c:9c:86:8d:95:88:da:53:fd:80:b7:ba:1f:
                d6:b7:d3:16:ac:7d:10:80:e5:04:4d:d8:84:6d:b1:
                35:d2:8d:66:f1:84:a5:ff:59:c4:dc:33:2e:03:ff:
                95:6b:2b:be:fe:5e:8e:a9:4b:eb:1a:74:6b:c8:f0:
                8b:86:c3:4f:1d:02:89:81:41:bb:29:a1:73:32:01:
                15:aa:4c:e4:38:87:1f:ef:ad:fa:6c:67:c9:6b:45:
                ad:f0:26:a8:0c:77:26:b4:c3:66:7d:95:74:b4:eb:
                cd:f4:ac:9d:d0:5f:83:68:bc:3d:7c:d1:9f:86:8f:
                83:7e:bb:2b:c7:7d:75:4f:a1:8d:90:a1:5e:0b:bf:
                ab:01:0d:fd:d8:43:15:87:58:6e:08:e2:08:8c:1e:
                be:cd:92:2c:e6:ce:e7:b5:56:e8:2a:c5:0a:e0:e9:
                9d:de:e0:49:5d:f1:00:66:36:a0:db:6e:36:7b:24:
                59:c4:a4:41:d3:3c:8f:dd:d7:3d:6c:d1:8d:99:c7:
                d7:78:3e:cc:e6:dd:4c:7c:ec:cd:a8:ca:c6:1a:f9:
                ef:5c:f9
            Exponent: 65537 (0x10001)
    X509v3 extensions:
        X509v3 Basic Constraints: critical
            CA:TRUE
        X509v3 Subject Key Identifier:
            58:5E:3C:BC:5A:26:BF:EE:BB:0C:99:4C:22:D5:55:ED:F6:63:EB:E8
        X509v3 Key Usage: critical
            Digital Signature, Certificate Sign, CRL Sign
Signature Algorithm: sha256WithRSAEncryption
Signature Value:
    14:6d:fe:bb:81:ab:d0:ba:e4:e6:92:12:d7:8f:bb:7e:09:7b:
    31:89:67:7f:82:ac:4e:e1:7b:0c:e6:f0:0f:d9:d9:02:75:e5:
    fb:bb:74:55:ed:f4:cd:db:57:df:46:fa:a8:a6:3a:64:c9:66:
    e9:6a:3e:f3:3b:51:26:71:74:3f:34:66:d0:08:32:6d:02:8f:
    f0:49:2c:74:2a:f1:65:75:b9:63:54:6a:04:5a:5c:43:cb:91:
    b6:0a:79:e5:f8:68:54:ec:04:d7:15:ab:05:14:a1:0e:71:44:
    4e:03:c7:4f:25:ef:68:4d:bd:dd:1a:5e:ac:c7:39:c5:15:f0:
    af:4c:c8:ab:e8:1e:a5:7d:c1:9c:2e:e4:1f:04:b4:7c:e4:7a:
    83:c4:93:59:fe:c3:dc:dd:5b:99:35:3a:c9:1f:c3:17:db:b3:
    fb:fd:d3:f1:dc:c6:ca:6b:94:4f:79:48:32:d3:c4:82:4f:98:
    59:58:d7:71:2a:4d:08:18:65:80:c3:f3:b9:00:66:5c:37:83:
    7f:ac:69:b9:01:40:4f:e6:2d:03:3a:a4:f2:39:0f:c3:88:83:
    5d:03:98:a7:c8:16:1b:b3:4e:d2:ae:00:b7:13:7f:90:57:cd:
    6d:2c:bb:bf:da:11:b7:9f:22:47:37:d9:13:bf:bf:28:a8:db:
    7c:05:03:6f:c1:b1:07:0e:18:cc:10:d5:46:8f:b6:55:a0:2f:
```

```
        75:66:bb:0d:40:76:72:49:c4:5b:cf:cb:9c:fd:a2:b5:cd:7b:
        05:06:32:40:5c:ef:14:88:82:9c:29:45:42:4d:86:01:88:db:
        42:c6:93:cf:a2:24:c6:68:46:e2:ae:45:bc:8b:aa:1d:b9:00:
        0c:25:2f:e1:0a:64:1b:69:eb:cd:e4:b4:72:5c:df:df:73:2f:
        2a:e6:4a:c1:f6:b9:c8:13:f2:3e:5e:82:3d:88:b4:a0:28:2d:
        93:a0:cf:bd:e4:08:f6:2f:92:bb:74:13:53:32:fd:c6:60:82:
        80:ea:e2:ff:c9:7d:9c:72:8a:6d:ca:d8:74:0c:e3:29:91:e7:
        e0:f5:5f:fb:ad:f0:af:9b:5b:63:a1:25:00:e0:af:86:6e:ac:
        d2:c7:c9:5a:d7:f2:c1:d0:3f:9e:de:26:f1:7a:f6:d3:c0:10:
        0a:43:cf:f7:2c:be:58:e3:ae:a9:d0:30:78:28:92:e7:40:5e:
        61:7d:9f:76:1d:6e:47:57:69:f1:03:5d:33:97:59:ae:5e:4c:
        4b:72:e9:67:2e:10:f2:05:3b:3e:c0:0e:9e:d7:f3:77:5d:ed:
        8e:92:69:ad:b8:08:b1:0c:a5:1f:a9:30:db:72:f8:0a:2f:39:
        e1:63:11:2f:8b:e7:30:d7
-----BEGIN CERTIFICATE-----
MIIFAjCCAuqgAwIBAgIUZWnYxs3V+t15gKngTULmzfFyrqEwDQYJKoZIhvcNAQEL
BQAwGDEWMBQGA1UEAwwNTXlGaXJzdFJvb3RDQTAgFw0yNTAxMDQyMDUyMzlaGA8y
MDU0MTIyODIwNTIzOFowGDEWMBQGA1UEAwwNTXlGaXJzdFJvb3RDQTCCAiIwDQYJ
KoZIhvcNAQEBBQADggIPADCCAgoCggIBAMCOXMWU6PvefmrEJQz+DqepFapRmE+K
SLsxrL2RRAWlL+BgVDdpPPchQhzzv7n+zwXdwyQXTARgxSzVhmA87FuJYvEtZrzl
tqucfb0vIjIGSCcXBQpfsfxi7Nxl39UYP1FJb/vZ5rIS1qvXVjblLGo1YbGOsHS8
bPLxH5xsehZhrX5cce1myfgUXMngt7TZIhfZ9Prxs9zBaY5PkMe9BAfskMOh2OeU
7da2kL3bmadVD5xDyeUEmrx3W+6YmfQMCxuYuO9Uwgpfe50VJCn81McA/10NhaKZ
is8OKKAKYZW7IOjSSucdTh8HXNnHhL1qrNrxV3sLnVhZ6UjNhaWykI+L2iHJRARw
iaDvkke3r76UcBgZwIhv5T6VyGXGQNvdxXY2wr6i9gXNTSCEgZEMnIaNlYjaU/2A
t7of1rfTFqx9EIDlBE3YhG2xNdKNZvGEpf9ZxNwzLgP/lWsrvv5ejqlL6xp0a8jw
i4bDTx0CiYFBuymhczIBFapM5DiHH++t+mxnyWtFrfAmqAx3JrTDZn2VdLTrzfSs
ndBfg2i8PXzRn4aPg367K8d9dU+hjZChXgu/qwEN/dhDFYdYbgjiCIwevs2SLObO
57VW6CrFCuDpnd7gSV3xAGY2oNtuNnskWcSkQdM8j93XPWzRjZnH13g+zObdTHzs
zajKxhr571z5AgMBAAGjQjBAMA8GA1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFFhe
PLxaJr/uuwyZTCLVVe32Y+voMA4GA1UdDwEB/wQEAwIBhjANBgkqhkiG9w0BAQsF
AAOCAgEAFG3+u4Gr0Lrk5pIS14+7fgl7MYlnf4KsTuF7DObwD9nZAnXl+7t0Ve30
zdtX30b6qKY6ZMlm6Wo+8ztRJnF0PzRm0AgybQKP8EksdCrxZXW5Y1RqBFpcQ8uR
tgp55fhoVOwE1xWrBRShDnFETgPHTyXvaE293RperMc5xRXwr0zIq+gepX3BnC7k
HwS0fOR6g8STWf7D3N1bmTU6yR/DF9uz+/3T8dzGymuUT3lIMtPEgk+YWVjXcSpN
CBhlgMPzuQBmXDeDf6xpuQFAT+YtAzqk8jkPw4iDXQOYp8gWG7NOOq4AtxN/kFfN
bSy7v9oRt58iRzfZE7+/KKjbfAUDb8GxBw4YzBDVRo+2VaAvdWa7DUB2cknEW8/L
nP2itc17BQYyQFzvFIiCnClFQk2GAYjbQsaTz6IkxmhG4q5FvIuqHbkADCUv4Qpk
G2nrzeS0clzf33MvKuZKwfa5yBPyPl6CPYi0oCgtk6DPveQI9i+Su3QTUzL9xmCC
gOri/8l9nHKKbcrYdAzjKZHn4PVf+63wr5tbY6ElAOCvhm6s0sfJWtfywdA/nt4m
8Xr2O8AQCkPP9yy+WOOuqdAweCiS50BeYX2fdh1uR1dp8QNdM5dZrl5MS3LpZy4Q
8gU7PsAOntfzd13tjpJprbgIsQylH6kw23L4Ci854WMRL4vnMNc=
-----END CERTIFICATE-----
MyWebApp.pem: good
    This Update: Jan  5 20:42:40 2025 GMT
```

2.3 Erstellen einer 2-Tier PKI Hierarchie

2.3.1 Erstellen der CA Profile Analog zum erstellen eines Certificate Profils in 2.1.1 werden in diesem Schritt ein CA Profil für die Root-CA und eines für die Sub-CA angelegt.

Das Profil für die Root-CA wird aus dem bereits angelegten Profil geklont und als **MyPKIRootCAProfile** benannt.

Die meisten Einstellungen werden beibehalten, der Key Algorithm aber wie folgt eingestellt:

- **Available Key Algorithms** ECDSA auswählen
- **Available ECDSA curves** P-256 / prime256v1 / secp256r1 auswählen

Anschließend mit `Save` das Profil Speichern.

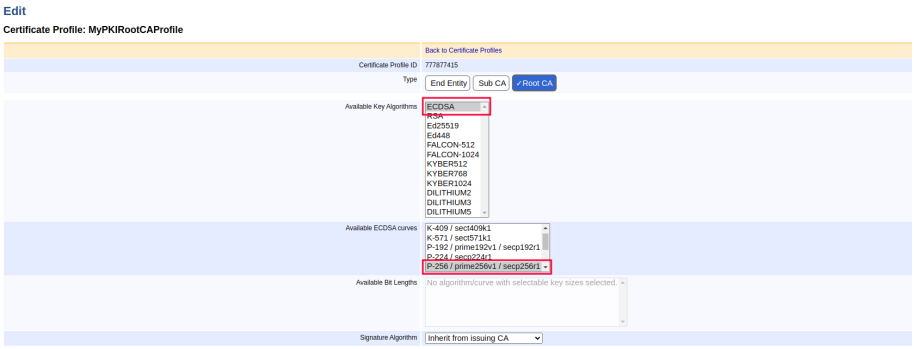

Figure 15: MyPKIRootCAProfile

Analog wird nun das Profil für die Sub-CA aus dem bereits vorhandenen **SUBCA** Profil geklont und mit **MYPKISubCAProfile** benannt.

Hier müssen folgende Einstellungen vorgenommen werden:

- **Available Key Algorithms** ECDSA auswählen
- **Available ECDSA curves** select P-256 / prime256v1 / secp256r1 asuwählen
- **Signature Algorithm** bestätigen, dass *Inherit from Issuing CA* ausgewählt ist
- **Validity or end date of the certificate** auf *15y* setzen
- Unter **X.509v3 extensions** die folgenden Anpassungen:
  - **Path Length Constraint** auswählen und auf 0 setzen um sicher zu stellen, dass diese Sub-CA keine weiteren Sub-CAs unter sich bestätigen kann. Es sollen von dieser CA nur End Entitäts Zertifikate ausgegeben werden
- Unter **X.509v3 extensions - Validation data** folgende Anpassungen:
  - **CRL Distribution Points** auswählen

  - **Use CA defined CRL Distribution Point** auswählen
  - **Authority Information Access** auswählen
  - **Use CA defined OCSP locator** auswählen
  - **Use CA defined CA issuer** asuwählen
- Unter **Other Data** LDAP DN order abwählen

Abschließend mit `Save` das Profil speichern.

2.3.2 Anlegen von Crypto Tokens Wie bereits in 2.1.2 müssen auch hier für die beiden CAs jeweils ein Crypto Token mit Signing und Encryption-Keys angelegt werden.

Zuerst wird ein Crypto Token für die Root-CA erstellt unter dem Namen **MyPKIRootCACryptoToken**.

Anschließend werden die folgenden Key-Pairs angelegt:

- **myPkiRootCaSignKey0001**: P-256 / prime256v1 / secp256r1
- **myPkiRootCaEncryptKey0001**: RSA 4096
- **testKey**: P-256 / prime256v1 / secp256r1

Analog dazu wird das Token für die Sub-CA angelegt:

Name: **MyPKISubCACryptoToken**

- **myPkiSubCaSignKey0001**: P-256 / prime256v1 / secp256r1
- **myPkiSubCaEncryptKey0001**: RSA 4096
- **testKey**: P-256 / prime256v1 / secp256r1

	Alias	Key Algorithm	Key Specification	SubjectKeyID	Action
☐	myPkiSubCaEncryptKey001	RSA	4096	34a55255b05f375bf768ace2e0803d1988151280	Test Remove Download Public Key
☐	myPkiSubCaSignKey001	ECDSA	prime256v1 / secp256r1 / P-256	34f489757a956059f32983d876bf4041c1ca2c4b	Test Remove Download Public Key
☐	testKey	ECDSA	prime256v1 / secp256r1 / P-256	d9f1017ab024725638365d4abb220777081618ee	Test Remove Download Public Key

Remove selected

Figure 16: SubCA Token

2.3.3 Anlegen der CAs Unter Punkt 2.1.3 wurde bereits gezeigt wie eine Root-CA angelegt werden kann, ähnlich dazu wird hier die Root-CA und die Sub-CA angelegt. Hierfür werden folgende Einstellungen vorgenommen:

Einstellungen der Root-CA:

- Name: **MyPKIRootCA-G1**
- **MyPKIRootCACryptoToken** auswählen in der Crypto Token Liste.
- **Signing Algorithm**: SHA512withECSDSA
- Für den defaultKey wurde der **myPkiRootCaEncryptKey0001** ausgewählt
- Für den signKey wurde der **myPkiRootCaSignKey0001** ausgewählt
- **Subject DN**: “CN = My PKI Root CA - G1, O = FH Campus, C = AT”
- **Signed By**: Self Signed

- **Certificate Profile**: MyPKIRootCAProfile
- **Validity**: 30.
- **LDAP DN order** abwählen
- Um CRL und OCSP zu nutzen:
  - **CRL Expire Period**: 3mo
  - **CRL Overlap Time**: 0m
  - **Default CRL Distribution Point**: `Generate`
  - **OCSP service Default URI**: `Generate`
  - **CA issuer Default URI**: http://ejbca-node01/certs/MyPKIRootCA-G1.crt

Einstellungen der Sub-CA:

- Name **MyPKISubCA-G1** On the Create CA page, update the following:
- **MyPKISubCACryptoToken** auswählen in der Crypto Token List
- **Signing Algorithm**: SHA256withECSDSA
- Für den defaultKey wurde der **myPkiSubCaEncryptKey0001** ausgewählt
- Für den signKey wurde der **myPkiSubCaSignKey0001** ausgewählt
- **Subject DN**: "CN = My PKI Sub CA - G1, O = FH Campus, C = AT"
- **Signed By**: MyPKIRootCA-G1
- **Certificate Profile**: MyPKISubCAProfile
- **Validity**: Specify 15y
- **LDAP DN order** abwählen
- Um CRL und OCSP zu nutzen:
  - **CRL Expire Period**: 3d
  - **CRL Issue Interval**: 1d
  - **CRL Overlap Time**: 0m
  - **Default CRL Distribution Point**: `Generate`
  - **OCSP service Default URI**: `Generate`
  - **CA issuer Default URI**: http://ejbca-node01/certs/MyPKISubCA-G1.crt

Wenn beide CAs korrekt eingestellt und erstellt wurden, sollten diese auch in der CA übersicht auftauchen. Nun können über die Sub-CA Zertifikate ausgestellt werden, welche über die Zertifikatskette auch mit der Root-CA verbunden sind.

2.4 Erstellen und widerrufen von TLS End-Entity Zertifikaten

2.4.1 Erstellen von TLS Zertifikaten Zum erstellen eines TLS Zertifikats muss das RA Web aufgerufen werden. Hier wird dann mit `Neue Zertifikatsanfrage` ein neues Zertifikat bei der CA angefordert.

Im nächsten Schritt können die Keys entweder von EJBCA generiert werden oder man erstellt einen eigenen CSR und lädt diesen hoch. Hier wird nur die zweite Version gezeigt:

Zuerst muss eine .cnf Datei erstellt werden mit der ein CSR generiert wird:

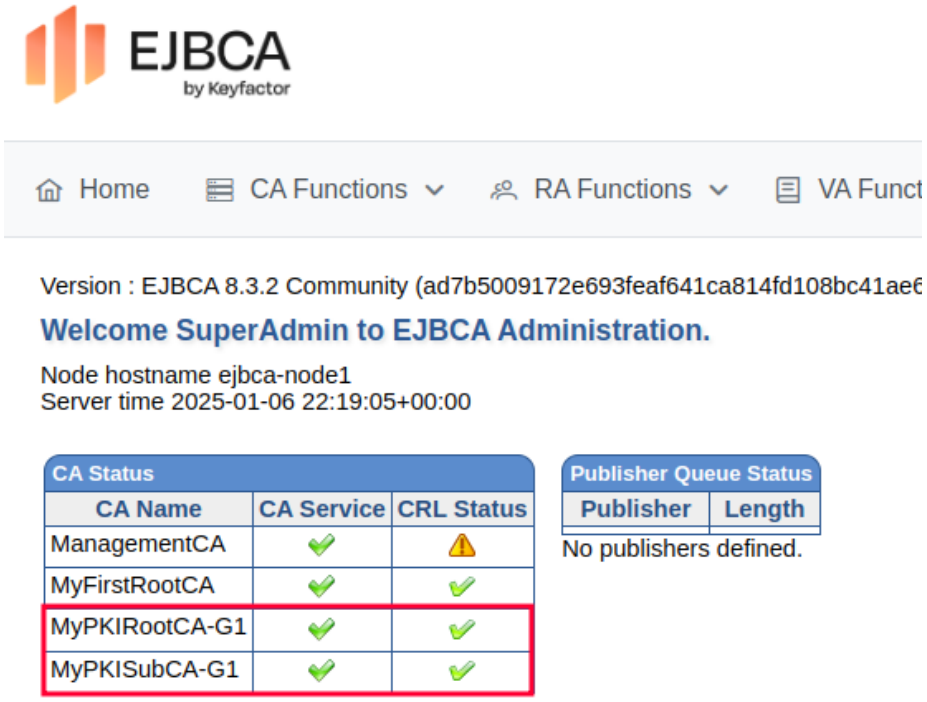

Figure 17: CA Overview

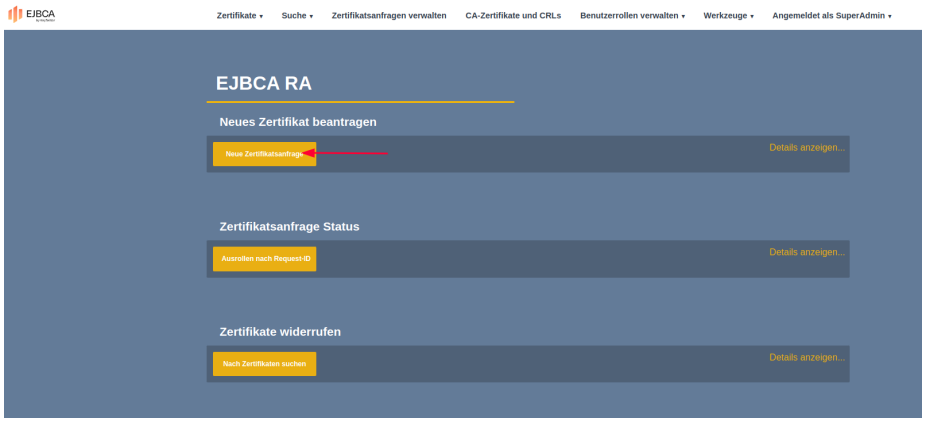

Figure 18: Cert Request

```
[ req ]
default_md = sha256
prompt = no
distinguished_name = dn

[ dn ]
CN = test.my.pki
O = FH Campus
C = AT
```

Anschließend werden mit OpenSSL die server keys erstellt.

```
openssl ecparam -genkey -name prime256v1 -out tls_server.key
```

Aus diesen beiden Teilen wird dann der CSR mit OpenSSL generiert:

```
openssl req -new -key tls_server.key -config tls_cert_req.cnf

# Example Output
-----BEGIN CERTIFICATE REQUEST-----
MIHyMIGZAgEAMDcxFDASBgNVBAMMC3Rlc3QubXkucGtpMRIwEAYDVQQKDAlGSCBD
YW1wdXMxCzAJBgNVBAYTAkFUMFkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDQgAE6SAZ
4WEeZGUbpJ2HSyXKh3BpySV7c8lTHuB4+RvAYYwsaH9YDL8cW8yOLBn7CZyIsObL
AEtCPt0JmadOdLvTt6AAMAoGCCqGSM49BAMCA0gAMEUCIQDF7k4DD+0qIih5ludM
+NwE5gOZPfQAYnQPc4H47rj+dQIgb4H9a1mquD6vD7qBD/xvkQI+FM7cUfWsiVYk
4+ChFuQ=
-----END CERTIFICATE REQUEST-----
```

Dieser CSR kann nurn im RA Web hochgeladen werden:

Anschließend kann die Full Certificate Chain als .pem Datei heruntergeladen werden:

Folgend wurde das Zertifikat via OCSP gegen die CA auf seine Gültigkeit geprüft:

```
openssl ocsp -issuer ~/Downloads/MyPKISubCA.cacert.pem -cert ~/Downloads/test.my.pki.pem -te

# Example Response

OCSP Request Data:
    Version: 1 (0x0)
    Requestor List:
        Certificate ID:
          Hash Algorithm: sha1
          Issuer Name Hash: 3205BE097AA74C9F3536A62D315E96AC8C797C0D
          Issuer Key Hash: 34F489757A956059F32983D876BF4041C1CA2C4B
          Serial Number: 6C989E14709615F0AAC926B1302E9418910051AE
    Request Extensions:
        OCSP Nonce:
            0410CD0CCDE8522FEBF562592F08A1AD0D3F
```

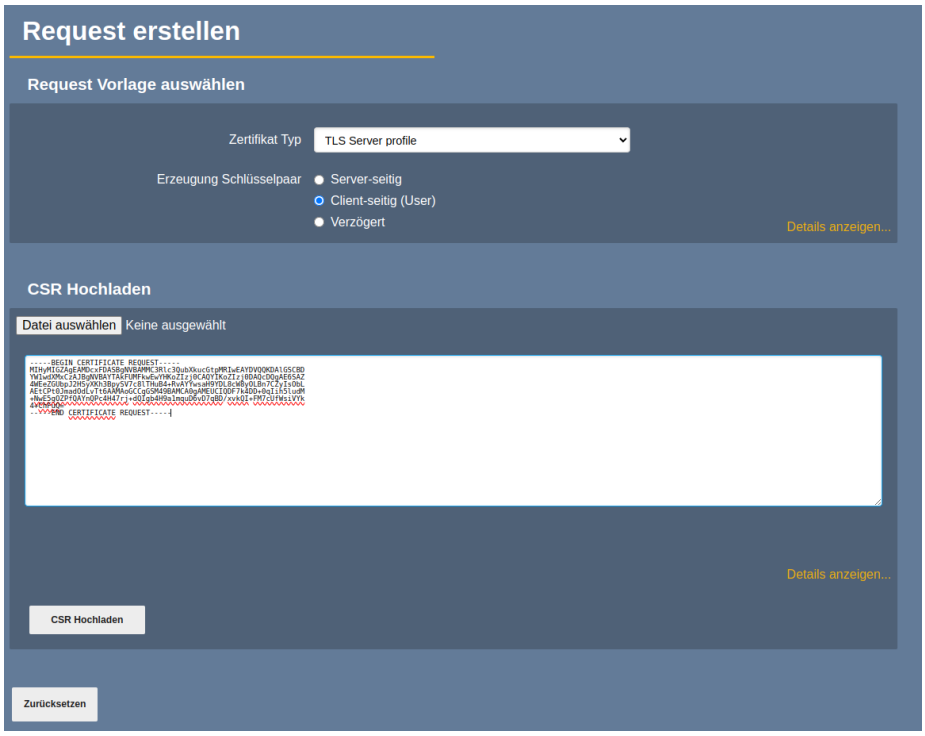

Figure 19: CSR Upload

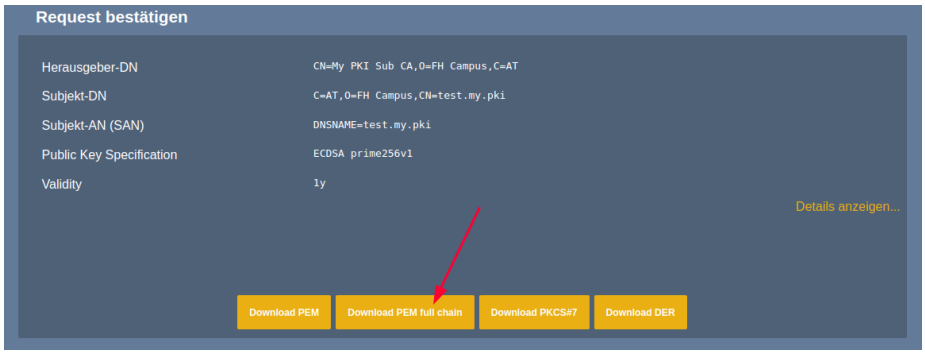

Figure 20: Full Chain

```
OCSP Response Data:
[...]
Response verify OK
test.my.pki.pem: good
        This Update: Jan  8 15:55:38 2025 GMT
```

2.4.2 Widerrufen von TLS Zertifikaten Um ein Zertifikat zu 'revoken' kann dieses in der RA Web gesucht und einfach durch den Button `Widerrufen` unter Angabe eines Grundes widerrufen werden.

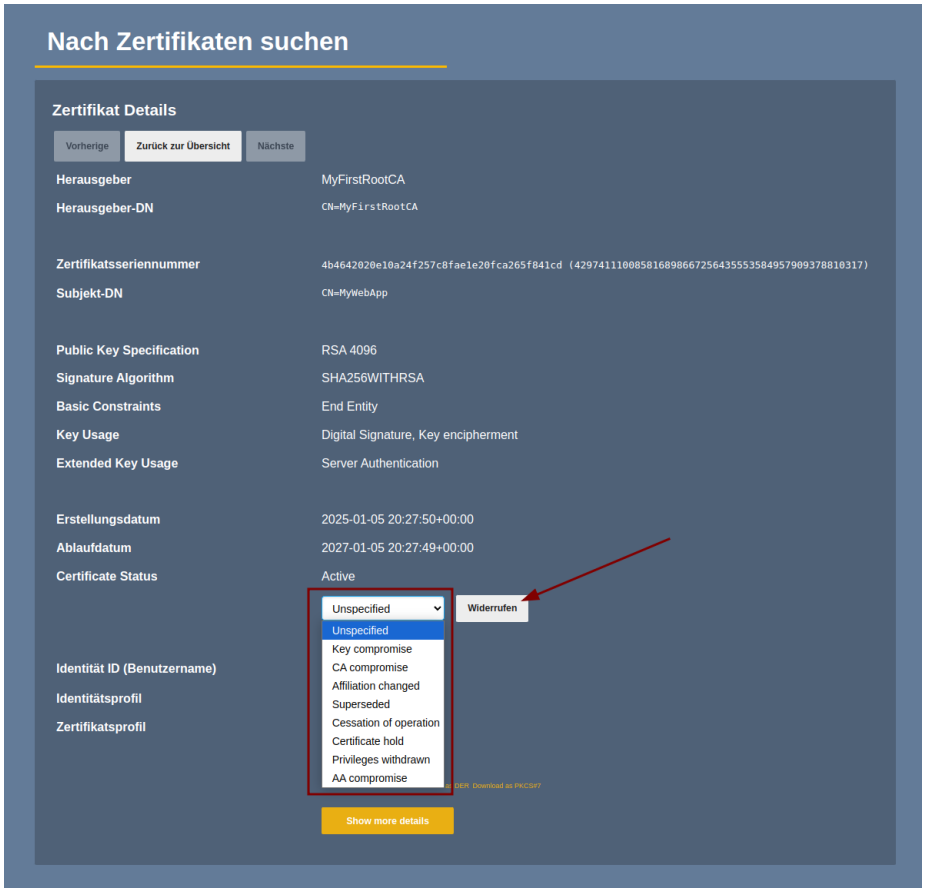

Figure 21: Revocation

Eine Prüfung der Gültigkeit via OCSP liefert nach dem Widerruf folgende Response:

```
OCSP Request Data:
    Version: 1 (0x0)
```

```
    Requestor List:
        Certificate ID:
          Hash Algorithm: sha1
          Issuer Name Hash: B71FF4FA62D45B811B06E47C1EB4A643D765E332
          Issuer Key Hash: 585E3CBC5A26BFEEBB0C994C22D555EDF663EBE8
          Serial Number: 4B4642020E10A24F257C8FAE1E20FCA265F841CD
    Request Extensions:
        OCSP Nonce:
           041022324CB4BB8D6184169F96028BEA7955
OCSP Response Data:

[...]

MyWebApp.pem: revoked
    This Update: Jan  5 21:11:02 2025 GMT
    Reason: unspecified
    Revocation Time: Jan  5 21:09:13 2025 GMT
```