Masterarbeit

Authentifizierte TLS-Kommunikation durch Verwendung einer Public-Key-Infrastruktur mit X.509 Zertifikaten im Embedded Systems Bereich

Bastian Dünhofen, BSc

Institut für Technische Informatik
Technische Universität Graz
Vorstand: Univ.-Prof. Dipl.-Inform. Dr.sc.ETH. Kay Uwe Römer

Begutachter: Dipl.-Ing. Dr. techn. Christian Kreiner
Betreuer: Dipl.-Ing. Dr. techn. Christopher Preschern

Graz, im August 2014

Kurzfassung

Sicherheitsexperten warnen bereits seit vielen Jahren vor unsicheren Zugängen oder unverschlüsselt übertragenen Daten sowie vor allgemeinen Sicherheitsrisiken besonders im Bereich von embedded Systems. Gezielte Angriffe auf Cyber Physical Systeme (CPS) - in welchen beispielsweise physikalische Anlagen zur elektrischen Energiegewinnung hauptsächlich von Software gesteuert und kontrolliert werden - wären als Ganzes kaum abschätzbar.
Diese Arbeit analysiert vorhandene Sicherheitsstandars für embedded Systems in Anlagen zur elektrischen Energiegewinnung und Verteilung. Neben der Einrichtung einer standardkonformen TLS-Verbindung zur Absicherung der übertragenen Daten, werden frei zur Verfügung stehende PKI-Implementierungen evaluiert. Mit Hilfe einer Anforderungsanalyse und Vorgaben aus den Standards werden praxisnahe Use-Cases erstellt, welche mit der für dieses Projekt am besten geeigneten PKI-Implementierungen realisiert werden. Dabei werden unter anderem Konzepte zur Verteilung der X.509 Zertifikate auf die einzelnen Devices oder automatische Benachrichtigungen vor dem Ablauf der Gültigkeit der Zertifikate analysiert und implementiert. Abschließend werden die Vorgaben aus dem Standard besprochen, sowie Anforderungen an dem Betrieb einer PKI und die Umsetzbarkeit der Use-Cases diskutiert.

Abstract

The confidentiality of transmitted data in computer networks is a hotly debated topic in the general public nowadays. Especially in the area of embedded systems, security experts are warning of unsecure accesses, unencrypted submitted data as well as of general security risks. The consequences of a cyber-attack on Cyber Physical Systems (CPS), which are mainly controlled from Software in physical plants for electrical energy generation, can produce damages which are hardly assessable. They could definitely endanger human life.

This paper analyses existing safety standards for embedded systems in facilities used for electrical energy generation and distribution. Beside the establishment of a TLS-secured connection for the protection of the free transmitted data, free available PKI implementations will be evaluated. Use Cases will be created in a practical manner, by using requirement analysis and inputs from standards. The Use Cases will be realized with the best suitable PKI-implementation for this project. This includes the analysis and implementation of concepts for the distribution of X.509 certificates on individual devices or automatic notifications before the validity of the certificates expire. Finally, the guidelines of the standards as well as the demands on the operation of the PKI and the feasibility of the Use Cases will be discussed.

EIDESSTATTLICHE ERKLÄRUNG

Ich erkläre an Eides statt, dass ich die vorliegende Arbeit selbstständig verfasst, andere als die angegebenen Quellen/Hilfsmittel nicht benutzt, und die den benutzten Quellen wörtlich und inhaltlich entnommenen Stellen als solche kenntlich gemacht habe.

Graz,am ....

(Unterschrift)

Danksagung

Diese Diplomarbeit wurde am Institut für Technische Informatik an der Technischen Universität Graz durchgeführt und somit gilt allen an diesem Projekt beteiligten Personen für ihre Unterstützung mein Dank. Besonderen Dank dabei gilt meinem Betreuer, Christopher Preschern für seine fachlich kompetenten Hilfestellungen und der sehr guten persönlichen Begleitung über den gesamten Projektverlauf hinweg.

An dieser Stelle möchte ich auch besonders meiner Familie und allen Freunden für die mentale Unterstützung und den starken Rückhalt, für die social events die damit verbundenen Diskussionen und schlussendlich für die gesellschaftliche und fachliche Erweiterung meines Horizontes während des gesamten Studiums danken.

Hauptsächlich jedoch möchte ich mich bei meiner Freundin bedanken, sie hat einen maßgeblichen Anteil daran, dass diese Arbeit überhaupt in der jetzigen Form vorliegt. Danke Ina, für die präzise Durchsicht dieser Arbeit und die konstruktive Kritik daran, sowie für die notwendig Ablenkung meinerseits und der Betreuung unseres Sohnes.

Graz, November 2014 Bastian Dünhofen

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Kapitel 1

Einleitung

Die Vertraulichkeit der in Computer-Netzwerken übertragenen Daten ist seit der Veröffentlichung der Snowden Dokumente auch ein in der allgemeinen Öffentlichkeit stark diskutiertes Thema. Sicherheitsexperten warnen jedoch schon seit vielen Jahren vor unsicheren Zugängen oder unverschlüsselt übertragenen Daten sowie vor allgemeinen Sicherheitsrisiken besonders bei Anlagen im Automatisierungsbereich. Ein besonderer Forschungsschwerpunkt hinsichtlich der Security und Safety entstand dabei zuletzt auf dem Gebiet von sogenannten Cyber Physical Systems (CPS), in welchen beispielsweise physikalische Anlagen zur elektrischen Energiegewinnung hauptsächlich von Software gesteuert und kontrolliert werden. Die Folgen eines gezielten Cyber-Angriffes auf solche CPS wären als Ganzes kaum abschätzbar, und könnten durchaus auch menschliches Leben gefährden.

Diese Arbeit ist Teil eines Security und Safety Konzeptes für ein CPS. Dabei werden Sicherheitsanalysen durchgeführt und diverse Konzepte wie beispielsweise ein High Assurance Boot oder Trusted Computing Komponenten für embedded Systems realisiert. Nachdem in einem vorangegangene Projekt, eine per Transport Layer Security (TLS) abgesicherte Datenverbindung auf einem embedded System (Freescale i.MX28) eingerichtet und die allgemeinen bekannten Schwachstellen des Protokolls analysiert wurden [Bas14], ist der Fokus dieser Arbeit auf das Verwalten und Verteilen der für eine TLS-Verbindung benötigten X.509 Zertifikate gelegt. Im Wesentlichen gliedert sich diese Arbeit wie folgt:

- Vorstellung der verbreitetsten Public-Key-Infrastrukturen (PKI).
- Analyse der existierenden Sicherheitsstandards respektive Vorgaben für embedded Systems in Analgen zur elektrischen Energiegewinnung.
- Erhebung der aktuell existierenden und frei verfügbaren (Open Source) Implementationen zur Verwaltung von X.509-Zertifikaten.
- Wissenschaftliche Bewertung der PKI-Implementation Aufgrund einer zuvor erstellten Anforderungsanalyse.

- Integration der PKI in das Projekt unter Erstellung und anschließender Implementierung von Use-Cases.
- Auswertung der Ergebnisse

Kapitel 2

Related Work

Am Beginn dieses Kapitels werden zuerst einige allgemeine Grundlagen bezüglich der Absicherung von Kommunikation mittels Security-Protokollen und elektronischen Zertifikaten geklärt. In weiterer Folge werden die zurzeit eingesetzten Public-Key-Zertifikat Infrastrukturen vorgestellt, sowie die Vor- und Nachteile der einzelnen Systeme benannt.
Anschließend wird auf internationale Implementierungsvorgaben für sichere Datenkommunikation und Authentifizierung mit Zertifikaten im Bereich von Systemen zur elektrischen Energiegewinnung und Verteilung, eingegangen. Dabei ist der Fokus auf eine Client-Server Architektur, in TCP-IP basierenden Netzen, gelegt. So werden unter anderem die verpflichtenden und optionalen Parameter für eine Absicherung der Kommunikation mit TLS und X.509 Zertifikaten erarbeitet.

2.1 Grundlagen Zeritfikat Management

Organisationen, wie die IETF versuchen durch ihr Schaffen, die Kommunikation in Computernetzwerken sicherer zu gestalten. Dadurch entstanden in den letzten Jahren einige neue Protokolle, wie zum Beispiel Secure/Multipurpose Internet Mail Extensions (S/MIME). Andere, bereits länger bestehende Sicherheitsprotokolle, wie Secure Sockets Layer (SSL) bzw. Transport Layer Security (TLS) oder Internet Protocol Security (IPSec) werden laufend weiterentwickelt. Alle gemeinsam verfolgen das Ziel, Datenverbindungen in unsicheren Netzen zu schützen.

Um

- Datenintegrität
- Vertraulichkeit und
- Authentifizierung

zu gewährleisten setzten die genannten Security-Protokolle unter anderem asymmetrische Kryptographie ein [RT10] [DR08] [KS05]. Jeder Kommunikationspartner,

welcher an einer, auf diese Weise abgesicherten Verbindung teilnehmen will, benötigt zumindest ein asymmetrisches Schlüsselpaar. Der private oder geheime Schlüssel (pirvate key) verbleibt in der Obhut des Users. Der öffentliche Schlüssel (public key) wird mit zusätzlichen Informationen versehen, in einem Zertifikat gespeichert und publiziert. Neben technischen Daten enthalten die elektronischen Ausweise (= Zertifikat) des weiteren Angaben über den Inhaber, um seine Identität im besten Fall zweifelsfrei an den öffentlichen Schlüssel binden zu können.

Diese Zertifikate werden entweder von einer dritten Partei, der sogenannten Trusted Third Party ausgestellt. oder die Ausweise werden im Nachhinein beglaubigt. Dabei ist es notwendig die Angaben zur Identität des Schlüsselbesitzers zu überprüfen. Diese Aufgeben werden von einer Zertifizierungsstelle der Certification Authority (CA) übernommen. Im praktischen Einsatz erfolgt die Prüfung der Authentizität eines Zertifikates nur mit dem öffentlichen Schlüssel der Zertifizierungsstelle. Die Vertrauenswürdigkeit dieser Zertifizierungsstellen bildet dabei das Rückgrat der Security-Protokolle im Internet.

Die dabei zu verwaltende Anzahl an Schlüsseln, respektive Zertifikaten nimmt bereits in kleineren Institutionen oder Firmen schnell unüberschaubare Formen an. Durch den Einsatz einer Public-Key-Infrastruktur (PKI) wird ein sauberes und effizientes Management der Schlüssel und Zertifikate gewährleistet.

2.2 Vergleich von Public-Key-Zertifikaten

Das effiziente und sichere Verwalten der öffentlichen Schlüssel, sowie der dazugehörigen Zertifikate stellt IT-Verantwortliche seit der Einführung der Security-Protokolle vor neue Aufgaben. Im weitesten Sinn ist jede Veröffentlichung des public Keys und der damit verbundenen Identität bereits als PKI zu werten. Dies wurde erstmalig von Whitfield Diffie und Martin Hellman 1976 publiziert.

> *„The enciphering key E can be made public by placing it in a public directory along with the user's name and address. Anyone can then encrypt messages and send them to the user, but no one else can decipher messages intended for him. Public key cryptosystems can thus be regarded as multiple access ciphers.“* [DH76]

Dieser Ansatz wurde zwei Jahre später erweitert. Um die Identität einer Person besser an die kryptographischen Schlüssel binden zu können wurden Zertifikate eingeführt [Koh78]. Diese Arbeit gilt allgemein als die Grundlage für PKI-Systeme. So wurde unter anderem der Begriff „trust a third party“ eingeführt [EFL+99].

Aktuell befinden sich zwei komplett unterschiedliche Ansätze in der praktischen Anwendung. Die Standards

- PGP
- PKIX

existieren seit 1988 parallel nebeneinander und wurden von den Usern für unterschiedliche Anwendungen gut angenommen. Zehn Jahre später wurde ein weiterer Standard publiziert, welcher die Vorteile der beiden alten Systeme miteinander verband. Obwohl dieser neue, Simple-PKI genannte Standard, einige Verbesserungen beinhaltet, kommt er jedoch bis heute in der Praxis kaum zur Anwendung. Nachstehend wird auf die unterschiedlichen Vertrauensmodelle aller drei Standards eingegangen.

2.2.1 PKIX, hierarchisches Modell

Der Begriff PKIX steht für:

- Public Key Infrastructure Exchange
- Public Key Infrastructure X.509[1]
- eine Arbeitsgruppe der IETF[2]

Allgemein sind diese Begriffe als Synonym zu betrachten, sodass mit PKIX eine hierarchische Zertifikatsinfrastruktur auf Basis von X.509 Zertifikaten gemeint ist.

Der X.509 Standard wurde erstmalig 1991 veröffentlicht und ist von der X.500-Standard-Serie, welche einen Verzeichnisdienst (directory service) beschreiben, abgeleitet. Dies hat unter anderem zur Folge, dass:

- öffentliche Schlüssel an globale Namen gebunden werden und
- Zertifikate im Verzeichnisdienst gespeichert werden

Seit dem Jahr 1995 hat die IETF eine Arbeitsgruppe eingesetzt, welche sich ausschließlich mit der Weiterentwicklung dieses Standards beschäftigt. Dabei wurden zum einen die Richtlinien für die X.509-Zertifikate immer wieder an die laufenden technischen Veränderungen angepasst [CSF+08]. Und zum anderen wurden dem System auch neue Funktionalitäten hinzugefügt, so dass über die Jahre ein paar Dutzend RFCs veröffentlicht wurden.[3]

[1] https://www.bsi.bund.de/DE/Themen/Cyber-Sicherheit/Glossar/cs_Glossar_P.html (Aufgerufen: Jänner 2014)

[2,3] http://datatracker.ietf.org/wg/pkix/ (Aufgerufen: Jänner 2014)

Das hierarchische Vertrauensmodell

Abbildung 2.1 zeigt eine vereinfachte schematische Darstellung des hierarchischen Vertrauensmodelles eines PKI-Systemes mit X.509 Zertifikaten. An der Spitze einer solchen Anordnung steht in jedem Fall ein sogenannte Root-CA, welche im einfachsten Fall selbst bereits Zertifikate für Personen oder zum Beispiel Webserver, den sogenannten Entitäten, ausstellen kann. Aus technischen Gründen und auch aus sicherheitsrelevanten Aspekten werden jedoch eine oder mehrere Sub-CAs in die Struktur integriert. Root-CAs besitzen selber ebenfalls eigene Zertifikate und werden von diversen kommerziellen Firmen, aber auch noch nicht kommerziellen Organisationen betrieben. Je nach Marktmacht bzw. finanziellen Potential dieser Firmen oder Organisationen werden die Zertifikate der Root-CAs, sowie der Sub-CAs in den diversen Clients, wie zum Beispiel den Web-Browsern vorinstalliert.
Beim Aufruf einer Webseite ist es dann für den Client möglich, über diese Vertrauenskette das Zertifikat des Webservers und somit auch dessen Identität zu prüfen. Dabei werden, die im Serverzertifikat hinterlegten Sub-CAs in aufsteigender Reihenfolge einzeln abgearbeitet. Am Ende dieser Vertrauenskette wird das Zertifikat der Root-CA überprüft.

Sub-CAs, respektive deren Zertifikate, werden auch an Dritte weitergegeben. Diese können dann in einer Firma wiederum als interne Root-CA verwendet, und um Sub-CAs erweitert werden. So ist es möglich die Struktur einer Firma in einer Zertifikatshierarchie abzubilden.

Bei diesem Modell vertrauen letztendlich alle Entitäten einer einzigen Organisation. Nämlich dem Unternehmen, welches die Root-CA betreibt. Diese Zertifizierungsstelle muss darauf achten, dass sie die Zertifikatswerber eindeutig identifiziert und bei Missbrauch die betreffenden Zertifikate widerruft und dies auch publiziert. Als Erweiterung dazu gibt es die Möglichkeit der Cross-Zertifizierung von Root- und Sub-CAs.

Ein großes Anwendungsgebiet dieser Art von Zertifikatsinfrastrukturen ist das SSL/TLS Protokoll, über welches zum Beispiel E-Mailkonten (Internet Message Access Protocol Secure (IMAPS) und Simple Mail Transfer Protocol Secure (SMTPS)), Webmail-Konten, E-Banking-Accounts oder Online-Shops, letztere mit Hypertext Transfer Protocol Secure (HTTPS), abgesichert werden. E-Government-Systeme, wie die Bürgercard, E-Signaturen, der elektronische Steuerausgleich usw. wären ohne entsprechende PKIX im Hintergrund, welche Zertifikate für neue TeilnehmerInnen ausstellt, oder die Gültigkeit von bestehenden Zertifikaten prüft, nicht aufrecht zu erhalten.

[4] http://www.pki.fraunhofer.de/cmsExtern/de/pkis-und-zertifizierungsstellen.html (Aufgerufen: Jänner 2014)

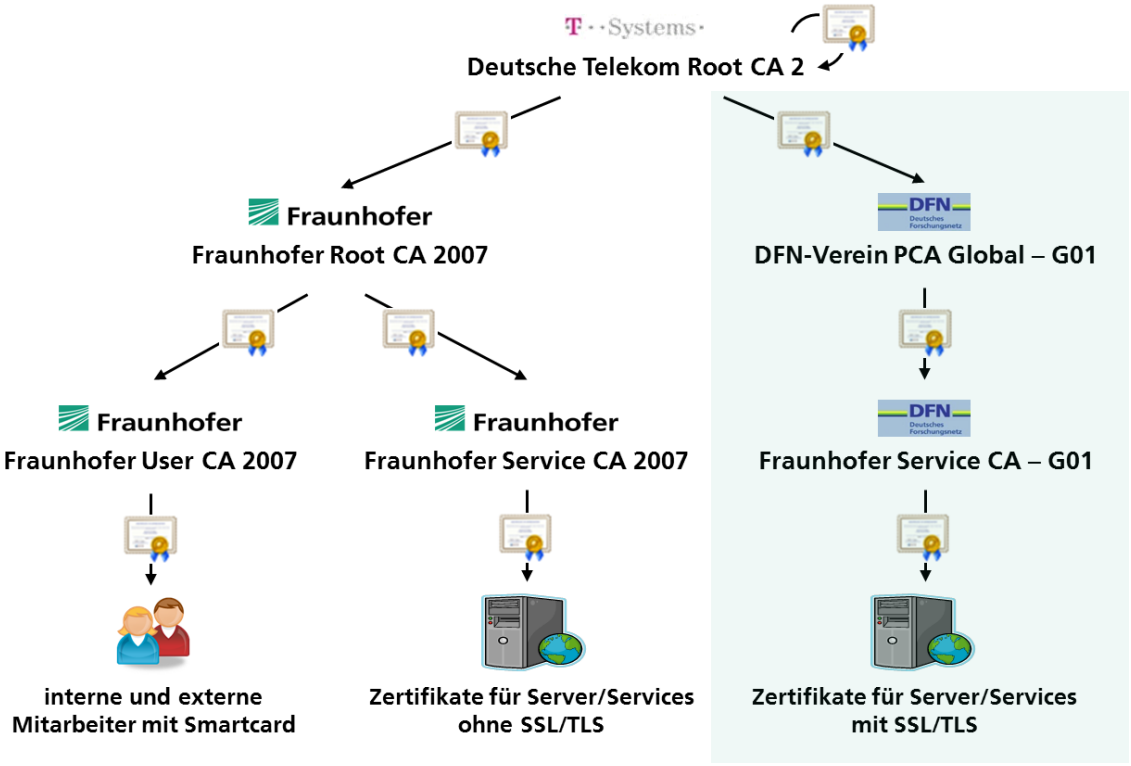

Abbildung 2.1: Zertifikatshierarchie X.509 - Quelle: Frauenhofer[4]

Das Ausstellen von X.509 Zertifikaten hat sich in den letzten Jahren zu einem sehr erfolgreichen Geschäftsmodell entwickelt, sodass für ein einzelnes fremd signiertes ein Jahr lang gültiges SSL-Zertifikat, je nach Funktionsumfang, mehr als 1000 EUR bezahlt werden[5]. Alternativ dazu werden vom gemeinnützigen Verein der CAcert[6] auch kostenlos Zertifikate zur Verfügung gestellt. Der Nachteil jedoch ist, dass das Root-Zertifikat der CAcert bei den meisten Clients nicht als vertrauenswürdig eingestuft ist.

2.2.2 PGP, Web of Trust

Neben den zumeist kommerziell betriebenen PKIX-Systemen ist alternativ dazu die von Pretty Good Privacy (PGP) geschaffe Zertifikatsinfrastruktur weit verbreitet. PGP bezeichnet dabei ein bestimmtes Programm bzw. die Derivate GnuPG und OpenPGP, sowie die von diesen Programmen geschaffenen Standards für das Format verschlüsselter Nachrichten und der entsprechenden Schlüssel respektive der Zertifikate.

[5]http://www.symantec.com/de/de/page.jsp?id=compare-ssl-certificates
[6]http://www.cacert.org/

PGP wurde 1991 von Phil Zimmermann veröffentlicht, erst nachträglich wurden die von PGP benutzten Protokolle für Zertifikat-, Nachrichten-, Schlüsselaustausch und dergleichen niedergeschrieben und somit standardisiert [ASZ96]. Mit der Entwicklung von OpenPGP und mit den bisher gesammelten Erfahrungen entstand 1998 ein weiterer Standard [CDFT98]. In ihm flossen ebenfalls Erweiterungen auf Basis des X.509v3-Standards ein. Das neue Singnaturformat ermöglicht es seitdem, mit der Hilfe von Sub-Paketen, zusätzliche Informationen zu der betreffenden Unterschrift zu speichern. Angelehnt zum X.509v3-Format können somit unter anderem folgende Informationen mit abgespeichert werden:

- Gültigkeitsdauer (Beginn und Ende) eines Schlüssels
- bevorzugte Algorithmen für die Erstellung des Hash
- zu verwendender symmetrischer Verschlüsselungsalgorithmus
- Angaben zum Keyserver
- Widerrufsinformationen

Mit diesen zusätzlichen Informationen zum Schlüsselinhaber wurden die Grundlagen zum Web of Trust (WOT) gelegt. Beide Standards [ASZ96] und [CDFT98] wurden daraufhin in [CDF+07] zusammengeführt.

Das Web of Trust

Das Web of Trust ist im Gegensatz zu den auf X.509 basierenden hierarchischen PKI-Systemen ein dezentraler Ansatz, um die Authentizität eines public Keys zu bestätigen. Dabei werden zwei unterschiedliche Vertrauens- bzw. nicht Vertrauensmodelle unterschieden:

- Es wird direkt darauf vertraut, dass ein Schlüssel, den man nicht selber signiert hat gültig ist und somit einer bestimmten Person zugeordnet werden kann.
- Es wird indirekt auf einen dritten Schlüssel vertraut, indem man der Person Glauben schenkt, welche den dritten Schlüssel ebenfalls vertraut.

Das indirekte Vertrauensmodell wird auch „Owner Trust“ genannt und kennt fünf Vertrauensstufen, welche von „der Benutzer ist unbekannt“, bis hin zum absoluten Vertrauen in einen Benutzer, respektive seiner Schlüssel, reichen. Über eine festgelegte Vorschrift wird dann das Vertrauen in einen öffentlichen Schlüssel, der sogenannte „Key-Legitimacy“-Wert, errechnet.

Abbildung 2.2 zeigt schematisch das Web of Trust und seinen dezentralen Ansatz. Verglichen mit dem hierarchischen Modell aus Abbildung 2.1 kann in diesem Fall jeder User, welcher über ein Zertifikat verfügt, andere Zertifikate bestätigen und

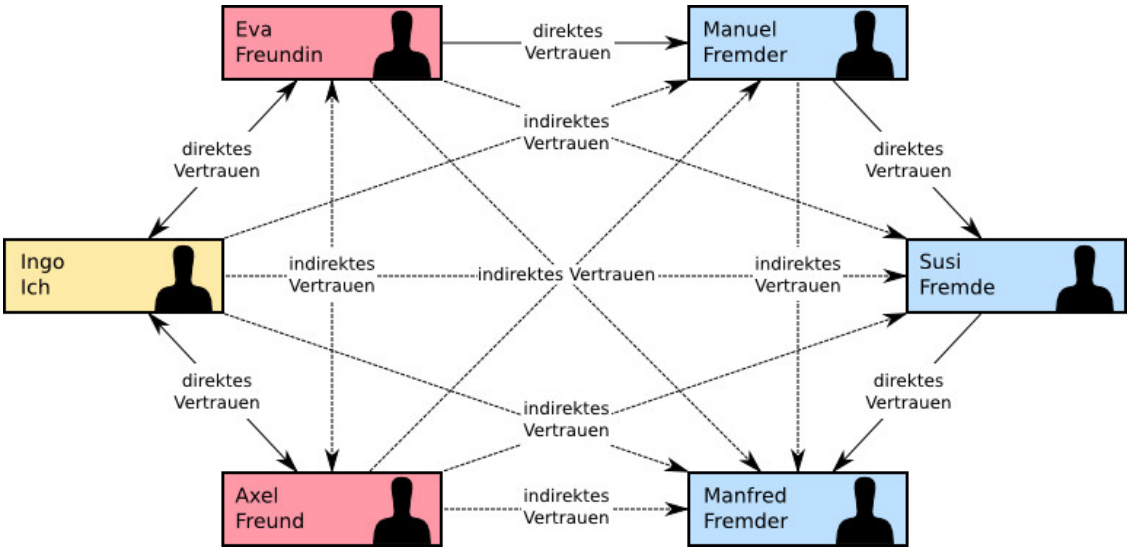

Abbildung 2.2: Web of Trust - Quelle: Wikipedia[7]

ähnelt somit einer CA aus dem hierarchischen Modell. Inwieweit ein User einem anderen vertaut muss individuell entschieden und festgelgt werden.

Starke Verbreitung findet dieses System beim Signieren oder Verschlüsseln von E-Mails und deren Anhängen. Dabei wird die dafür notwendige Funktionalität meist über Plugins dem Mail-Client hinzugefügt. In der Open Source Szene wird GnuPG unter anderem auch zum Signieren von Software oder Softwarepaketen bei den diversen Linux-Distributionen eingesetzt.

Das System und auch die zum Betrieb notwendigen Tools sind erhältlich ohne, dass dafür Lizenzkosten zu entrichten sind und steht den Usern somit kostenlos zur Verfügung.

2.2.3 SPKI

Für dieses System werden in der Literatur unter anderem folgende Begriffe verwendet:

- Simple Public Key Infrastructur (SPKI)
- Simple Distributed Security Infrastructure (SDSI)

Dabei ist SDSI das ältere System und SPKI die Weiterentwicklung davon. Derzeit gebräuchlich sind auch noch Kombinationen aus beiden Bezeichnungen wie SPKI/SDSI vgl. dazu [CEE+01].

[7] http://commons.wikimedia.org/wiki/File:Web_of_Trust_2.svg (Aufgerufen: Jänner 2014)

Wie es aus dem Namen Simple-PKI bereits zu entnehmen ist, ist dieses Projekt angetreten, um hauptsächlich die bekannten Probleme wie:

- Komplexität und Kostenintensität
- Skalierbarkeit
- Global Names

der auf den traditionellen X.509-Zertifikaten basierenden PKIs zu beheben [CEE+01]. Die dazu von der IETF 1998 eingesetzte Working Group[8] hat zwei Standards [EFL+99] und [Ell99] diesbezüglich publiziert, jedoch sind beide nach wie vor als experimentell gekennzeichnet.

Obwohl wissenschaftliche Arbeiten bereits anhand von DNS [Ter00] und HTTP [RSC01] in der Vergangenheit zeigten, dass das Simple-PKI-System mit etablierten Protokollen und Anwendungen verwendet werden kann, fand dieses System keine Akzeptanz in der praktischen Anwendung. Aktuell ist das SPKI-System immer noch Gegenstand von Forschungsprojekten hauptsächlich dort wo eine einfache Umsetzung einer PKI gefragt ist [SYSkB10].

Globale Zertifizierungsstellen, vergleichbar mit denen aus dem PKIX-System, sind in diesem Modell nicht notwendig, somit gibt es, anders als bei PKIX, keine Möglichkeiten, diese Systeme kommerziell zu nutzen.

2.3 PKI und TLS im embedded Bereich

Embedded Systeme, welche in industriellen Anlagen eingesetzt werden, müssen zumeist speziellen Anforderungen entsprechen. Diese Regeln und Normen werden in den diversen nationalen und internationalen Standards definiert. Je nach Anwendungsgebiet gelten unterschiedliche und oft mehrere Betriebs- und Sicherheitsvorschriften.

Dieses Kapitel beleuchtet die für die elektrische Energiegewinnung und Verteilung maßgeblichen Standards. Dabei ist der Fokus auf eine Client-Server Architektur in TCP/IP Netzen gelegt. Zu Beginn wird der in Europa weit verbreitete IEC 61850 Standard vorgestellt. Darauffolgend wird auf die für diese Arbeit relevanten Teile der Richtlinie IEC 62351, welche den IEC 61850 um Funktionalität hinsichtlich der Datensicherheit erweitert, eingegangen. Obwohl der Sicherheitsstandard IEC 62351 noch relativ neu und in Teilen noch unveröffentlicht ist, wird auf ihn in diversen

[8] https://datatracker.ietf.org/wg/spki/

wissenschaftlichen Arbeiten bereits verwiesen, zum Beispiel [FAAM11], [BS11]. Abschließend wird in diesem Kapitel auf alle notwendigen Vorgaben des neuen Regelwerkes IEC 62351 hinsichtlich der Absicherung einer Netzwerkverbindung mittels TLS unter der Verwendung von X.509 Zertifikaten eingegangen.

2.3.1 IEC 61850

Der IEC 61850 [IEC13a], welcher in seiner ersten Version zwischen 2003 und 2005 publiziert wurde, hat sich zu einem globalen Kommunikationsstandard im Automatisierungsbereich von elektrischen Umspannwerken entwickelt. Dabei wird er sowohl von Herstellern als auch von Kunden angenommen. Über die Jahre entwickelte sich der IEC 61850 Standard zu einem allgemeinen Referenzwerk der Automatisierungstechnik im Energiesektor. So beschreibt zum Beispeil IEC 61850-7-410, Kommunikationsmethoden für Kontrollmechanismen und Monitoring in Wasserkraftwerken. Aktuell liegt der Standard bereits teilweise überarbeitet, in seiner zweiten Version vor und ist auch unter der Bezeichnung DIN EN 61850 veröffentlicht worden.

Im Wesentlichen werden nachstehende Daten- respektive Kommunikationstypen unterschieden:

- Umsetzung der Manufacturing Messaging Specification (MMS) unter der Verwendung des TCP/IP oder des ISO Transport Service Protokolls als Basisübertragungsprotokolle für klassische Client-Server-Kommunikation. Definiert in IEC 61850-8-1
- Übertragung von zeitkritischen oder real-time Nachrichten, den sogenannten GOOSE-Nachrichten, spezifiziert in IEC 61850-8-1
- Sampled Values (SV) für die Rohdatenübertragung, z.B.: Abtastwerte von Messumwandler und dergleichen, spezifiziert in IEC 61850-9-1

Dabei werden die für die echtzeitfähige Kommunikation benötigten GOOSE und SV Pakete am Ethernet Link Layer Peer-to-Peer übertragen. Abbildung 2.3 veranschaulicht das objektorientierte Datenmodell des IEC 61850 Standards. Sicherheitsfunktionen, wie zum Beispiel das Verschlüsseln der zu übertragenden Daten oder das Authentifizieren der Kommunikationspartner ist im IEC 61850 nicht vorgesehen. Diesbezüglich wurde von der IEC ein weiterer Standard (62351) veröffentlicht. Nachfolgende Ausführungen richten den Fokus auf eine sichere Datenübertragung, basierend auf TCP/IP Netze nach IEC 61850 und 62351, d.h. die Absicherung der echtzeitfähigen Kommunikation wird in dieser Arbeit nicht behandelt.

2.3.2 IEC 62351

Der IEC 62351-Standard [IEC13b] befasst sich mit der sicheren Datenübertragung bzw. mit der Informationssicherheit in Anlagen zur elektrischen Energiegewinnung

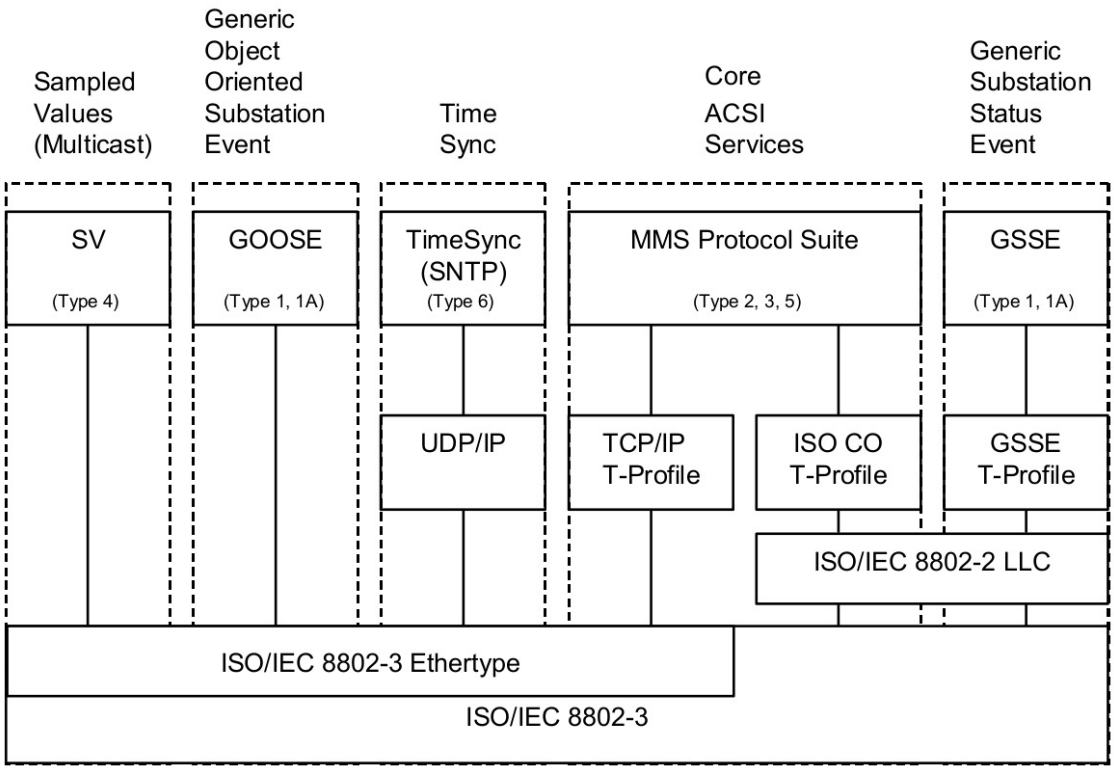

Abbildung 2.3: Datenmodelle und Kommunikationsprofile nach IEC 61850 - Quelle: [IEC13b]

und -verteilung. Dabei werden unter anderem die im IEC-61850 [IEC13a] definierten Datenmodelle um Sicherheitsfunktionen unter der Einhaltung der eingeführten Randbedienungen, wie zum Beispiel der Antwortzeiten, von den Datenpaketen erweitert. Voraussichtlich wird dieser Sicherheitsstandard in 11 Teile gegliedert sein. Eine Übersicht der einzelenen Teile bietet Tabelle 2.1.

Project Reference	Title	Current Stage	Forecast Publication Date	Stability Date
62351-1 Ed. 1.0	Part 1: Communication network and system security - Introduction to security issues	PPUB	2006-12	2013
62351-2 Ed. 1.0	Part 2: Glossary of terms	PPUB	2008-09	2013
62351-3 Ed. 1.0	Part 3: Communication network and system security - Profiles including TCP/IP	PPUB	2007-06	2013
62351-4 Ed. 1.0	Part 4: Profiles including MMS	PPUB	2007-06	2013
62351-5 Ed. 2.0	Part 5: Security for IEC 60870-5 and derivatives	PPUB	2013-04	2014
62351-6 Ed. 1.0	Part 6: Security for IEC 61850	PPUB	2007-06	2013
62351-7 Ed. 1.0	Part 7: Network and system management (NSM) data object models	PPUB	2010-08	2013
62351-8 Ed. 1.0	Part 8: Role-based access control	PPUB	2011-10	2014
62351-8 f1 Ed. 1.0	Part 8 f1: Develop recommendations for a comprehensive security process for power industry operations environments	MERGED	2007-10	2014
62351-9 Ed. 1.0	Part 9: Cyber security key management for power system equipment	1CD	2013-03	
62351-10 Ed. 1.0	Part 10: Security architecture guidelines	PPUB	2012-10	2015
62351-11 Ed. 1.0	Part 11: Security for XML Files	ANW	2015-07	

Tabelle 2.1: Übersicht der einzelnen Teile im IEC 62351 - Basierend auf [IEC13b]

Anders als der IEC 61850 ist der von der TC 57 WG 15 entwickelte Sicherheitsstandard IEC 62351 derzeit noch nicht vollständig veröffentlicht. Teil neun, *„Part 9: Cyber security key management for power system equipment“*, welcher im Wesentlichen die für diese Arbeit wichtigen Grundlagen zu Thema PKI enthalten wird, liegt zum Zeitpunkt der Erstellung dieser Arbeit erst als Entwurf vor. Weitere für dieses Projekt wichtige Teile des Standards sind neben den allgemeinen Teilen 1 und 2 die Dokumente *„Part 3: Communication network and system security – Profiles including TCP/IP“* und *„Part 4: Profiles including MMS“*, auf welche nachfolgend eingegangen wird [IEC13b].

Ein wichtiger Aspekt bei allen neuen Standards ist seine Akzeptanz bei den Herstellern der Geräte, sowie bei den Abnehmern dieser. Dies kann nur erreicht werden, indem sich Geräte, welche den neuen Standard unterstützen, nahtlos in bestehende Anlagen integrieren lassen und somit rückwärts kompatibel sind. So muss es zum Beispiel möglich sein, dass neue Hardware, welche Verschlüsselung unterstützt, ebenso unverschlüsselt mit alter Technik kommunizieren kann [HBA10]. Um Kompatibilität zwischen den Herstellen zu gewährleisten wird die Verwendung von zwingend zu implementierenden kryptographischen Funktionen vorgeschrieben. Darüber hinaus bleibt es dann den Herstellern überlassen, zusätzliche und sicherere Funktionalität anzubieten. Übergangsfristen, wie zum Beispiel bis zu welchem Stichtag alle Geräte auf verschlüsselte Datenübertragung umgestellt werden müssen definiert der Standard nicht.
Ein weiterer Faktor in der Annahme eines neuen Standards ist die usability der Geräte, welche den neuen Anforderungen entsprechen. Erfahrungen aus der Vergangenheit zeigen, dass Sicherheit oft durch zu komplizierte Handhabung oder Wartung kompromittiert wird [HBA10]. Ist zum Beispiel die Funktion zum zurücksetzen des Standardpasswortes im Konfigurationsinterface schwer zugänglich, bleiben diese Passwörter meistens auf den Voreinstellungen. So wird auch diesbezüglich gefordert, die sicherheitsrelevanten Einstellungen, wie zum Beispiel das parallele Aktivieren von ver- und unverschlüsselten Protokollen möglichst einfach und transparent umzusetzen.

Somit lassen sich zumindest nachstehende nicht direkt sicherheitsrelevante Vorgaben zusammen fassen:

- Abwärtskompatibilität durch abschalten der Security Funktionalität (konfigurierbar)
- Kompatibilität von Geräten unterschiedlicher Hersteller durch zwingend vorgeschriebene minimale Funktionalität
- Usability durch einfache Konfigurationsmöglichkeiten

Hinsichtlich der sicherheitsrelevanten Vorgaben wird vom Standard zumeist die Verwendung von bereits etablierten Protokollen wie zum Beispiel Transport Layer Se-

curity (TLS) 1.0 vorgeschrieben.

Neben der Absicherung der in IEC 61850 definierten Datenmodelle werden im Sicherheitsstandard IEC 62351 darüber hinaus auch die Industriestandards IEC 60870-6, und 60870-5 um Sicherheitsfunktionen erweitert. Abbildung 2.4 zeigt die Abhängigkeiten zwischen den Standards.

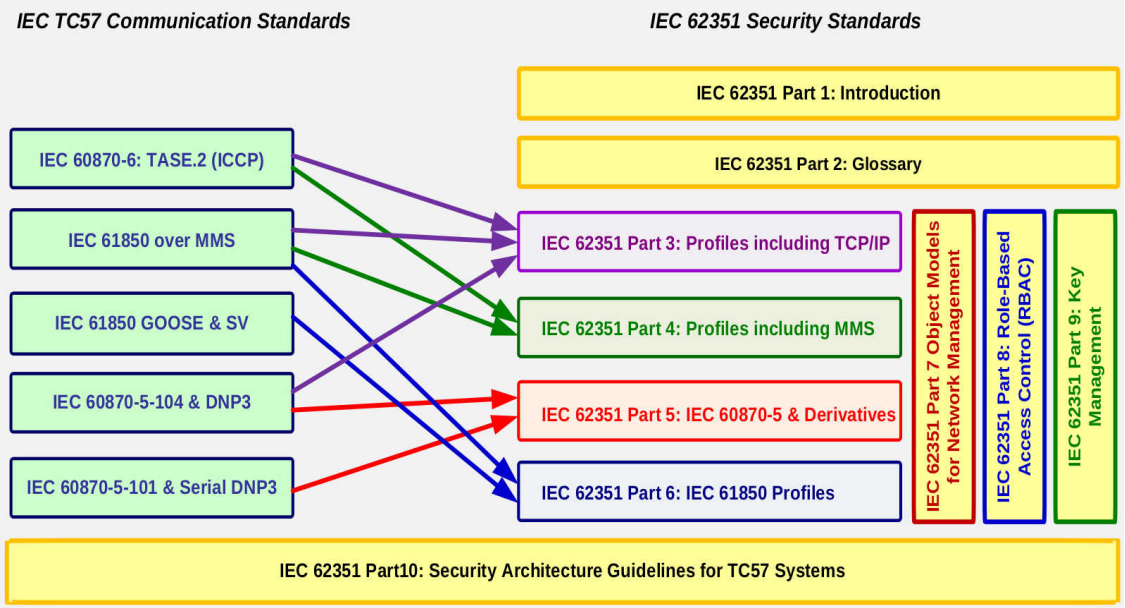

Abbildung 2.4: Beziehungen zwischen IEC 62351 und anderen Standards - Quelle: [IEC13b]

IEC 62351 Teil 3: Communication network and system security – Profiles including TCP/IP

Der dritte Teil des Sicherhietsstandards [IEC07a] enthält Vorgaben zu Vertraulichkeit, Manipulationserkennung (tamper detection) und message level authentication für Supervisory Control and Data Acquisition (SCADA)-Systeme auf TCP/IP Protokoll Ebene und ist somit für diese Arbeit von Relevanz. Dabei referenziert der Standard hinsichtlich der Transportsicherheit der Daten auf das mittlerweile veraltete TLS-Protokoll 1.0 [DA99]. Weitere normative und somit einzuhaltende Referenzen sind neben den ersten beiden Teilen (62351-1 und 62351-2) des Sicherheitsstandards IEC 62351 die nachstehend von der IETF publizierten offenen Normen:

- RFC 2712: Addition of Kerberos Cipher Suites to Transport Layer Security (TLS) [MH99]
- RFC 3268: Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS) [Cho02]

- RFC 3280: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile [HPFS02]

Der Geltungsbereich für den dritten Teil des Standards [IEC07a] ist auf die Absicherung der Datenübertragung auf der Transportschicht, d.h. auf den OSI-Schichten 4 und darunter, beschränkt. Die Absicherung höhern Ebenen, wie zum Beispiel der Applikationsschicht werden in anderen Teilen des IEC 62351 behandelt.

Der unautorisierte Zugang und das unautorisierte Verändern von Informationen, welche über das TCP/IP Protokoll übertragen werden, wird durch das verschlüsselte und authentifizierte Übertragen der Nachrichten auf der Transportschicht gewährleistet.

Als konkrete Bedrohungs- und Angriffsszenarien werden nachstehende Attacken benannt:

- Man-in-the-middle Attacken
- replay Attacken
- Abhören der übertragenen Daten (eavesdropping)

Dabei kann das Abhören der übertragenen Informationen natürlich nicht verhindert werden, jedoch ist es möglich durch das Verwenden starker Verschlüsselung diese Daten für lange Zeit dem Angreifer vorzuenthalten. Man-in-the-middle (MITM) Attacken und replay Attacken können in Kombination oder auch getrennt von einander erfolgen. Jedenfalls ist ihnen mit geeigneten Mitteln entgegenzuwirken, wie zum Beispiel durch das Verwenden von Message Authentication Code (MAC), Nonce oder Authentifizierung, wie sie zum Beispiel in TLS 1.2 RFC 5246 [DR08] beschrieben sind.

Anforderungen hinsichtlich des TLS-Protokolls:

Alle Versionen des TLS-Protokolls, wie es zum Beispiel in RFC 5246 [DR08] definiert ist, erlauben die Verwendung diverser unterschiedlicher Cipher Suites. Im Fall einer Implementierung von Endgeräten bleibt es nun den Herstellern überlassen, welche kryptographischen Funktionen zum Einsatz kommen. Dadurch ist es jetzt denkbar, dass Cipher Suites verschiedener Server und Clients untereinander nicht kompatibel sind. Die Folge wäre entweder, eine nicht zustande kommende Verbindung, oder der Aufbau einer unverschlüsselten und nicht authentifizierten Verbindung. Letzteres ist möglich, weil es per RFC 5246 erlaubt ist, dass TLS-Protokoll ohne die Sicherheitsfunktionen, wie zum Beispiel Verschlüsselung oder MAC zu betreiben.
Um die Interoperabilität der Dienste und Anwendungen zu gewährleisten wird im Standard zumindest eine fix zu implementierende Cipher Suite sowie ein fix definierter Satz von TLS Parametern spezifiziert. Weiters schließt der Standard die

Verwendung der *„NULL“* Cipher Suites, also jene Kombination von kryptographischen Funktionen aus, in welchen die Sicherheitsfunktionen entweder zum Teil oder komplett deaktiviert sind. So sind zumindest folgende Einstellungen zu eliminieren:

- TLS_NULL_WITH_NULL_NULL
- TLS_RSA_NULL_WITH_NULL_MD5
- TLS_RSA_NULL_WITH_NULL_SHA

Darüber hinaus bleibt es den Herstellern überlassen, welche zusätzliche kryptographische Funktionen unterstützt werden und welche nicht. Obwohl aus Gründen der Interoperabilität zumindest eine immer zu unterstützende Cipher Suite notwendig ist, wird diese im dritten Teil (62351-3) nicht benannt. Nähere Informationen dazu enthalten die Teile 4 und 5 (62351-4, 62351-5).

Beim TLS-Protokoll ist es vorgesehen, dass während einer Session, im Regelfall der symmetrische Schlüssel nicht wechselt. Diesbezüglich definiert der Sicherheitsstandard IEC 62351-3 eine sogenannte *„Cipher renegotiation“*, in der gefordert wird, dass nach einer konfigurierbaren zeitlichen Konstante oder nach einer einstellbaren Anzahl an gesendeten Paketen respektive Bytes ein neuer symmetrischer Schlüssel zwischen den Kommunikationspartnern ausgehandelt wird. Ist es nicht möglich den neuen Schlüssel nach einer weiteren einstellbaren Zeitkonstante (timeout) auszutauschen, muss die Verbindung abgebrochen werden.

Zum Schlüsselaustausch während der TLS-Handshake-Phase müssen folgende Methoden zwingend unterstützt werden:

- RSA
- DSS
- Diffe-Hellman-Merkle

Im Fall von RSA muss sowohl die Verschlüsselung als auch die Signatur Funktionalität implementiert sein. Somit ist es möglich, Schlüsselmaterial mit Hilfe des Diffe-Hellman-Merkle Protokolls sowohl mit RSA als auch nach dem Digital Signature Standard (DSS) signiert, auszutauschen. Diese Verfahren ist auch unter den Namen Perfect Forward Secrecy bekannt. Als minimal zu unterstützende Schlüssellänge werden 1024 Bit gefordert.

Anforderungen hinsichtlich Zertifikate und CAs

Bezüglich der Authentifizierung wird gefordert, dass sich beide Kommunikationsteilnehmer gegenseitig mittels X.509-Zertifikaten ausweisen können. Dabei wird empfohlen, dass die Zertifikate nicht größer als 8192 Bytes sein dürfen.

In Bezug auf die Unterstützung von CAs wird vom Standard gefordert, dass:

- die Geräte Zertifikate von unterschiedlichen CAs verwalten können müssen.
- dabei sowohl individuelle (RBAC), als auch jedes andere allgemein gültige Zertifikat einer autorisierten CA zu akzeptieren ist.

Wobei die Anzahl der zu verwaltenden Zertifikate, als auch die Akzeptanz der Authentifizierungsmethode, konfigurierbar sein muss.

Das Widerrufen respektive die Kontrolle über die ungültigen oder abgelaufenen Zertifikate ist nach den Vorgaben von RFC 3280 [HPFS02] zu implementieren. Das Überprüfen der CRL muss dabei in einen konfigurierbaren Intervall geschehen. Das Abfragen der CRL darf eine bestehende Verbindung nicht abbrechen. Ist die Widerrufsliste nicht verfügbar, darf dies ebenfalls eine bestehende Verbindung nicht unterbrechen.

Hinsichtlich der Ungültigkeit wird unterschieden in

- zeitlich abgelaufene und
- widerrufene

Zertifikate. Dies entspricht dem RFC 3280 [HPFS02]. Ein zeitlich abgelaufenes Zertifikat darf dabei zum einen eine aufrechte Verbindung nicht beenden, zum anderen ist es nicht gestattet, dass eine neue Verbindung mit abgelaufenen Zertifikaten aufgebaut werden kann. Bei einem widerrufenen Zertifikat ist darüber hinaus in jedem Fall auch eine bereits existierende Verbindung abzubrechen.

Aus Gründen der Abwärtskompatibilität fordert der Standard die Koexistenz von sowohl verschlüsselten als auch nicht verschlüsselten Verbindungen, sodass der per TLS abgesicherte Datentransport auf einen gesonderten TCP/IP Port abzuwickeln ist.

IEC 62351 Teil 4: Profiles including MMS

Zum einen werden im vierten Teil des Standards (62351-4) [IEC07b] die eher allgemein gehaltenen Vorgaben zur Verschlüsselung mit TLS näher ausformuliert. Zum anderen definiert die TC 57 WG 15 hinsichtlich der Informationssicherheit in MMS drei unterschiedliche Profile:

- Ein Applikationsprofil (A-Profile), welches Standards für Protokolle der OSI-Schichten 5-7 enthält und
- zwei Transportprofile (T-Profile), welche Standards für Protokolle der OSI-Schichten 1-4 definieren.

Abbildung 2.5 veranschaulicht den Aufbau der Profile im Detail. Im vierten Teil des Standards (62351-4) wird ausschließlich auf das A-Profile und auf das TCP T-Profile eingegangen. Das OSI T-Profile befindet sich somit außerhalb des Anwendungsbereiches. Unter *„A-Profile security“* werden Maßnahmen zur Absicherung auf Applikationsebene beschrieben, da dies jedoch nicht Teil dieser Arbeit ist, werden sie hier auch nicht weiter beschrieben. Hinsichtlich der *„T-Profile security“* ist es

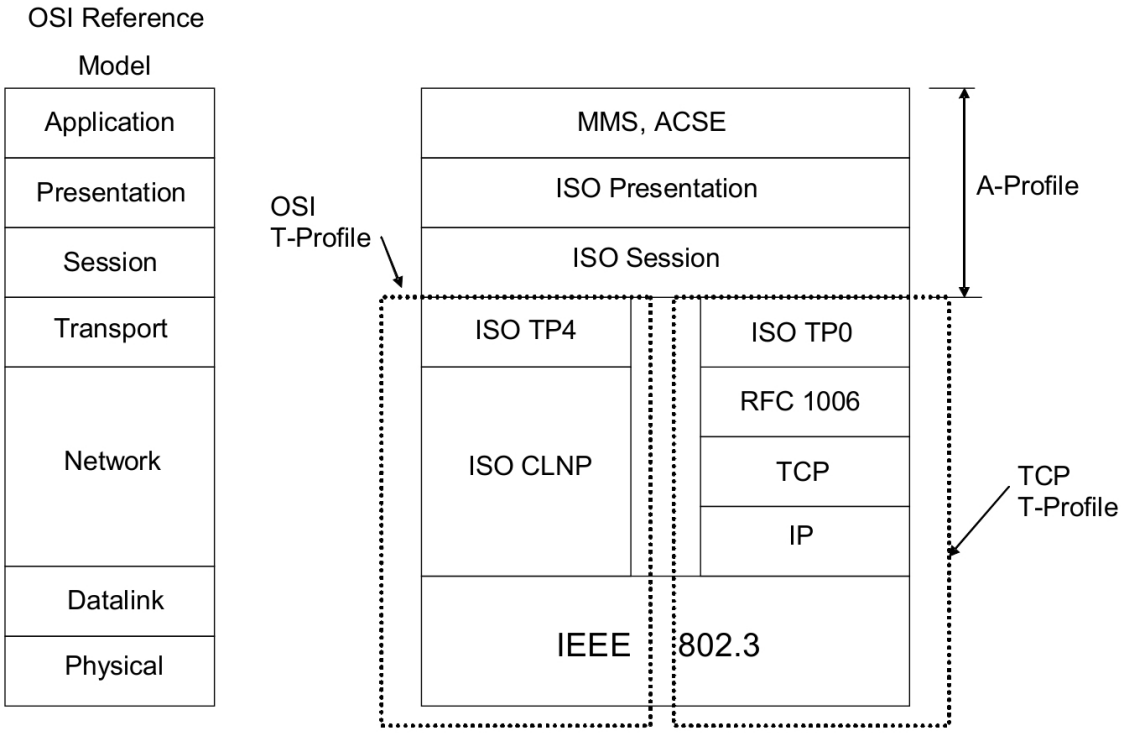

Abbildung 2.5: Profile hinsichtlich der Informationssicherheit - Quelle: [IEC07b]

nicht das Ziel des Standards neue Sicherheitsprotokolle zu publizieren. So wird beschrieben, wie vorhandene, bereits eingesetzte Protokolle mittels des TLS-Protokolls, abzusichern sind. Im Speziellen wird dabei auf die richtige Konfiguration des ISO on TCP/IP-Protokolls [RC87] eingegangen.

Bei der Verwendung von ISO on TCP/IP RFC 1006 müssen folgende Bedingungen erfüllt werden:

- strikte Einhaltung der vorgegebenen Größen für die TPDU
- das Feld Versionsnummer darf nicht ausgewertet werden
- das Längenfeld darf einen Wert von 2056 Oktett nicht überschreiten
- Implementierung der TCP-Keepalive Funktion, der Wert sollte mit einer Minute oder kleiner spezifiziert sein
- eine unverschlüsselte Verbindung hat über TCP Port 102 zu erfolgen

Die im dritten Teil (62351-3) geforderten Vorgaben hinsichtlich der Verwendung von TLS und Zertifikaten werden wie folgt erweitert:

- die TLS Funktionalität muss abschaltbar sein
- Geräte welche mehrere simultane Verbindungen zulassen, müssen in der Lage sein, sowohl verschlüsselte, wie auch nicht verschlüsselte Kommunikation abzuwickeln
- die verschlüsselte Kommunikation hat über TCP-Port 3782 zu erfolgen
- nach 5000 übertragenen ISO TPUs oder nach einer Verbindungsdauer länger als zehn Minuten, muss der synchrone Schlüssel neu ausverhandelt werden
- Zertifikate, welche größer 8192 als Byte sind, können unterstützt werden
- die Zeitspanne in welcher auf widerrufene Zertifikate geprüft wird muss konfigurierbar sein. Als Voreinstellung sind 12 Stunden zu wählen
- ein neuer Verbindungsaufbau mit widerrufenen Zertifikaten ist nicht gestattet
- die Cipher Suite *„TLS_DH_DSS_WITH_AES_256_SHA“* ist in jedem Fall zu unterstützen, weitere vorgeschlagenen, zu implementierende Cipher Suites befinden sich im Anhang A.1.
- es sind *„minimal-maximal“* [IEC07b] vier verschieden CAs zu unterstützen

Abbildung 2.6 zeigt den detaillierten Protokollstack für die parallele Verwendung von sowohl verschlüsselter als auch unverschlüsselten Verbindungen gemäß IEC 62351.

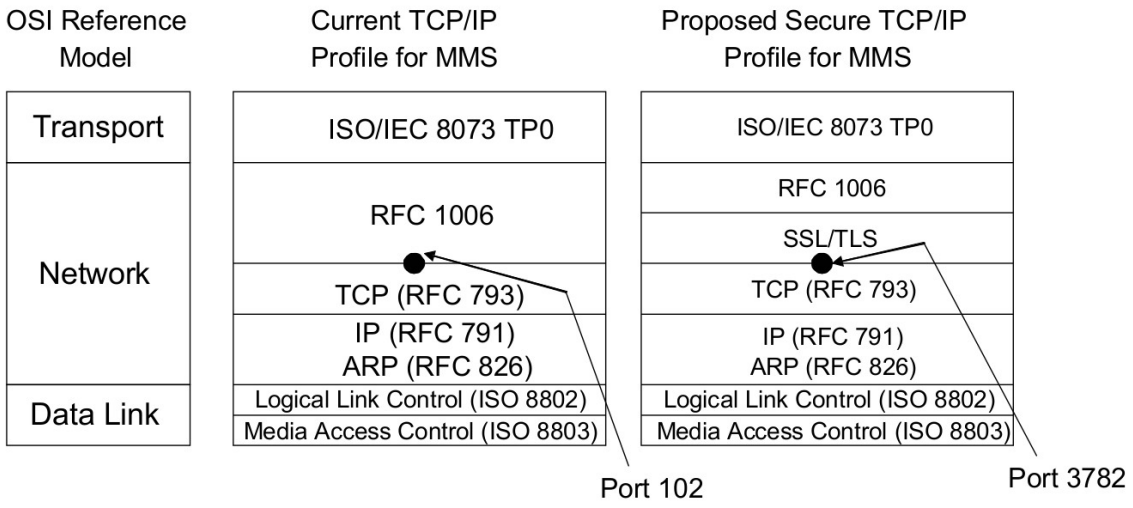

Abbildung 2.6: Bisheriges un- und neues abgesichertes TCP T-Profile nach IEC 62351 - Quelle: [IEC07b]

Kapitel 3

Architektur und Konzept

Das Ziel dieser Arbeit ist es, ein Konzept für eine praktische Umsetzung einer Public-Key-Infrastruktur für einen Industriebetrieb zu liefern. Dabei bestand die Aufgabe zuerst darin, X.509-Zertifikate für eine Client-Server-Verbindung im embedded Bereich, zur Absicherung des HTTP-Protokoll per TLS auszustellen und zu verwalten. Im Verlauf der Arbeit an diesem Projekt wurden die Anforderungen an diese firmeneigene PKI ständig erweitert, sodass derzeit davon ausgegangen werden muss, dass in einem möglichen Produktivbetrieb mehrere hundert Zertifikate zu verwalten sein werden.
Welche Anforderungen die Zertifizierungsstelle zu erfüllen hat und mit welchem System dies umgesetzt wird, soll dieses Kapitel klären.

Am Beginn wird die interne Struktur der PKI, sowie sie zur Zeit geplant ist, vorgestellt und mögliche Varianten des Root-Zertifikates diskutiert. Anschließend werden die wichtigsten Methoden zur Entscheidungsfindung, welche für dieses Projekt angewandt wurden, erläutert. In der darauf folgenden Anforderungsanalyse werden die wichtigsten zu erfüllenden Eigenschaften der PKI-Implementierung diskutiert.
Ein großer Teil dieses Kapitels befasst sich mit der Vorstellung respektive der Analyse der zur Zeit aktuell angebotenen Software zur Umsetzung einer Zertifizierungsstelle. Dieser Teil liefert im Wesentlichen die Grundlage für die anschließende Evaluierung der vollwertigen PKI-Implementierungen.
Abschließend wird in diesem Kapitel auf die Komponenten einer Zertifizierungsstellen am Beispiel der EJBCA eingegangen.

3.1 Strukturierung der Zertifizierungsstelle

Die Entscheidung eine hierarchische Public-Key-Infrastruktur auf Basis von X.509 Zertifikaten zu implementieren ist im Wesentlichen auf die nachstehenden beiden Faktoren zurückzuführen:

- Zum einen wird vom Sicherheitsstandard IEC 62351 [IEC13b] für Anlagen zur elektrischen Energiegewinnung und Verteilung gefordert, dass zur Absicherung der Datenübertragung das TLS-Protokoll verwendet werden soll. Dabei werden für die Authentifizierung der Entitäten und zum Austausch des Schlüsselmaterials X.509-Zertifikate eingesetzt.
- Zum anderen werden X.509-Zertifikate von weiteren Systemen verwendet, welche unmittelbar auf dieses Projekt aufsetzen. Dies sind beispielsweise ein High Assurance Boot (HAB) oder ein mit SSH abgesicherter Wartungszugang für Techniker mit Authentifizierung per X.509-Zertifikaten.

Abbildung 3.1 zeigt eine Übersicht der hierarchische Zertifikatsinfrastruktur wie sie in diesem Projekt zum Einsatz kommt. Dabei besteht diese Public Key Infrastruktur

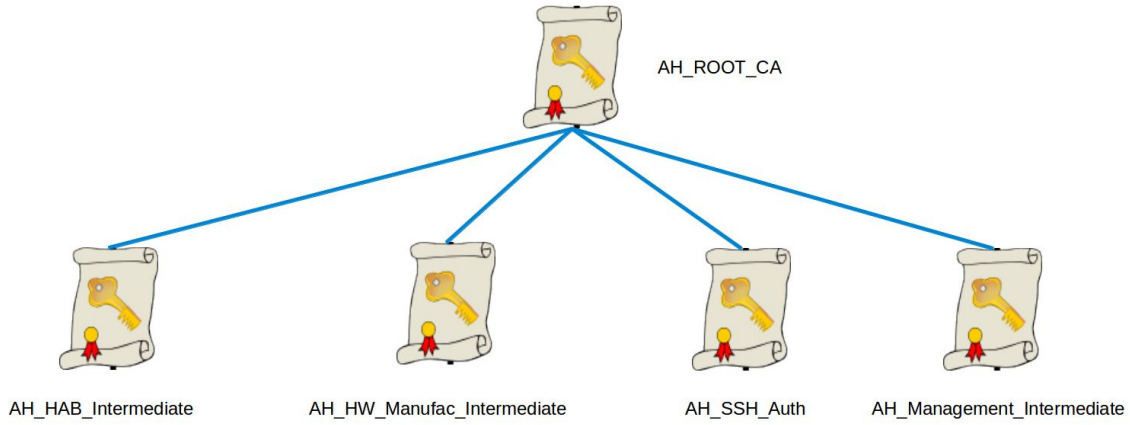

Abbildung 3.1: Strukturierung der Zertifizierungsstelle Übersicht

im Wesentlichen aus:

- einer Root-CA mit einem selbst signierten Zertifikat und
- mehrere, aus verwaltungs- sowie sicherheitstechnischen Gründen eingezogenen Sub-CAs.

Alle Sub-CAs entweder dabei von der Root-CA direkt oder von den ihnen übergeordneten Sub-CAs zertifiziert. Eine detailierte Aufstellung der Zuständigkeitsbereiche, sowie der Konfiguration der Sub-CAs, als auch der Root-CA erfolgt im Kapitel 4.

3.1.1 Konzept des Root-Zertifikates

Für die Erstellung des Zertifikates der Root-CA wurden die drei nachstehenden Möglichkeiten diskutiert:

- selbst signiertes Zertifikat
- fremd signiertes Zertifikat
- cross signiertes Zertifikat

Ein selbst signiertes Root Zertifikat wird von den meisten PKI-Implementierungen zum Zeitpunkt der Installation erstellt. Ein solches Zertifikat kann durchaus für den weiteren Betrieb der PKI verwendet werden. Der Nachteil bei selbst signierten Root-Zertifikaten ist, dass der Anwender sich zum einen selber von der Echtheit des Zertifikates überzeugen muss, also die Prüfung der Identität selbst vorzunehmen hat und zum anderen müssen diese Zertifikate manuell in die Software importiert werden.

Ein fremd signiertes Zertifikat, welches berechtigt ist darunter eine Public-Key-Infrastruktur zu errichten, ist vom technischen Standpunkt her betrachtet einem selbst signierten Zertifikat zu bevorzugen. Solche fremd signierte Zertifikate werden unter anderem von der Firma GlobalSign[1] angeboten. Eine Anfrage über die Kosten dieser Zertifikate wurde von GlobalSign jedoch nicht konkret beantwortet.

Eine Cross-Zertifizierung bietet im Vergleich mit einem fremd signierten Zertifikat ebenfalls die Möglichkeit, Zertifikate leichter über die Grenzen der eigenen PKI hinaus zu nutzen. Dabei stellen sich zwei gleichberechtigte PKI-Systeme ein sogenanntes Cross-Zertifikat aus. Diese Zertifikatsinfrastruktur ist somit eine Mischform aus selbst und fremd signierten Zertifikaten. Als Erweiterung dieser Variante ist es möglich, eine weitere übergeordnete sogenannte Bridge-CA zu verwenden. Ist diese Institution sowohl vertrauenswürdig als auch groß genug, bietet sie eine wirkliche Alternative zu den kommerziellen Zertifizierungsstellen.
Als Beispiel sei hier die European Bridge CA[2], betrieben vom TeleTrusT – Bundesverband IT-Sicherheit e.V., angeführt. Mitglieder dieses nicht Gewinn orientieren Vereins sind internationale Firmen wie Siemens oder Microsoft sowie Behörden, öffentliche Einrichtungen und Institutionen. Teilnehmen an der European Bridge CA kann prinzipiell jeder. Die Integration einer bereits aktiven PKI ist ebenso möglich wie die das Einbinden von Zertifikaten, welche von einem öffentlichen Trust Center ausgestellt wurden. Eine Beteiligung an den Kosten der European Bridge CA ist gestaffelt - an den eigenen ausgestellten Zertifikaten - und beginnt bei 500 EUR im Jahr für PKIs mit weniger als 300 Zertifikaten.

Da es sich im hier vorliegenden Projekt um eine firmeneigene PKI handelt, welche von Fachkräften betreut wird, kann davon ausgegangen werden, dass sich die User um die Installation des Root-Zertifikates im Client keine Gedanken machen

[1] https://www.globalsign.com/de-de/zertifizierungs-stelle-root-signation/ (Aufgerufen: Februar 2014)

[2] https://www.ebca.de (Aufgerufen: Februar 2014)

müssen oder, dass sie im Umgang mit der PKI und der Verwendung von Zertifikaten geschult werden. Zum anderen soll dieses Projekt eine mögliche Umsetzung einer PKI zeigen, sodass es zur Zeit nicht zweckmäßig ist, sich bereits jetzt dem aufwendigen Audit-Prozess eines externen Trust Centers zu unterziehen. Aus diesen Gründen wird für die Root-CA ein selbst signiertes Zertifikat verwendet. Sollte dieses Konzept jedoch seine Umsetzung finden, ist es in jedem Fall angebracht eine der alternativen Lösungen für das Root-Zertifikat in Erwägung zu ziehen.

3.2 Methoden zur Entscheidungsfindung

Einen erste, kurze Recherche zu den in Frage kommenden PKI-Systemen lieferte mehr Ergebnisse als erwartet. Auch mussten die Anforderungen an die Zertifizierungsstelle klar benannt werden können, um ausgehend davon eine passende PKI-Implementierung auswählen zu können, welche auch zukünftigen, zu erwartenden Anforderungen gerecht wird.

Aus diesen Gründen werden nachfolgend jene Methoden zur Entscheidungsfindung vorgestellt, welche einerseits benutzt wurden, um den notwendigen Funktionsumfang zu ermitteln und welche andererseits für den Evaluierungsprozess zu Hilfe genommen wurden.

Intuitive Entscheidungsfindung: Über das Bauchgefühl, die intuitive Entscheidungsfindung, fließen oft in der Vergangenheit gemachte und meist unterbewusst verankerte Erfahrungen in den Entscheidungsfindungsprozess mit ein. Besonders bei komplexen Entscheidungen spielen diese Einflüsse auf jeden Fall immer dann eine Rolle, wenn alle technischen Analyseverfahren abgeschlossen sind und kein eindeutiges Ergebnis vorliegt, oder in jenen Situationen, in welchen nicht genügend Informationen oder zu wenig Zeit für einen wissenschaftlichen Entscheidungsfindungsprozess zur Verfügung stehen [Pfo77]. Das emotionale Erfahrungsgedächtnis, die sogenannten positiven und negativen somatischen Marker, lösen Körperreaktionen aus, die entweder für Zustimmung oder Ablehnung stehen können. Beispielsweise können sich positive Erfahrungen im Umgang mit Linux und negative Wahrnehmung bei der Verwendung von Windows, welche in der Vergangenheit gemacht wurden, indirekt auf Entscheidungen auswirken, wenn ein geeignetes Betriebssystemes für ein neues Projekt ausgewählt werden soll.
Sind diese Körpersignale bekannt, respektive ist eine Person in der Lage dies richtig zu deuten, kann damit eine Entscheidung hinterfragt werden, um sie zum Beispiel einem technischen Analyseerfahren zuzuführen. Insbesondere bei all jenen Entscheidungen, die sehr schnell getroffen werden müssen, ist es von Vorteil, seine somatische Marker richtig einschätzen zu können.

Negative Erfahrungen respektive Gelerntes, welches durch Bestrafung, in welcher Form auch immer erlangt wurde, prägen sich in das menschliche Bewusstsein zumeist stärker ein als positive Ereignisse [MSF06]. Erfahrungsentscheidungen sollten somit nie unreflektiert in den Entscheidungsfindungsprozess mit einfließen.

Wurden intuitive Entscheidungen in der Vergangenheit oft unter dem Vorwand nicht wissenschaftlich zu sein vom Entscheidungsfindungsprozess komplett ausgeschlossen, werden diese Methoden der Entscheidungsfindung in neueren wissenschaftlichen Arbeiten hingegen untersucht [AW12].

Consider All Facts: Die Methode Consider All Facts (CAF) nach Edward de Bono [DB04] ermöglicht es, in relativ kurzer Zeit einen Überblick über die Rahmenbedingungen des zu bewertenden Systemes zu gewinnen. Dabei sollen möglichst alle Einflussfaktoren ausfindig gemacht werden, welche sich direkt oder indirekt auf das zu bewertende System auswirken. Dieser Prozess kann zum Beispiel mit Brainstorming, Brainwriting oder anderen Kreativitätstechniken unterstützt werden, wobei darauf zu achten ist, dass die entsprechenden Regeln eingehalten werden. Allgemein ist es bei diesem Vorgehen natürlich von Vorteil, möglichst viel an Vorwissen zu besitzen. Dieses Fachwissen bring andererseits jedoch eine eingeschränkte Sichtweise mit sich, daher ist es hilfreich, bei diesem Entscheidungsfindungsprozess auch ausstehende Personen mit einzubeziehen.

Je mehr Kriterien gefunden wurden, desto leichter fällt im Allgemeinen die im Anschluss stattfindende Entscheidungsfindung. Wurden alle Charakteristika ausfindig gemacht, werden sie anschließend gruppiert und von etwaigen doppelten Vorkommen bereinigt. Abschließend wird noch grob zwischen wichtigen und eher unwichtigeren Merkmalen vorsortiert. Die Ergebnisse dieser in relativ kurzer Zeit durchführbaren Analyse können über den weiteren Projektverlauf als allgemeine Checkliste verwendet werden.

Gewichtete Entscheidungsmatrix: Sie bietet einen rationalen Ansatz um mehrere Systeme miteinander zu vergleichen und liefert am Ende ein eindeutiges Ergebnis. Im Unterschied zu einer einfachen Entscheidungsmatrix bei welcher alle Kriterien gleichrangig sind, wird dieses Verfahren eingesetzt, wenn sich die Eigenschaften unterschiedlich stark auf die Entscheidung auswirken.

Am Beginn beider Prozesse müssen zuerst alle Kriterien nach welchen die Bewertung stattfinden soll, klar definiert sein. Dabei muss jedes Kriterium positiv besetzt sein, d.h. je besser eine Eigenschaft vom zu bewertenden System erfüllt ist, desto höher ist dieses Kriterium zu bewerten. Im einfachen Fall werden alle gefundenen und zu bewertenden Eigenschaften eines Systemes in der

ersten Spalte einer Matrix untereinander angeführt. Daneben wird für jedes zu bewertende System eine weitere Spalte angefügt. Anschließend wird jedes Kriterium in jedem System jeweils mit Punkten zwischen eins und sechs bewertet, wobei sechs Punkte dann vergeben werden, wenn das Kriterium bei einem System optimal erfüllt ist. Abschließend werden alle Punkte einer Spalte, d.h. eines zu bewertenden Systemes addiert, sodass jenes System mit den meisten Punkten von dieser Methode als das am besten bewertete anzusehen ist.

Bei der gewichteten Entscheidungsmatrix wird gleich neben der Spalte mit den Kriterien eine weitere Spalte für die Gewichte der einzelnen Eigenschaften eingefügt, sodass jedes Kriterium prozentual gewichtet wird. Eine Addition aller Gewichte muss genau 100% ergeben. Die durchschnittliche oder mittlere Gewichtung errechnet sich, indem die Anzahl aller Kriterien durch 100 dividiert wird, sodass Kriterien mit Werten über der mittleren Gewichtung überdurchschnittlich wichtiger sind, als jene unter der mittleren Gewichtung. Abschließend werden die vergebenen Punkte mit der Gewichtung multipliziert und durch 100 dividiert. Die Addition der Summen ergibt das Gesamtergebnis. Diese System kann verfeinert werden in dem die Kriterien gruppiert werden und jede Gruppe extra gewichtet wird.

3.3 Anforderungsanalyse

Aus der Anforderungsanalyse, welche zum Teil mit oben angeführten Methoden durchgeführt wurde, gingen nachstehend angeführte grundlegende Funktionen hervor:

- Aufgrund der Firmenstruktur muss eine Bedienung der CA und RA über ein Web GUI möglich sein.
- Automatisches Erstellen bzw. Update, sowie Import und Export Funktionalität für die CRL
- CLI zum automatisierten Erzeugen und Exportieren von Zertifikaten (Batch processing), sowie für die automatisierten von wiederkehrenden Aufgaben
- Suchfunktion für User respektive deren Zertifikate
- RFC und IEC 62351 Konformität

Neben der Grundfunktionalität wurden weitere Kriterien wie folgt gruppiert:

- Benutzerfreundlichkeit und Support
- Sicherheit sowie
- Skalierbarkeit

Unter dem Überbegriff Benutzerfreundlichkeit wird nicht nur ein aufgeräumtes, mehrsprachiges und intuitiv zu bedienendes GUI verstanden, sondern auch eine ausreichend gut sortierte Dokumentation und ein entsprechend guter Support, auf welchen bei nicht lösbaren Problemen im Bedarfsfall zurückgegriffen werden kann. Dabei ist der Support-Bereich von besonderem Interesse.

Unter Sicherheit (Vertraulichkeit und Integrität) ist zum einen das geschützte Aufbewahren der privaten Schlüssel und Passwörter, und der andere userbezogenen Daten sowie eine verschlüsselte Verbindung zur PKI zu verstehen. Zum anderen sind unter diesem Punkt jedoch auch Dinge wie eine Rollen- und auf mehr Faktoren basierende Authentifizierung auf Basis von Zertifikaten, sowie laufende Sicherheitsupdates und Bugfixes der PKI-Software selbst, zu berücksichtigen.

Skalierbarkeit ist besonders für den zukünftigen Betrieb der Zertifizierungsstelle ein relevantes Thema. Während der Arbeit an diesem Projekt wurde die Struktur der Zertifizierungsstelle, siehe Abbildung 3.1 bereits mehrmals erweitert. Sollte das System angenommen werden, ist davon auszugehen, dass weitere SubCAs in die Struktur integriert werden und die PKI somit in der Lage sein muss, weitere Zertifikate zu verwalten. Aus diesen Gründen soll darauf geachtet werden, dass die ausgewählte PKI-Implementierung in der Lage ist, auch eine große Anzahl an Zertifikaten zu verwalten. Um diese zu evaluieren wurde unter anderem auch gezielt nach Referenzimplementierungen gesucht. Des weiteren ist es aus sicherheitstechnischer Sicht sinnvoll, dass sich die einzelnen Komponenten der PKI, wie beispielsweise RA, CA, Datenbanken und der gleichen auf physikalisch getrennter Hardware installieren lassen.

Nachfolgend werden zuerst die derzeit aktuellen PKI-Implementierungen einzeln vorgestellt und anschließend jene Systeme, welche für dieses Projekt in Frage kommen können, näher evaluiert.

3.4 Übersicht PKI-Implementierungen

Um X.509 Zertifikate zu erstellen oder zu verwalten, müssen zum einen die Vorgaben der relevanten RFCs eingehalten werden und zum anderen werden kryptographische Funktionen benötigt, um das asymmetrische Schlüsselpaar zu erstellen oder das Zertifikat zu signieren.

OpenSSL liefert neben den Bibliotheken zur Implementation des TLS-Protokolls eine Vielzahl an zusätzlichen Programmen mit diversen Funktionalitäten, sodass das Ausstellen und Widerrufen von Zertifikaten durchaus mit den von OpenSSL mitgelieferten Werkzeugen zu bewerkstelligen ist. Sollen jedoch mehrere Zertifikate verwaltet werden, so stoßen die von OpenSSL bereitgestellten Funktion für das Zertifikatsmanagement schnell an ihre Grenzen. Als vorteilhaft hingegen erweisen sich die Command-Line-Funktionen von OpenSSL, wenn es zum Beispiel darum geht ein Zertifikat von einem Format in ein anders überzuführen, aber auch wenn ein Zertifikat oder nur Teile davon in eine für den Menschen lesbare Form zu konvertieren sind. Aus diesem Grund existieren einige sowohl kommerzielle als auch freie Programme, welche alle nötigen Funktionen einer PKI in einem Gesamtpaket zur Verfügung stellen. Nachfolgend werden die wichtigsten frei zur Verfügung stehenden Programme zum Erstellen und Verwalten von Zertifikaten einzeln vorgestellt. Teilweise greifen die Implementierungen dabei auf OpenSSL zurück.

TinyCA[3] besteht aus Perl-Scripten, welche ein benutzerfreundliches GUI für OpenSSL bereitstellen. Das Tool selbst ist schnell installiert und auch eine CA bzw. ein Zertifikat ist mit ein wenig Grundlagenwissen in wenigen Minuten ausgestellt. Dabei werden alle gängigen Zertifikatsformate, sowie die wichtigsten Erweiterungen, unterstützt. Sowohl Zertifikate als auch CRLs lassen sich in den wichtigsten Formaten exportieren, sodass sie in den allermeisten Clients und Servern problemlos einsetzbar sind. TinyCA speichert dabei alle Daten, darunter auch die privaten Schlüssel im Filesystem, des Rechners ab. Die Software wurde in der frühen Phase dieses Projektes verwendet um Zertifikate für die ersten Tests der TLS-Verbindung zu realisieren. Neben der XCA eignet sich die TinyCA sehr gut, wenn einige wenige Zertifikate für, zum Beispiel eigene Webserver, zu erstellen und verwalten sind. Anleitungen zur Installation und der Erstellung von Zertifikaten finden sich in ausreichender Form im Internet.

XCA[4] ist in C++ geschrieben und bringt ebenfalls ein GUI mit. Gleich wie die TinyCA greift auch XCA für die Umsetzung der kryptografischen Funktionen auf die OpenSSL-Bibliothek zurück. Die Software ist plattformübergreifend und in den meisten Linux-Distributionen sehr einfach über die interne Paketverwaltung zu installieren. Die privaten Schlüssel werden passwortgeschützt in einer Berkeley-Datenbank gehalten. Dieses File kann auch optional, zum Beispiel aus Sicherheitsgründen, auf einen externen Datenträger, wie einem USB-Stick ausgelagert werden. Mit Hilfe eines nach dem PKCS #11-Format standardisierten Interfaces ist auch ein Verwalten von Smart-Cards möglich. XCA wurde in diesem Projekt verwendet um die Export und Import Funktionalität der EJBCA zu testen. Der Code für das Projekt wird sowohl auf GitHub[5] als auch

[5]https://github.com/chris2511/xca (Aufgerufen: Februar 2014).

auf SourceForge gehostet. Die letzten Code-Änderungen wurden Anfang 2011 eingepflegt. Auch zu dieser PKI-Implementierung befindet sich ausreichend Dokumentation und praxisbezogene Beispiele im Internet.

gnomint[6] bietet ein in GTK+ geschriebenes und in 12 Sprachen übersetztes UI, sowie ein CLI mit Batchprozessor zum automatischen Generieren von Zertifikaten. Grundsätzlich werden auch hier die wichtigsten Funktionalitäten zum Erstellen und Widerrufen von Zertifikaten unterstützt. Ähnlich wie bei der XCA gibt es auch hier die Möglichkeit die privaten Schlüssel der Zertifikate in externen Files zu speichern. Welche kryptographische Bibliothek im Hintergrund verwendet wird, konnte nicht ermittelt werden. Dokumentation respektive Anleitungen sind vergleichsweise wenige zu finden.

pyCA[7] ist eine in Pyton geschriebene PKI-Implementierung, welche 2003 bereits die wichtigsten Funktionalitäten unterstützte. Durch ein Webinterface mit einer integrierten Suchfunktion, sowie der Möglichkeit, des Speicherns der Zertifikate in einer LDAP-Datenstruktur, bringt diese Implementierung alle Voraussetzungen mit, um auch eine große Anzahl an Zertifikaten dezentral verwalten zu können. Das Projekt, welches 1999 aus einer Diplomarbeit [Str99] hervorging und als PKI-Lösung für die Firma Propack Data GmbH gedacht war, wird zur Zeit jedoch nicht mehr aktiv weiterentwickelt Bug-Fixes und Sicherheitsupdates werden derzeit noch eingefügt.

OpenCA ist ein klassisches Open-Source Projekt, welches bereits 1999 begonnen wurde. Seit 2001 wird der Sourcecode bei SourceForge [8] gehostet und weiterentwickelt. Das letzte Update wurde im August 2013 veröffentlicht. Dieses PKI-Tool ist in C und Perl geschrieben und verwendet zur Umsetzung der kryptographischen Funktionen OpenSSL. Für einen ordentlichen Betrieb der OpenCA sind zumindest ein Apache Webserver, OpenLDAP und eine Datenbank notwendig. Hinsichtlich der Datenbank besteht durch das Perl-DBI-Modul, Kompatibilität zu allen gängigen Datenbanken, wie zum Beispiel MySQL, Oracle oder PostgreSQL. Über das Web-Interface können sowohl alle funktionalen PKI-, als auch die administrative Aufgaben erledigt werden. Weiters zum OpenCA-Projekt gehören folgende Komponenten:

- LibPKI: eine Bibliothek, welche die grundlegende PKI-Funktionalität einfach für die Command-Line zur Verfügung stellt.
- OpenCA OCSPD: ein schlanker RFC 2560 konformer OCSP-Responder-Daemon
- PRQP Server: ein Daemon, welcher das PKI Resource Query Protocol (PRQP) zum Datenaustausch von PKI-Systemen untereinander, zur Verfügung stellt.

[8] http://sourceforge.net/projects/openca/(Aufgerufen: Februar 2014)

OpenCA gestattet den Einsatz von HSMs zum sicheren Speichern oder generieren des Schlüsselmaterials. Des weiteren wird unter anderem ein Batchprozessor zum automatisierten Erstellen von Zertifikaten oder das Simple Certificate Enrollment Protocol (SCEP) für einen vereinfachten Zertifikat Verteilungsprozess zur Verfügung gestellt.

OpenCA ist somit eine vollwertige PKI-Implementierung und für den Einsatz in Firmen und Organisationen gedacht, in denen sie sich tatsächlich auch im produktiv Betrieb bewährt hat. Darüber hinaus wird sie des öfteren als Referenz für andere Implementierungen herangezogen [GP06]. Mit OpenCA-ng (OpenCA-Next Generation) ist eine Neuimplementierung geplant, welche zusätzliche Funktionalitäten erhalten, sowie bekannte Einschränkungen der alten Version, eliminieren soll.
Die unter anderem langen Releasezyklen, von zum Teil zwei Jahren und die zum Teil nicht mehr vorhandene Dokumentation - Dead Links im Wiki und auf der Hompage des Projektes - hinterlassen nicht den besten Eindruck. Die alternativ dazu verlinkte pdf-Datei ist zwar mit 190 Seiten sehr ausführlich, jedoch bezieht sie sich auf die veraltete Version 0.9.2+ von OpenCA. Aktuell wird die PKI-Implementierung in Version 1.5.0 auf der Homepage zum Download angeboten. Die Anzahl der Downloads bei SourceForge lassen jedoch darauf schließen, dass OpenCA bei den Usern trotzdem Zuspruch findet.

OpenXPKI[9] entstand im Jahr 2005 aus der Abspaltung einer Gruppe der damaligen Hauptentwickler von OpenCA und hatte sich zum Ziel gesetzt eine Zertifizierungsstelle im „enterprise-level“ zur Verfügung zu stellen.

Auf Grund der Erfahrungen von anderen Projekten und mit der Kenntnis über deren Schwächen, wurde OpenXPKI nahezu von Grund auf neu konzipiert. Hauptaugenmerk wurde unter anderem auf die einfache Konfigurierbarkeit und auf ein modulares und skalierbares Konzept gelegt. Die Trennung der einzelnen Komponenten der PKI, wie zum Beispiel einer online Registration Authority (RA) und einer offline CA ist bei dem neuen Design optional. OpenXPKI ist ebenfalls in Perl geschrieben und läuft als Daemon auf Unix artigen Systemen. Über eine standardisierte „OpenXPKI Server API“ werden Interfaces zur Verfügung gestellt, welche die diversen Funktionalitäten, wie z.B.: SCEP oder OCSP, bereitstellen. Dieses Konzept ermöglicht es, die Software um neue Anforderungen zu erweitern und gibt den Benutzern die Möglichkeit die PKI ihren eigenen Bedürfnisse anzupassen. Primär verwendet diese PKI-Implementierung auch OpenSSL um die kryptographischen Funktionen bereit zu stellen, dabei wird jedoch eine Abstraktionsschicht eingezogen, womit es möglich ist auch andere kryptographische Bibliotheken zu verwenden. Des weiteren existiert eine Unterstützung für HSMs. Über einen DBAL können alle gängigen Datenbanken angebunden werden, wobei die Standardinstalla-

tion auf SQLite setzt. Mit dem Auditing-Modul besteht die Möglichkeit die laufenden Daten sowohl per Syslog als auch in einer Datenbaken zu protokollieren. Über einen Plugin-Mechanismus und Hooks, welche für bestimmte Events aufgerufen werden, können Monitoringsysteme eingebunden werden. Die Konfigurationsparameter für systemrelevante Dienste sowie für das Setup der PKI selbst, werden in XML-Notation wahlweise in einem oder mehreren Files gespeichert. Änderungen an der Konfiguration über das Web-Interface sind möglich, sofern diese über Policies freigegeben wurde. Über ein Vererbungsmodell können Zertifikate für SubCAs und End-Entities einfach an die geänderten Vorgaben angepasst werden.
Über sogenannte „PKI Realms“ wird der Betrieb von mehreren komplett voneinander unabhängigen CAs innerhalb einer PKI ermöglicht. Mit Hilfe dieser Technik und des automatischen „CA Rollover“, sollen Probleme, welche bei einem zeitlichen Ablauf der SubCAs-Zertifikate entstehen können, behoben werden. Dabei werden für einen hierarchischen Subtree innerhalb einer CA-Struktur mehrere SubCAs mit unterschiedlichen Gültigkeitszeiträumen verwendet. Die PKI erkennt dabei automatisch wann sie auf die aktuelle SubCAs wechseln muss. Die Möglichkeit eines „multi-node“ Setups mit automatischer „fail-over“ Funktionalität soll es ermöglichen, dass OpenXPKI auch in sehr großen Umgebungen einsetzbar ist [AMM06], [Bar05].

Das Projekt wechselte 2012 nach GitHub[10]. Die Commit-Frequenz lässt auf eine derzeit aktive Weiterentwicklung des Projektes schließen. Weiters werden vom OpenCA-Projekt einige Teile wie zum Beispiel die „openca-tools“, welche unter anderem das SCEP implementiert, aktiv weiterentwickelt. Zum Zeitpunkt der Evaluierung stand die Hompage und somit der Großteil der Dokumentation noch veraltet und unübersichtlich im Subversion Repository. Im Moment jedoch befindet sich die Website gerade im Umbau. Community Support für die OpenXPKI erhält man über die mailing lists. Des weiteren wird kommerzieller Unterstützung von der Firma Cynops GmbH angeboten.

r509[11] stellt eine in Ruby geschriebene API zur Verfügung und verwendet, wie die meisten anderen hier vorgestellten PKI-Implementierungen zur Erzeugung des kryptographischen Materials im Hintergrund, wieder OpenSSL. Somit vereinfacht auch r509 das Handling im Umgang mit Zertifikaten auf Basis der am weitesten verbreiteten kryptographischen Open-Source-Bibliothek. Mit diversen weiteren Projekten[12] wie zum Beispiel:

- r509-ocsp-responder
- r509-ca-http

[10] https://github.com/openxpki (Aufgerufen: Februar 2014)
[12] https://github.com/r509 (Aufgerufen: Februar 2014)

wird eine komplette PKI nach RFC 5280 für produktive Umgebungen bereitgestellt. Die Homepage des Projektes bietet eine Installationsanleitung und eine im Verhältnis zur Komplexität des Themas eher minimale Dokumentation. Eine grundlegende Beschreibung der Architektur respektive des Konzeptes oder eine genauere Auflistung der Funktionalität konnte nicht gefunden werden. Die Code Frequenz auf GitHub deutet auf eine aktuelle Entwicklung der Software hin.

EJBCA wird von der schwedischen Firma PrimeKey Solutions AB[13] unter der Beteiligung von Freiwilligen entwickelt und ist somit die einzige hier vorgestellte PKI-Implementierung hinter, welcher ein kommerziell agierendes Unternehmen steht. Dieser Umstand spiegelt sich unter anderem in der Gestaltung der Homepage und der Dokumentation wieder, welche immer am aktuellen Stand sind und sich von ihrer Quantität klar von den anderen Kandidaten abhebt. Des weiteren befinden sich im Internet diverse Anleitungen und Blogposts, sodass daraus geschlossen werden kann, dass diese PKI-Implementierung auch von der Community angenommen wird.

Der gesamte Sourcecode ist unter der GNU Lesser General Public License (LGPL v2.1) lizenziert und wird unter anderem von PrimeKey selber gehostet[14]. Die bereitgestellte Funktionalität ist entsprechend für den Einsatz in sehr großen Zertifizierungsstellen ausgelegt. Trotzdem unterscheidet sich der Aufwand für eine minimale Installation nicht wesentlich von den anderen hier vorgestellten Lösungen. Konfigurationsdateien mit sinnvollen Voreinstellungen und die Verwendung des „Apache Ant build tools“ sowie zertifizierte Software downloads sind einige Beispiel für das durchdachte Konzept.
PrimeKey bietet für den Einsatz der EJBCA als Zertifizierungsstelle in Unternehmen und Organisationen neben den Community-Support ein zweistufiges professionelles Supportmodell an, welches unter anderem eine vor Ort Unterstützung ermöglicht.

Sowohl auf die Architektur, als auch auf die für diese PKI-Implementierung besonderen Funktionalitäten wird in den nachfolgenden Kapiteln näher eingegangen werden.

Dogtag gehört neben der EJBCA und der OpenXPKI zu den am weitesten entwickelten freien PKI-Implementierungen. Das Projekt wurde von Red Hat 2008 initialisiert und zuletzt im April 2014 in der Version 10.1.1 veröffentlicht.

Obwohl Dogtag plattformunabhängig konzipiert ist, einige Komponenten der Software sind in Java geschrieben und setzen eine Apache Tomcat voraus,

[13] http://ejbca.org/ (Aufgerufen: Februar 2014)
[14] http://ejbca.org/repository.html (Aufgerufen: Februar 2014)

andere Subsysteme setzten auf C++ und Apache als Webserver, werden die Installationsanleitungen nur für Red Hat Distributionen zur Verfügung gestellt. Zur Interaktion mit der PKI stehen sowohl ein Web-Interface als auch ein CLI bereit. Zum Speichern der Daten kommt zum einen der Fedora Directory Server, eine freie Open Source LDAP Implementierung, zum anderen eine SQLite Datenbank zum Einsatz. Ein1 Austausch dieser beiden Systeme, bzw. die Verwendung alternativer Software ist derzeit nicht möglich. Als Back-End, zur Bereitstellung der kryptographischen Funktionalität, werden die von der Mozilla Foundation entwickelten Bibliotheken Network Security Services (NSS) und Network Security Services for Java (JSS) verwendet. Dabei wird unter anderem die Verwendung von HSMs ermöglicht oder die Unterstützung des PKCS #11 gewährleistetet.

Die unter mehreren freien Lizenzen veröffentliche PKI-Implementierung wurde bereits 2009 in einschlägigen Magazinen [Sch09] als funktionierende Komplettlösung im Unternehmensbereich vorgestellt. Das US Department of Defense betreibt eine modifizierte Version dieser PKI-Implementierung, in welcher 10 Millionen Zertifikate verwaltet werden [MA11]. Red Hat bietet keine offiziellen Support-Modelle für Dogtag an, stattdessen wird auf den sogenannten „Community Support“ über Mailing-Lists und IRC-Channels verwiesen. Die Dokumentation auf der Homepage und im Wiki sind diesbezüglich detailliert. Des weiteren werden How Tos für unterschiedlichste Szenarien bereitgestellt.

3.4.1 Gruppierung und Zusammenfassung

Die oben vorgestellten PKI-Implementierungen lassen sich zusammenfassend in zwei Gruppen unterteilen:

- standalone Varianten und
- vollwertige respektive verteilte Lösungen

Die beiden Gruppen haben zum Teil gravierende Unterschiede in den Bereichen Sicherheit, Skalierbarkeit und Administrationsaufwand.

3.4.2 Standalone Varianten

Dabei haben die standalonen Varianten den Vorteil, dass sie sehr schnell, in den meisten Fällen über das Paketmanagement, zu installieren sind und ohne größeren Konfigurationsaufwand einsetzbar sind. Dafür stoßen sie gerade bei der Verwaltung von mehreren Zertifikaten schnell an ihre Grenzen.

Des weiteren ist eine rollenbasierte Authentifizierung schwer bis gar nicht umsetzbar, sodass es im Nachhinein nicht nachvollziehbar ist, wer für wen, wann und warum

ein bestimmtes Zertifikat ausgestellt oder widerrufen hat. Auch der sicherheitstechnische Aspekt wird nur teilweise erfüllt. Ein entferntes Verwalten von Zertifikaten bzw. das verteilte Arbeiten an diesen PKI-Systemen ist nur über Umwege, zum Beispiel mit Hilfe von SSH möglich.

Standalone Varianten eigenen sich somit gut zum Verwalten einiger weniger Zertifikate in kleineren Strukturen, bei geringem Installations- und Wartungsaufwand. Zu dieser Gruppe zählen:

- TinyCA
- XCA
- gnomint

Dabei wurden mit TinyCA und auch mit der XCA durchaus positive Erfahrungen gemacht. Wobei die XCA durch einige nützliche Zusatzfunktionen einen besseren Gesamteindruck hinterlässt.

Die vollwertigen bzw. verteilten PKI-Implementierungen, dazu gehören Dogtag, EJBCA, r509, OpenXPKI, OpenCA und pyCA, werden nachvolgend im Detail evaluiert.

3.4.3 Evaluierung der vollwertigen PKI-Implementierungen

TinyCA, XCA und gnomint erfüllen wichtige prinzipielle Anforderungen, wie sie in Kapitel 3.3 aufgestellt wurden, nicht und scheiden somit bereits im Vorhinein aus.
Die pyCA würde diese Grundfunktionalitäten erfüllen wird aber derzeit nicht mehr weiter entwickelt, und wird somit ebenfalls nicht bewertet.
Auch die r509 erfüllt, soweit dies beurteilt werden kann, sämtliche minimale Anforderungen, welches dieses Projekt an eine Zertifizierungsstelle stellt. Die zum Teil sehr spärliche Dokumentation und das Nichtvorhandensein eines Live-Demos oder eines virtuellen Images machte es unmöglich, dieses System mit einem vertretbaren Zeitaufwand zu evaluieren.

Nachstehende vier PKI-Implementierungen werden somit im Detail genauer evaluiert:

- OpenCA
- OpenXPKI
- EJBCA und
- Dogtag

Informationsquellen wie Homepages, Withepapers, oder andere wissenschaftliche Arbeiten, welche als Grundlage der Bewertung dienen, wurden bereits im Kapitel 3.4 angeführt.
Als Evaluierungsmethode wurde eine gewichtete Entscheidungsmatrix, wie sie ebenfalls bereits in 3.2 beschrieben wurde, verwendet. Dabei wurden die Kriterien wie folgt gruppiert:

- Grundfunktionalität
- Benutzerfreundlichkeit
- Sicherheit
- Skalierbarkeit und
- Weiteres

Wobei die ersten vier Gruppen aus der Anforderungsanalyse (vgl.: 3.3) stammen. In der letzten Gruppe hingegen sind Kriterien zusammengefasst, welche sich im Nachhinein, d.h. im Zuge der Evaluierung, ergeben haben.

In der Gruppe Sicherheit, welche die sechs Kriterien Vertraulichkeit, Integrität, Authentifizierung, Release Zyklen, Sicherheitsupdates und Bugfixes, sowie gesicherter SW download umfasst, liegt die subjektiv vergebene mittlere Gewichtung bei ungefähr 16%. Da alle Kandidaten die ersten drei Kriterien (Vertraulichkeit, Integrität und Authentifizierung) nahezu gleich gut erfüllen, wurden diese Eigenschaften absichtlich unterbewertet, um den restlichen drei Kriterien mehr Gewicht verleihen zu können.

In der Gruppe Weiteres wurde unter „Modularität, Abhängigkeit“ bewertet, ob sich die Komponenten, wie zum Beispiel der Webserver, die Datenbank oder der LDAP-Verzeichnisdienst, austauschen lassen, sodass es einerseits möglich ist Software zu verwenden, welche bereits im Betrieb eingesetzt wird und andererseits die Sicherheit besteht, dass sich keine weiteren Abhängigkeiten zu anderen Softwareprojekten ergeben.
Unter „Plattform (Language)“ wurde das der PKI-Implementierung zu Grunde liegende Framework respektive die Programmiersprache bewertet. Wichtig dabei war, dass Techniken verwendet werden, welche aktuell sind, aber auch, dass das Framework von einer genügend großen Community unterstützt wird.
Alle hier bewerteten Implementierungen sind vollwertige PKI-Systeme, sodass die Unterstützung von HSMs, das Bereitstellen von SCEP oder OCSP und der gleichen von jedem Kandidaten erfüllt werden. Unter dem Punkt „erweiterte Funktionalität“ wird bewertet, in wie weit die PKI-Implementierung Funktionalität anbietet, welche über das Standardmaß hinausreicht. Darunter fallen zum Beispiel eine automatische Benachrichtigung per Mail an Administratoren, Template Systeme, breite Unterstützung von kryptographischen Algorithmen oder ein automatisches Rollover.

Ergebnisse

Tabelle 3.1 zeigt die gewichtete Bewertungsmatrix für die vier zu untersuchenden PKI-Implementierungen.
Aus der Evaluierung geht die EJBCA der Schwedischen Firma PrimeKey Solutions AB mit 29,1 Punkten als zu präferierende PKI-Implementierung hervor.

Nachstehend eine Vervollständigung der in Kapitel 3.4 ausgeführten Konzepte der einzelnen PKI-Implementierungen.

Nach der EJBCA erhält die OpenXPKI die zweitbeste Bewertung, welche bei vielen Kriterien im Vergleich mit der EJBCA, als äquivalent zu betrachten ist. Hinsichtlich der Skalierbarkeit und der angebotenen Funktionalität sind beispielsweise keinerlei Unterschiede zu erkennen. Jedoch überzeugte das Supportmodell für Unternehmen nicht vollständig und die sich im Umbau befindliche Homepage und die damit verbundenen unvollständige Dokumentation wurden nachteilig bewertet.
Dogtag steht prinzipiell im Funktionsumfang nicht weit hinten an, jedoch zwingen Designentscheidungen dazu, dass beispielsweise nur eine Datenbankimplementierung verwendet werden kann. Somit bestehen Abhängigkeiten, welche unter anderem einen Mehraufwand erfordern könnten, sollten die Anwender mit der zum Betrieb der PKI benötigten Software nicht vertraut sein. Andererseits besteht ein erhöhtes Risiko dahingehend, dass bei der Einstellung des Softwareprojektes die darauf basierende PKI-Implementierung unbrauchbar wird. Allgemein hinterlässt Dogtag den Eindruck, als ob sich diese PKI-Implementierung hauptsächlich an RedHat Kunden richten würde.
OpenCA, war in den vergangenen Jahren Maß der Dinge hat jedoch gegenüber der Konkurrenz ein wenig an Boden verloren. Es bleibt abzuwarten ob und in welche Richtung sich dieses Projekt in Zukunft weiter entwickeln wird.

3.5 Komponenten und Architektur einer PKI am Beispiel der EJBCA

Vollwertige PKI-Implementierungen setzen unter anderem ein hohes Maß an Fachwissen im Bereich der Systemadministration voraus. Um ihren Betrieb zu ermöglichen ist der Einsatz von Datenbanken, Verzeichnisdiensten, Webserver, Mailserver und dergleichen erforderlich. Dabei müssen alle diese Dienste ständig gewartet respektive upgedatet und abgesichert werden, um einen sicheren Betrieb der PKI gewährleisten zu können. Steigt die Anzahl der zu verwaltenden Zertifikate an, steigen auch die Anforderung hinsichtlich der Ausfallsicherheit und der Lastverteilung. In solchen Szenarien sind somit auch Dinge wie Backup und Recovery, Datenbankreplikationen oder Blackout Tests mit zu berücksichtigen. Eine fachgerechte Installation, sowie die Aufrechterhaltung des laufenden Betriebes einer PKI-Implementierung er-

	Gewicht	OpenCA		OpenXPKI		EJBCA		Dogtag	
Grundfunktionalität									
Web GUI	20	6	1.2	6	1.2	6	1.2	6	1.2
CLI Batch processing	20	6	1.2	6	1.2	6	1.2	6	1.2
CRL support, updates, im-export	20	6	1.2	6	1.2	6	1.2	6	1.2
import/export von Zertifikaten	20	6	1.2	6	1.2	6	1.2	6	1.2
Suchfunktion	20	6	1.2	6	1.2	6	1.2	6	1.2
Gruppenbewertung			6		6		6		6
Benutzerfreundlichkeit									
Bedienbarkeit GUI	20	4	0.8	4	0.8	6	1.2	4	0.8
mehrsprachiges GUI	10	4	0.4	4	0.4	4	0.4	4	0.4
Dokumentation	30	2	0.6	3	0.9	6	1.8	5	1.5
Support	40	3	1.2	5	2	6	2.4	4	1.6
Gruppenbewertung			3		4.1		5.8		4.3
Sicherheit:									
Vertraulichkeit (Confidentiality)	10	6	0.6	6	0.6	6	0.6	6	0.6
Integrität (Integrity)	10	6	0.6	6	0.6	6	0.6	6	0.6
Authentifizierung (Rollen basiert)	10	6	0.6	6	0.6	6	0.6	5	0.5
Release Zyklen	30	1	0.3	4	1.2	6	1.8	4	1.2
Sicherheitsupdates und Bugfixes	20	1	0.2	4	0.8	6	1.2	4	0.8
gesicherter SW download	20	3	0.6	3	0.6	6	1.2	3	0.6
Gruppenbewertung			2.9		4.4		6		4.3
Skalierbarkeit									
Referenzimplementierung	20	3	0.6	4	0.8	6	1.2	5	1
Trennung der Komponenten	40	6	2.4	6	2.4	6	2.4	4	1.6
Load balancing im Konzept	10	1	0.1	6	0.6	5	0.5	1	0.1
Module, Plugins, Erweiterbarkeit	30	6	1.8	6	1.8	5	1.5	4	1.2
Gruppenbewertung			4.9		5.6		5.6		3.9
Weiteres									
Kosten / Lizenzmodell	10	6	0.6	6	0.6	6	0.6	6	0.6
Modularität, Abhängigkeit	30	6	1.8	6	1.8	5	1.5	4	1.2
Community Akzeptanz	20	6	1.2	6	1.2	6	1.2	5	1
Plattform (Language)	10	6	0.6	6	0.6	6	0.6	6	0.6
erweiterte Funktionalität	20	3	0.6	6	1.2	6	1.2	4	0.8
Live Demo	10	6	0.6	6	0.6	6	0.6	3	0.3
Gruppenbewertung			5.4		6		5.7		4.5
Gesamtergebnis			22.2		26.1		29.1		23

Tabelle 3.1: Ergebnisse der Evaluierung der PKI-Implementierungen

fordert, somit einen nicht zu vernachlässigenden personellen Aufwand.

Im weiteren Verlauf dieser Arbeit wird hauptsächlich auf die integralen Bestandteile einer Public-Key-Infrastruktur am Beispiel der EJBCA eingegangen. Auf die Beschreibung der externen Komponenten wird ausschließlich dann ausgeführt, wenn es für das Verständnis notwendig ist. Anderenfalls wird an den entsprechenden Stellen auf weiterführende Literatur verwiesen.

3.5.1 Komponenten einer Zertifikatsinfrastruktur

Zu den integralen Bestandteilen einer PKI gehören im Wesentlichen folgende Komponenten:

- Registrierungsstelle RA (Registration Authority)
- Zertifizierungsstelle CA (Certificate Authority)
- Zertifikatssperrliste CRL (Certificate Revocation List)
- Validierungsdienst OCSP (Online Certificate Status Protokol)
- Logging- und Backup-Server

Die Aufgabe einer RA, welche über ein öffentliches Netzwerk direkt erreichbar ist, besteht darin, die User an der PKI zu registrieren, ihre Daten festzuhalten und die Anfragen zum beglaubigen der Schlüssel entgegenzunehmen und zu überprüfen. Diese Anfragen zur Erstellung respektive zum Signieren von Zertifikaten werden Certificate Signing Request (CSR) genannt.

Das Ausstellen bzw. das Signieren der Zertifikate, also das Verknüpfen der Identität mit den öffentlichen Schlüsseln ist Aufgabe der CA. Dabei stellt aus Sicherheitsgründen niemals die Root-CA selber, oder die übergeordneten SubCAs direkt Zertifikate für EEs aus. Die hier verwalteten Daten sind im höchsten Maß sicherheitsrelevant. Aus diesem Grund sollte die CA nicht direkt über ein öffentliches Netz erreichbar sein.
Je nach Art der PKI-Implementierung leitet die RA alle CSRs gesammelt an die CA weiter oder die CA selber fragt in regelmäßigen Abständen die RA nach zu verarbeitenden Anfragen ab. Letzteres ist bei der EJBCA der Fall.

Zertifikate können aus diversen Gründen entweder dauerhaft oder temporär ihre Gültigkeit verlieren. Um dies zu prüfen werden Zertifikatssperrlisten und Validierungsdienste eingesetzt. Der parallele Betrieb beider Techniken kann unter Umständen sinnvoll sein. CRLs beinhalten im Wesentlichen die Seriennummer der gesperrten Zertifikate sowie den Grund der Sperrung. Die Veröffentlichung erfolgt in festgelegten Intervallen, zum Beispiel über einen Verzeichnisdienst oder per Webserver für

jede CA getrennt. Der Vorteil dieser Technik ist, dass die CRL, sobald sie einmal importiert wurde, immer wieder abgefragt werden kann. Eine dauerhafte Verbindung zur PKI ist somit nicht zwingend notwendig. Der Nachteil liegt jedoch darin, dass Sperrlisten schnell sehr groß werden können, und dass sie zumeist manuell importiert werden müssen. Je nach Veröffentlichungsintervall der CRLs besteht darüber hinaus die Gefahr, dass bereits widerrufene Zertifikate nicht als solche erkannt werden.
Ein Validierungsdienst, welcher beispielsweise mit Hilfe des OCSP umgesetzt wird, hat den Vorteil, dass jedes Zertifikat einzeln in Echtzeit auf seine Gültigkeit überprüft werden kann. Dies setzt natürlich voraus, dass der Validierungsdienst jederzeit öffentlich erreichbar ist und dass eine Verbindung zur CA besteht.

In Fällen einer Kompromittierung oder bei Fehlern, welche im laufenden Betrieb auftreten können, muss nachvollziehbar sein, aus welchem Grund das Problem aufgetreten ist. Deshalb ist die Aufzeichnung der internen Vorgänge, ebenso wie ein funktionierendes Backup und Recovery, ein integraler Bestandteil einer PKI.

3.5.2 Verteiltes Setup Konzept der EJBCA

Abbildung 3.2 zeigt ein mögliches verteiltes Setup der für den Betrieb einer Zertifizierungsstelle notwendigen Komponenten, am Beispiel der EJBCA. Dabei ist das Verteilen der einzelnen Komponenten der PKI auf voneinander physikalisch getrennter Hardware sowohl aus sicherheits- als auch aus performancetechnischen Aspekten, sinnvoll.
Direkt aus einem öffentlichen Netz zu erreichen, sind der OCSP Responder, der Verzeichnisdienst zum Veröffentlichen der Zertifikate sowie die RA. Die CA, als auch das Backup und Logging sind durch eine weitere Firewall (FW2) geschützt. Und sind somit ausschließlich über ein internes Subnetz erreichbar. Einzelne Komponenten der PKI, welche erwartungsgemäß mehrere hundert Requests pro Sekunde zu beantworten haben oder möglichst ausfallsicher konzipiert sein müssen, können mehrfach ausgeführt, hinter einem vorgeschalteten Load Balancer, Platz finden. Dabei stellt beispielsweise der OCSP-Responder der EJBCA eine interne Funktionalität zur Verfügung, welche es dem Administrator ermöglicht dieses Service in ein externes Monitoring und Load-Balancing System zu integrieren.

3.5.3 Architektur der EJBCA

Die EJBCA wird auf dem Enterprise JavaBeans (EJB) Framework entwickelt, welches standardisierte Schnittstellen für die Entwicklung verteilter Softwaresysteme mittels Java bereitstellt. Mit Hilfe des JBOSS respektive WildFly Application Server können somit nicht nur Konzepte für Unternehmensanwendungen realisiert werden, sondern auch Datenpersistenz, Sicherheit oder Ressourcenmanagement gewährleistet werden. Dabei ist ein Austausch des Application Server möglich.

[15] http://www.ejbca.org/docs/architecture.html (Aufgerufen: Jänner 2014)

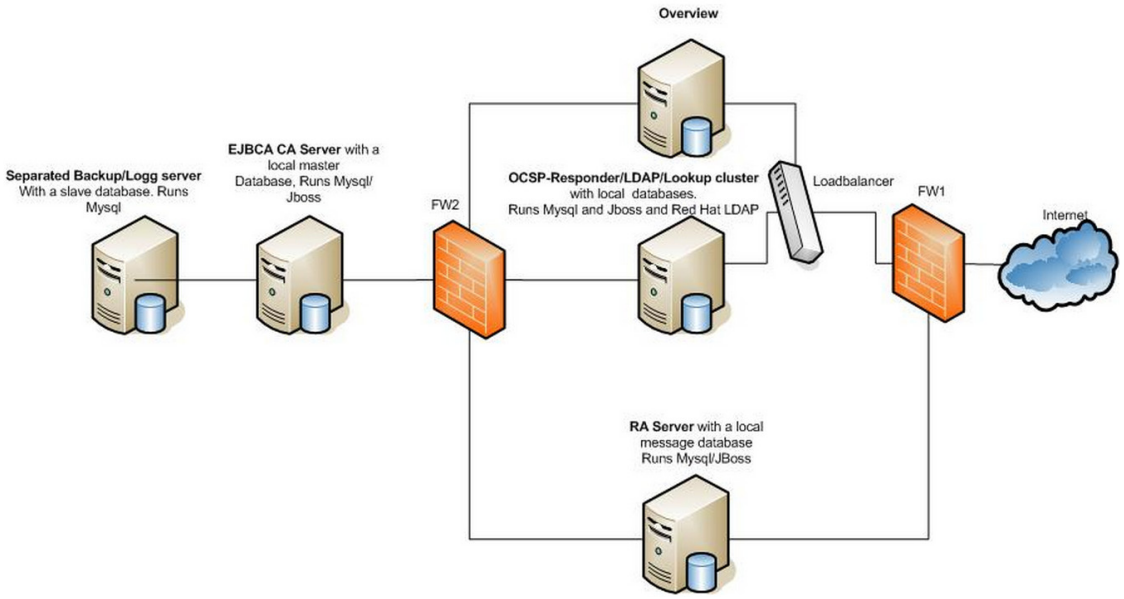

Abbildung 3.2: Möglichkeit eines verteilten Setups der EJBCA - Quelle: EJBCA[15]

Die PKI-Implementierung wird sowohl in einer Community- als auch in einer Enterprise-Edition angeboten. Der Funktionsumfang[16] der Community Edition ist für die Realisierung dieses Projektes mehr als ausreichend.
Zur Administration der PKI, stehen neben einem Web-GUI, das in mehrere Sprachen übersetzt ist, darunter auch Deutsch und Englisch, eine Web-API (Web Services) und ein CLI, zur Verfügung. Sodass alle Tätigkeiten sowohl manuell, als auch automatisiert, über unterschiedliche Methoden umsetzbar sind.
Neben dem Application Server lassen sich weitere essentielle Services wie zum Beispiel die Datenbank oder der Verzeichnisdienst austauschen bzw. durch eigen Plugins erweitern. Der Installationsprozess ist mit Hilfe des Apache Ant Tools[17] einfach realisiert. Dabei ist es einerseits möglich alle Komponenten der PKI gemeinsam auf einem Rechner, in einem sogenannten „all-in-one-setup“, einzurichten. Die Migration auf ein verteiltes System ist durchaus auch nach der Grundinstallation bzw. mit dem Anwachsen der zu verwaltenden Zertifikate realisierbar.

Hinsichtlich der Erstellung und Verwaltung von Zertifikaten und der Validierung werden alle notwendigen internationalen Standards, welche durch diverse RFC vorgegeben werden, sowie viele nationale Erweiterungen wie zum Beispiel das „Cert-Hash“, eine in deutschen Behörden verbreitete Absicherung des OCSP, unterstützt.

[16] http://ejbca.org/features.html (Aufgerufen März 2014)
[17] http://ant.apache.org/ (Aufgerufen März 2014)

Eine „client toolbox“ stellt Funktionalität bereit, um CLI basierend automatisch die Funktionalität der PKI zu testen. Ein automatisches Benachrichtigungssystem per E-Mail, ein Profil bzw. Vorlagensystem für Zertifikat- und Userverwaltung oder die Unterstützung für Cross-Zertifikate lassen auf ein durchdachtes Konzept schließen. Abbildung 3.3 zeigt die interne Architektur respektive die logischen Schichten der EJBCA, die grundlegenden Funktionen sowie die Schnittstellen zu den Clients und die zusätzlich benötigten Services.

Für die Umsetzung einer Zertifizierungsstelle auf Basis der EJBCA in Konzer-

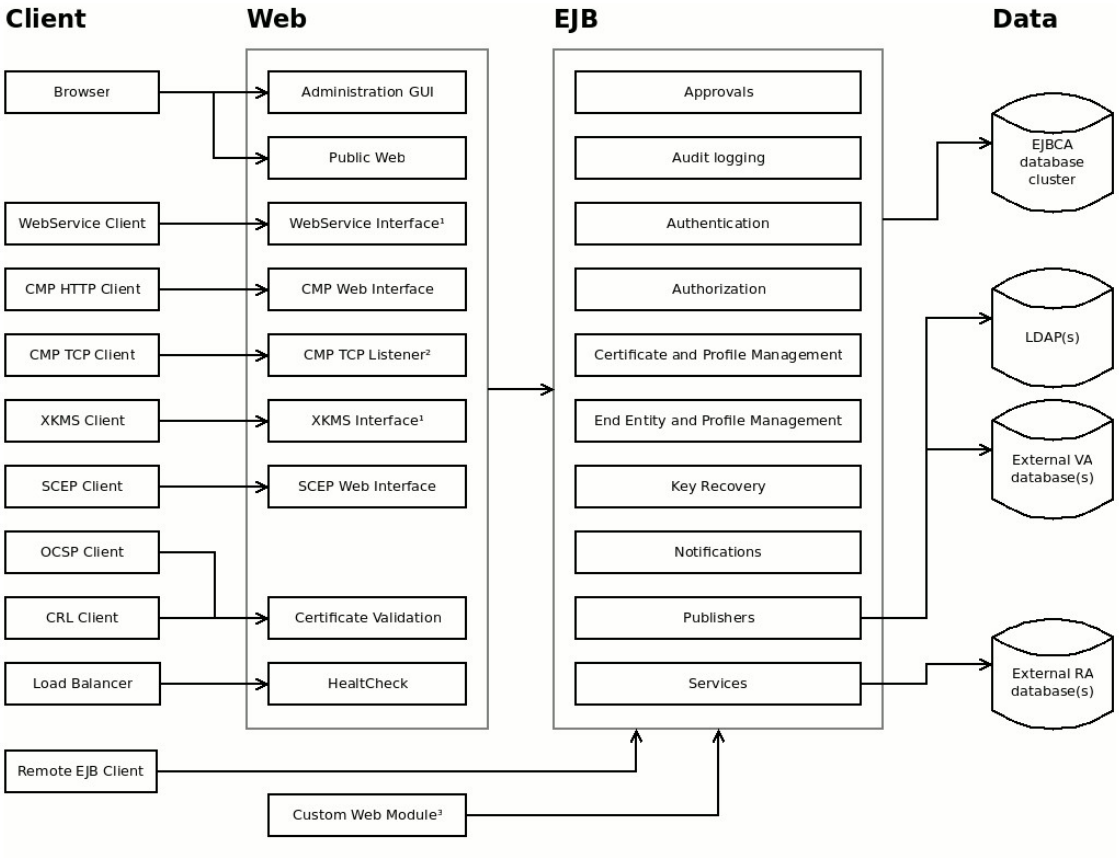

Abbildung 3.3: Übersicht der Architektur - Quelle: EJBCA[18]

nen und größeren Organisationen, werden zumindest zwei professionelle Support-Modelle angeboten. Der „EJBCA Enterprise Support“ beinhaltet dabei unter anderem auch ein Upgrade von der Community Edition zur Enterprise Version der PKI-Implementierung.

[18]http://www.ejbca.org/docs/images/architecture.png (Aufgerufen: Jänner 2014)

Die sehr umfangreiche Funktionspalette, Referenzimplementierungen[19] mit zum Teil mehr als 100.000 zu verwaltenden Zertifikaten, sowie die Tatsache, dass neben der kommerziellen Weiterentwicklung durch PrimeKey Solutions AB auch eine große Community hinter diesem Projekt steht, lässt darauf schließen, dass diese PKI-Implementierung auch in Zukunft Bestand haben wird.

[19] http://ejbca.org/installations.html (Aufgerufen: Jänner 2014)

Kapitel 4

Design

Nachstehend wird das Design dieses Projektes anhand von real existierenden und fiktiv erdachten Use-Cases beschrieben.

Anwendungsfälle ergeben sich beispielsweise durch die Umsetzung der im Sicherheitsstandard [IEC13b] angeführten Vorgaben. Aber auch durch erweiterte Anforderungen, welche im Verlauf des Projektes entstanden sind.
Da es sich bei diesem Projekt um eine konzeptionelle Umsetzung handelt und es zur Zeit nicht vollständig absehbar ist, welche Funktionalität schlussendlich benötigt wird, wurden einige Use-Cases möglichst realitätsnahe angenommen. In wie weit sie schlussendlich eine Umsetzung finden, bleibt abzuwarten. Eine Umsetzung dieser fiktiven Anwendungsfälle zielt hauptsächlich darauf ab, zu prüfen, in wie weit sich die PKI-Software an spezifische Prozesse anpassen und sich somit in eine bestehende Struktur integrieren lässt.

4.1 Use-Case 1: System Setup

Auf ein verteiltes Setup der PKI-Implementierung, wie es in Kapitel 3.5.2 beschrieben ist, wird in erster Linie deshalb verzichtet, da es sich in diesem Projekt um die Umsetzbarkeit einer Zertifizierungsstelle für Geräte im embedded Bereich handelt. Es ist nicht das primäre Ziel, ein möglichst sicheres und leistungsstarkes System zu konzipieren, vielmehr soll gezeigt werden, dass ein automatisches Erstellen und Verteilen der Zertifikate auf einem embedded Device möglich ist. Dies hat zur Folge, dass alle Komponenten der PKI-Implementierung in einem „all-in-one-setup“, auf einem handelsüblichen Rechner installiert werden.
Darüber hinaus ist die Topologie des firmeneigenen Netzwerkes, sowie die dort bereits eingesetzten Services, wie zum Beispiel Datenbanken, Verzeichnisdienste und dergleichen, nicht bekannt, sodass es auch aus diesen Gründen nicht zielführend ist, eine angepasste Referenzinstallation abzuliefern. Ferner wird davon ausgegangen, dass die eingerichtete PKI ausschließlich zu Testzwecken verwendet und anschließend evaluiert wird, d.h. dass eine Zertifizierungsstelle für einen Produktivbetrieb

neu eingerichtet werden muss und dass eventuell eine Erweiterung der Struktur der CAs sowie der PKI Funktionalität selbst in Betracht gezogen werden muss.

Hinsichtlich der Austauschbarkeit der Komponenten soll gezeigt werden, wie eine externe Datenbank am Beispiel von MySQL in das System zu integrieren ist.
Da die PKI in der Evaluierungsphase von mehreren Personen für unterschiedliche Zwecke eingesetzt wird, soll es möglich sein unterschiedlich konfigurierte und von einander unabhängige Installationen der EJBCA auf einem System betreiben zu können.

4.2 Use-Case 2: Abbildung der CA Strukturen

Aus sicherheitstechnischen und organisatorischen Gründen soll die PKI in mehrere der RootCA untergeordneten SubCAs gegliedert werden.
Die oberste Strukturebene der Zertifizierungsstelle ist in Abbildung 3.1 ersichtlich, nachstehend wird die Funktionalität respektive der Aufgabenbereich der benötigten SubCAs im Detail beschrieben.
Die dabei entstehende Struktur ist in der PKI unter der Berücksichtigung, dass eine Erweiterung in jedem Fall gegeben sein muss, abzubilden.

4.2.1 HW Manufac Intermediate SubCA

Die „HW_Manufac_Intermediate“ SubCA ist hierarchisch gesehen direkt unter der RootCA angeordnet und ist nicht berechtigt Zertifikate für EEs auszustellen. Sie dient ausschließlich zur Verwaltung der ihr untergeordneten Hardware-Manufacturer-SubCAs und ist somit dazu berechtigt, Zertifikate für weitere SubCAs auszustellen und zu widerrufen. Im Wesentlichen dient sie als logische Abstraktionsschicht respektive als Container für die Zertifizierungsstellen der Hardwarehersteller. Auf diese Weise wird eine bestmögliche Verwaltung bzw. Management der darunterliegenden SubCAs gewährleistet. Abbildung 4.1 zeigt die Teilstruktur der „HW_Manufac_Intermediate“ SubCA sowie zwei exemplarische Hardware Manufacturer SubCAs mit Zertifikaten für die TLS-Server auf den embedded Devices.

Hardware-Manufacturer SubCAs

Für die Herstellung und Auslieferung der embedded Devices werden diverse externe Firmen, die sogenannten Hardware-Manufacturer beauftragt. Nach der mechanischen Konstruktion der Geräte, müssen neben den Uboot- und Linux-Image auch die für die Authentifizierung der TLS-Kommunikationspartner notwendigen X.509-Zertifikate in das Gerät eingebracht werden. Dies geschieht bei den Hardware-Herstellern vor Ort.
Nach Abschluss des Produktionsvorganges werden alle embedded Devices einem

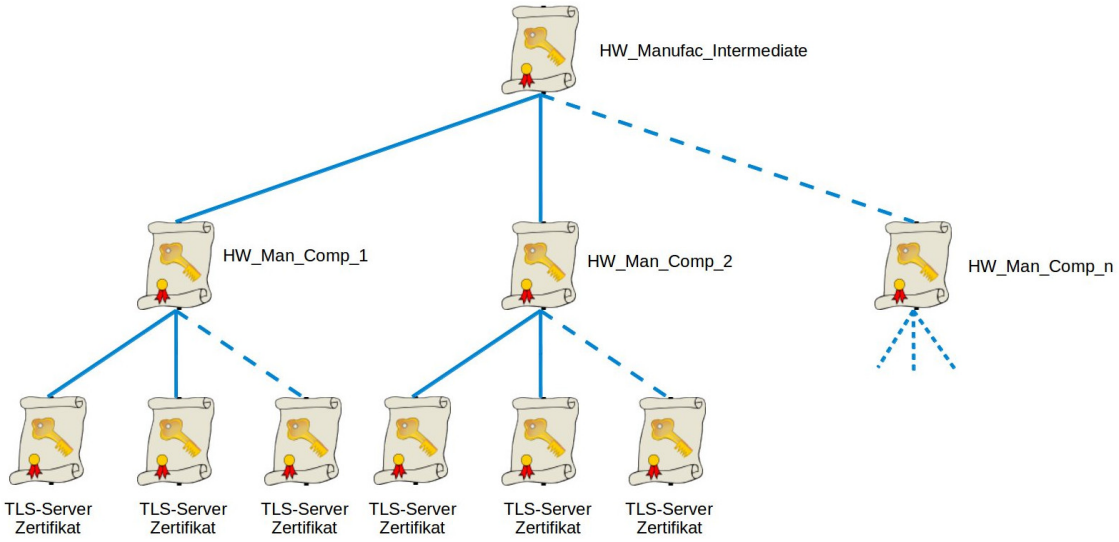

Abbildung 4.1: Teilstruktur der Hardware Manufacturer SubCA

Funktionstest unterzogen, dabei wird unter anderen auch die verschlüsselte Verbindung getestet. Für das Installieren der Software respektive des Betriebssystemes, sowie für die anschließend notwendigen Test und für die Qualitätssicherung werden den Hardware Herstellern eigenen Micro-PC, welche diese Prozesse weitest gehend automatisiert abwickeln, zur Verfügung gestellt [Miced].

Für jeden per Vertrag verpflichteten Hardware Hersteller wird eine eigene Hardware-Manufacturer SubCA (HW_Man_Comp_n) erstellt, sodass es in der Zertifizierungsstelle immer gleich viele gültige Hardware-Manufacturer SubCAs gibt, als physikalische Hardware Hersteller unter Vertrag stehen. Diese SubCAs sind nur berechtigt, Zertifikate an EEs auszustellen, in diesem Fall ist der Enduser ein Webserver, welcher die TLS verschlüsselte Verbindung am embedded Device bereitstellt. Um diese TLS-Server-Zertifikate nachvollziehbar einem embedded Device zuordnen zu können, wird eine einzigartige ID, zum Beispiel der Seriennummer der Hardware, im DN des Zertifikates verankert.

Da die Herstellung der embedded Devices auf Grund dieses Modelles rund um den Globus verteilt sein kann, wurden nach einer Möglichkeiten der Verteilung der Zertifikate gesucht.

Die technisch zu bevorzugende Variante wäre der Einsatz des Simple Certificate Enrollment Protocol (SCEP). Das Protokoll wurde von VeriSign und Cisco entwickelt, um Zertifikate auf Netzwerkgeräten einfach verteilen zu können. Eine erweiterte

Form dieses Protokolls wird bei iOS im Mobile Device Management, zum Beispiel in iPhones oder iPads, eingesetzt. Im Wesentlichen besitzt jedes Gerät ein vom SCEP-Server generiertes One-Time-Passwort, welches zur Authentifizierung dient und in einem SCEP-Request eingebunden, an die PKI gesendet wird. Kann das in der PKI hinterlegte Password bestätigt werden, wird das Zertifikat an das Endgerät ausgeliefert.

Eine Vorgabe an die Verteilung der Zertifikate war es jedoch, dass davon ausgegangen werden muss, dass eine Netzverbindung zu den Hardware-Herstellen oft auch mehrere Tage bis Wochen offline sein könte. Eine Verteilung der Zertifikate mittels SCEP setzt jedoch eine aufrechte Netzwerkverbindung voraus, sodass eine Unterbrechung dieser unter Umständen Lieferschwierigkeiten zur Folge haben könnte.

Für die Verteilung der Zertifikaten wurden somit nachstehende zwei Alternativen in Betracht gezogen:

- Das Exportieren und Auslagern einer kompletten Hardware-Manufacturer SubCA und
- die Entwicklung eines Bestellprozesses inklusive einer gesicherten Verteilung von Zertifikaten und deren privaten Schlüssel

Beide nachstehend näher beschriebenen Varianten sind nicht optimal, da in jedem Fall eine Kontrolle über die ausgestellten respektive eingesetzten Zertifikate verloren geht.

Einfach zum Umsetzen, jedoch mit einem höheren Kontrollverlust behaftet, ist die erste Variante. Dabei wird das Zertifikat einer Hardware-Manufacturer SubCA inklusive ihres privaten Schlüssels und der CA-Chain, komplett aus der EJBCA exportiert und auf einen sicheren Kanal, dem Hardware-Hersteller übergeben. Dieser ist somit berechtigt, eine unbegrenzte Anzahl an Zertifikaten zu erstellen, um sie dann auf die embedded Geräte aufzubringen. In der betriebsinternen Zertifizierungsstelle bleibt nur mehr die Möglichkeit, die komplette SubCA zu widerrufen. Die vom Hardware-Hersteller ausgestellten Zertifikate müssen im Nachhinein wieder importiert werden, um ein Mindestmaß an Verwaltung und Übersicht zu waren.
Diese Variante wird sich hauptsächlich für jene Gerätehersteller eignen, denen man vollständig vertraut bzw. mit welchen bereits über Jahre eine Geschäftsbeziehung besteht.

Die Implementierung eines Bestellprozesses ist aufwendiger, bietet jedoch den Vorteil, dass eine bessere Übersicht und Verwaltung über die ausgestellten Zertifikate erhalten bleibt. In diesem Fall bekommt der Gerätehersteller immer nur eine variable Anzahl an Zertifikaten und privaten Schlüssel, inklusive einer Kontrolldatei auf einem sicheren Kanal übermittelt. Die Kontrolldatei, derzeit im XML-Format,

enthält unter anderem die Seriennummer der Zertifikate und die einzigartigen IDs der embedded Devices, sodass nach dem Herstellungsprozess ein eindeutiges Zertifikat einem eindeutigen Gerät zugewiesen werden kann. Nachdem der Kunde das Gerät erhalten hat bzw. nach erfolgter Bezahlung, kann davon ausgegangen werden, dass die Zertifikate entsprechend der Kontrollinformationen, im Einsatz sind. Ausgestellte und nicht verwendete Zertifikate können auf diese Art und Weise in der PKI temporär oder für immer widerrufen werden. In jedem Fall behält man so ein Mindestmaß an Übersicht in der Zertifizierungsstelle.

4.2.2 HAB Intermediate SubCA

Neben der Notwendigkeit zum Verwalten von TLS-Server-Zertifikaten war bereits in einer frühen Phase des Projektes klar, dass eine weitere Zertifikatsinfrastruktur für die Verwendung des High Assurance Boot (HAB) benötigt wird. HAB ist ein weiteres Security Konzept für embedded Devices, wodurch es möglich ist, ein signiertes Uboot-Image und darauf folgenden ein ebenfalls signiertes Linux Kernel Image zu laden. Mit Hilfe dieser Methode lassen sich CRLs für die TLS-Zertifikate in regelmäßigen Wartungsintervallen sicher auf die embedded Devices verteilen [Mat13].

Die „HAB Intermediate SubCA“ ist berechtigt, Zertifikate für weitere SubCAs auszustellen. Ein solches Zertifikat bzw. die SubCA wird aus der PKI exportiert und dem HAB-Tool als Root-Zertifikat zur Erzeugung der HAB-Zertifikatsinfrastruktur mitgegeben.
Abbildung 4.2 zeigt die Teilstruktur der „HAB Intermediate SubCA“ dabei wird die „HAB_SubCA_n“ respektive ihr Zertifikat exportiert.

4.2.3 SSH Maintenance SubCA

Im weiteren Verlauf des Projektes wurde ein Wartungszugang für Techniker für das embedded Device gefordert. Dabei fiel die Entscheidung auf einen Secure Shell Zugang. SSH bietet zum Authentifizieren der User per default zum einen die klassische Möglichkeit sich mit Username und Passwort anzumelden, zum anderen ist es auch möglich sich mit einem asymmetrischen Schlüsselpaar zu authentifizieren. Sollen eine Vielzahl, auf diese Weise abgesicherten SSH-Accounts, verwaltet werden, besteht die Notwendigkeit darin weitere Software zum Verwalten und Schützen der benötigten Secrets zu verwenden.
Mit Hilfe eines bereits seit mehreren Jahren gepflegten Patch für SSH, ist es möglich, dass sich User an einem SSH-Server zusätzlich mit X.509-Zertifikaten anmelden können [And14].

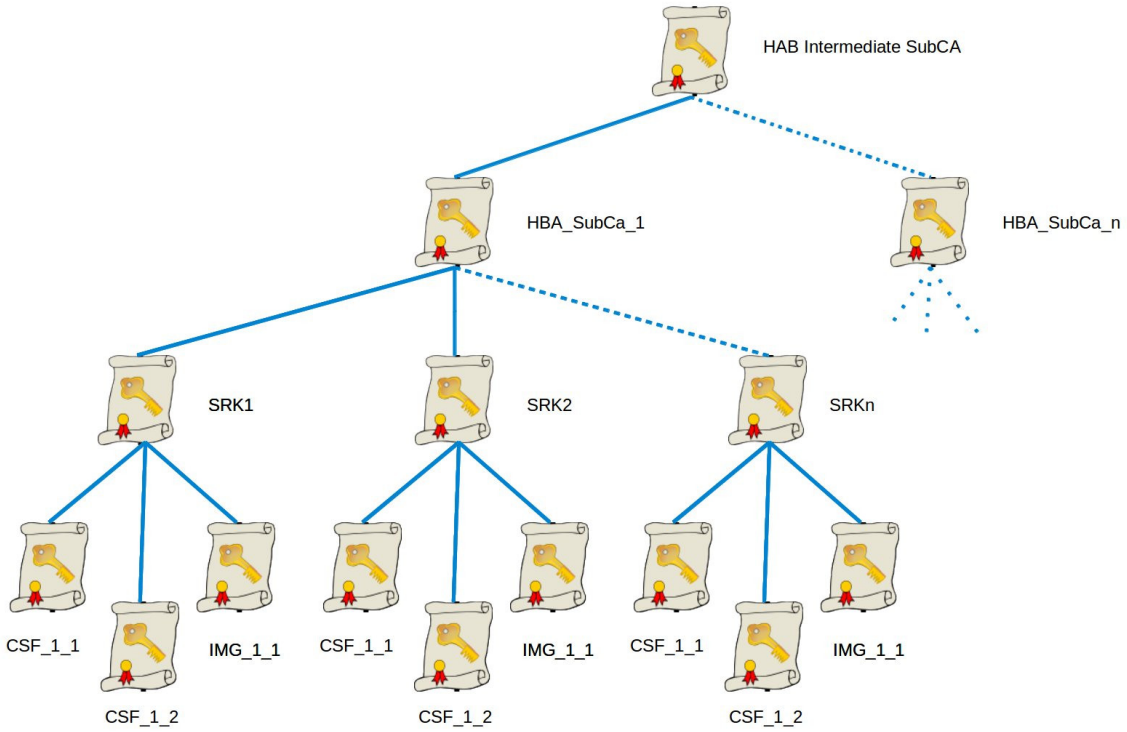

Abbildung 4.2: Teilstruktur der HAB Intermediate SubCA

Die dazu benötigten Zertifikate werden innerhalb der „SSH maintenance SubCA“ verwaltet.

4.2.4 Management Intermediate SubCA

Die durch die „Management Intermediate“ SubCA ausgebildete Teilstruktur der PKI wird benötigt, um infrastrukturelle Services und User der Firma zu zertifizieren. Dies können beispielsweise Mail oder Webserver sein, sowie in diesem Zusammenhang benötigte Client-Zertifikate. Aktuell wird die „Management_Usr“ SubCAgenutzt um Client-Zertifikate für die Administratoren der EJBCA auszustellen. Von der „Management_Srv“ SubCA wird das TLS-Zertifikat des Webservers der Zertifizierungsstelle signiert.
Als zwischengeschaltete CA, dient „Management Intermediate SubCA“ der Strukturierung und ist somit nicht berechtigt Zertifikate direkt an EEs auszustellen.
Abbildung 4.3 zeigt diese Teilstruktur der PKI inklusive der ausgestellten Zertifikate für die PKI-Administratoren, sowie dem Webserverzertifikat.

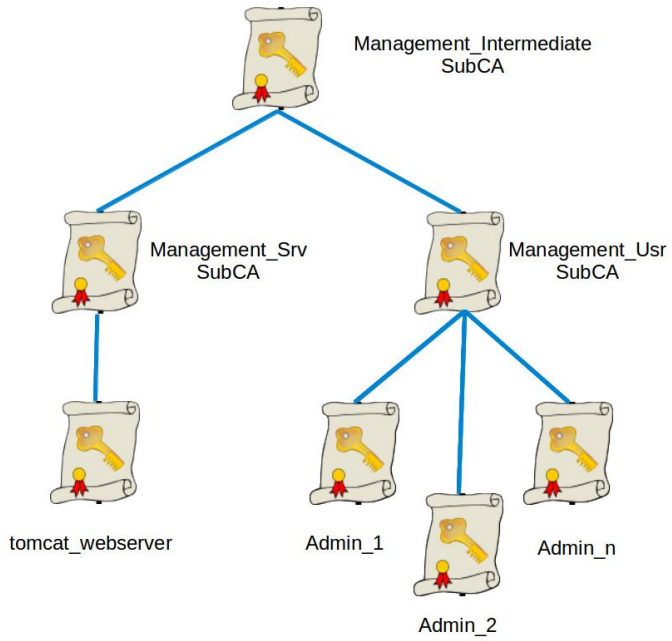

Abbildung 4.3: Teilstruktur der Management Intermediate SubCA

4.3 Use-Case 3: Zertifikatssperrung

Eine Zertifizierungsstelle ist neben der Prüfung der Identität und dem Ausstellen von Zertifikaten auch dafür verantwortlich, dass in bestimmten Fällen bereits ausgestellte Zertifikate widerrufen werden können.
Die Gründe für das Sperren von Zertifikaten können vielfältig sein. Nachstehend werden die für diese Projekt wichtigsten Fälle, in welchen Zertifikate widerrufen werden müssen, besprochen.

Gesperrte Zertifikate müssen veröffentlicht werden. Dazu können einerseits Validierungsdienste verwendet werden oder es werden sogenannten Sperrlisten (CRLs) veröffentlicht. Die Vor- und Nachteile beider Techniken werden im Zusammenhang mit den Vorgaben an dieses Projekt diskutiert. Abschließend wird die Umsetzung der Zertifikatssperrung anhand der einzelnen Teilstrukturen der PKI definiert.

4.3.1 Regeln für Zertifikatssperrungen

Im Wesentlichen wurden die nachstehenden drei Gründe ausfindig gemacht:

- Kompromittierung
- Servicetechniker verlässt die Firma
- Hardwarehersteller ist nicht mehr vertrauenswürdig oder Vertrag läuft aus

Kompromittierung: Ein X.509 Zertifikat besteht zum einen aus der Zertifikatsdatei selbst, welche neben dem öffentlichen Schlüssel und den Identifikationsmerkmalen des Inhabers und des Ausstellers des Zertifikates weitere Systemrelevante Informationen enthält und zum anderen gehört zu jedem X.509 Zertifikat der zum öffentlichen Schlüssel korrespondierende private Schlüssel, welcher gesichert aufbewahrt werden muss.
Geht dieser private Schlüssel im besten Fall zum Beispiel durch ein zerstörtes Dateisystem verloren, ist auch das Zertifikat nutzlos. Gerät dieser private Schlüssel jedoch in Hände Dritter, ist das Zertifikat kompromittiert. In beiden Fällen muss in letzter Konsequenz zuerst das betroffenen Zertifikat gesperrt und anschließend ein Neues erstellt werden.
Handelt es sich bei einem kompromittierten Zertifikat lediglich um ein EE-Zertifikat, zum Beispiel eines Users, ist der administrative Aufwand gering. Kommt jedoch der private Schlüssel einer ganzen SubCA oder sogar der Root-CA abhanden, sind die daraus entstehenden Folgen kaum abschätzbar. Ein aktuelles Beispiel für ein solches Szenario ist der Heartbleed-Bug[1] (CVE-2014-0160) in der TLS-Heartbeat Funktionalität. Dabei konnten bis zu 64KB des Hauptspeichers eines Rechners ausgelesen werden, in welchem sich auch die privaten Schlüssel befinden können. Von diesem Problem betroffen waren alle OpenSSL Versionen von 1.0.1 - 1.0.1f, ausgenommen der Version 1.0.1g.

In diesem Zusammenhang müssen einerseits, für das aufbewahren der öffentlichen Schlüssel Regeln erstellt werden, anderseits ist ständig darauf zu achten, dass die eingesetzte Software keine Schwachstellen enthält.

Mitarbeiter Fluktuation: Dass neue Mitarbeiter in ein Unternehmen eintreten und andere es verlassen ist ein normaler Prozess. Vergleichbar mit physikalischen Zutrittsberechtigungen für die Firma (z.B.: Büroschlüssel) bekommen Mitarbeiter auch elektronische Schlüssel ausgestellt. Im hier vorliegenden Fall werden für die Servicetechniker X.509-Zertifikate erstellt, welche ihnen einen Wartungszugang zu den embedded Systemen gewähren. Verlässt ein Mitarbeiter die Firma oder bezieht er eine andere Postion innerhalb des Unternehmens, müssen Prozesse existieren, welche sicherstellen, dass die ausgestellten Zertifikate dementsprechend zurückgesetzt werden.

[1] http://heartbleed.com/ (Aufgerufen Mai 2014)

Hardware Hersteller: Im Bezug auf die Hardware-Manufacturer SubCA wurden nachstehende mögliche Problematiken erkannt:

- Geräte-Hersteller ist nicht mehr vertrauenswürdig
- Vertrag mit Hardware-Hersteller läuft aus

Im Falle eines Vertrauensverlustes gegenüber einem Hardware-Hersteller ist es in letzter Konsequenz notwendig, die SubCA selbst und somit alle von ihr ausgestellten Zertifikate dauerhaft zu widerrufen.
Wird der Vertrag mit dem Hardware Hersteller gekündigt, muss in der Zertifizierungsstelle sichergestellt werden, dass mit der entsprechenden SubCA keine neuen Enduser Zertifikate mehr erstellt werden können.

4.3.2 OSCP und CRL

Das Veröffentlichen der Widerrufsinformationen gehört zu den notwendigen Grundfunktionen einer Zertifizierungsstelle. Die unterschiedlichen Gründe für die Sperrung eines Zertifikates lassen sich über die sogenannten „Reason Codes“ abbilden. Sie reichen von „hold“, welches eine temporäre Sperre bedeutet, bis zum endgültigen und nicht mehr umkehrbaren Widerruf („revocation“).

Zum Überprüfen der gesperrten Zertifikate haben sich derzeit folgende Techniken etabliert:

Certificate Revocation List (CRL)s bzw. Zertifikat-Sperrlisten bieten eine Möglichkeit die gesperrten Zertifikate zu veröffentlichen.
Bei einer CRL handelt es sich um eine von der PKI signierten Datei, welche unterschiedlich kodiert sein kann. Im Wesentlichen werden in ihr folgende Informationen gespeichert:

- Versionsnummer (v2 od. v3)
- Erzeuger der CRL
- Algorithmus für die Signatur
- Updatezeitpunkt der Ausstellung dieser CRL
- Updatezeitpunkt der Ausstellung der nächsten CRL
- Liste der zurückgezogenen Zertifikate (Seriennummer, Grund und Zeitpunkt des Widerrufs)

Über das X.509v3-Extension-Feld „CRL distribution points“ wird in jedem Zertifkat der Veröffentlichungsort (URIs) der Sperrlisten ausgewiesen. Dazu dient entweder ein Webverzeichnis oder ein LDAP-Verzeichnis.

Jedes Zertifikat, welches sich nicht in einer CRL befindet, wird somit als gültig betrachtet. In Anbetracht der oft langen Gültigkeitsdauer dieser Sperrlisten werden als bereist kompromittiert erkannte Zertifikate von Clients immer wieder als gültig eingestuft.
Obwohl CRLs für jede SubCA getrennt angelegt werden, erreichen diese Dateien eine nicht zu vernachlässigende Größe. Darüber hinaus muss, wenn auch nur ein Zertifikat einer CA überprüft werden soll, die gesamte Liste geladen und eingelesen werden. Obwohl mit Erweiterungen wie der „Delta CRLs“, welche nur mehr die Änderungen zwischen den neuen Veröffentlichungen speichern, versucht wurde dieser Problematik entgegenzuwirken, erwies sich die Verwendung von CRLs in der Praxis als wenig zufrieden stellend.
Des weiteren ist es nicht möglich zu ermitteln ob ein Zertifikat existiert bzw. ob es überhaupt ausgestellt wurde.

Online Certificate Status Protocol (OCSP) ist ein Online-Validierungsdienst für Zertifikate, welcher versucht, die mit den Sperrlisten auftretenden Probleme zu eliminieren. Das Client Server Protokoll kennt dabei sowohl eine verpflichtende als auch eine optionale Funktionalität und wurde im RFC [SMA+13] veröffentlicht.
In einem OCSP-Request richtet der Client genau eine Statusanfrage für ein bestimmtes Zertifikat an den Server. Der OCSP-Responder schickt in seiner signierten Antwort, die aktuellen Statusinformationen für das angefragten Zertifikat an den Client. Hinsichtlich des Status des Zertifikates kennt das System folgende Aussagen:

- unknow: bedeutet, dass das Zertifikat für das System unbekannt ist
- revoked: bedeutet, dass das Zertifikat aus irgendwelchen Gründen temporär oder für immer gesperrt wurde
- good: bedeutet, dass keinerlei Sperrinformationen über das Zertifikat vorliegen

Der Betrieb eines Validierungsdienstes ist aufwendiger als das Bereitstellen von einfachen Sperrlisten, bringt aber Verbesserungen mit sich. Damit dieser Dienst funktioniert, muss zum einen der OCSP-Responder online verfügbar sein und zum anderen benötigt der Validierungsserver auch permanenten Zugang zur CA respektive der Datenbank, in welcher die Informationen zu den Zertifikaten gespeichert sind. Ein Parallelbetrieb von OCSP und dem veröffentlichen von CRLs ist möglich.

Server-based Certificate Validation Protocol (SCVP) ist ein online Validierungsdienst und eine Erweiterung des OCSP. Darüber hinaus, bietet es zusätzliche Funktionalität um Client-Anwendungen den Zugang zu weiteren PKI-Funktionen respektive des Prüfen von Zertifikaten zu erleichtern.
Das Protokoll und die Erweiterungen dazu sind in den RFCs 5055 und 5276

veröffentlicht. Mit Hilfe von SCVP ist es möglich das komplette Überprüfen des Zertifikates das Ermitteln der Vertrauenskette (certification path) und das Validieren des EE-Zertifikates auszulagern und somit sicherzustellen, dass es zu keinen Fehlern in den Einzelschritten kommt. Dazu stehen zwei Methoden zur Verfügung:

- Delegated Path Validation (DPV)
- Delegated Path Discovery (DPD)

Beim DPD sammelt der Server alle notwendigen Informationen um ein Zertifikat zu überprüfen und schickt diese anschließend dem Client. Dieser wertet wie bisher die Ergebnisse aus. Beim DPV wird auch die Bewertung an den Server ausgelagert, sodass der Client hinsichtlich der Validierung der Zertifikate keinerlei Aufgaben mehr übernimmt. Angesichts der Tatsache, dass bei der Validierung von Zertifikaten auf Seiten der Clients respektive der Applikationen es immer zu Fehlern kommt, kann DPV an dieser Stelle eine Alternative darstellen.
SCVP wird erst von wenigen PKI-Implementierungen unterstützt und hat in der praktischen Anwendung zur Zeit noch keine Verbreitung gefunden.

4.3.3 Umsetzung der Zertifikats-Validierung

Die in diesem Projekt entwickelten embedded Devices werden in Anlagen zur elektrischen Energiegewinnung eingesetzt. Diese Anlagen sind per Vorgabe nicht an ein öffentliches Datennetz angebunden, sodass Online- Validierungsdienste wie das OCSP in Verbindung mit den embedded Devices nicht verwendet werden können. Umgesetzt auf die PKI-Struktur bedeutet dies, dass für nachstehende SubCAs Sperrlisten eingerichtet werden müssen.

- Hardware-Manufacturer SubCAs
- SSH Maintenance SubCA

Für die verbleibende Management SubCA würde der Einsatz des OCSP technisch gesehen sinnvoll sein. Da der Fokus dieser Arbeit auf die Verwendung von Zertifikaten im Bereich von embedded Devices liegt und das Setup eines OCSP-Responders komplex ist, werden auch hier Sperrlisten verwendet.

4.4 Use-Case 4: Rollenbasierte Authentifizierung

In kleineren Strukturen ist es nicht notwendig mehrere Verantwortliche mit dem Ausstellen von Zertifikaten zu beauftragen. Ebenso sollte bei der Einführung eines neuen Systemes dieses so einfach wie möglich gehalten werden. Aus diesen Gründen wird weiterhin mit der Rolle eines sogenannten „Superadministrators“ wie er bei der

Default-Installation der EJBCA angelegt wird, weiter gearbeitet. Dieser User hat uneingeschränkte Rechte für alle Funktionen der PKI. Dabei fordert der Use-Case 2, beschrieben in Kapitel (4.2.4) implizit, dass das bei der Installation ausgestellte Zertifikat des „Superadministrators“ zu tauschen ist.
Da eine rollenbasierte Authentifizierung sowie die Vergabe der Rechte für das Erstellen und Widerrufen von Zertifikaten im Zusammenhang mit den diversen SubCAs in größeren Umgebungen notwendig ist, werden fiktive berechtigte Personen festgelegt, welche exemplarisch mit diversen Rechten ausgestattet werden.
Somit ist es möglich, das Konzept der rollenbasierten Authentifizierung zu demonstrieren und trotzdem mit nur einem User die PKI zu verwalten. Der Grund hierfür ist auch in der Firmenstruktur begründet, in welcher zur Zeit das Konzept zum administrieren der PKI noch fehlt.
Neben dem „Superadministrator“ sollen nachstehende User und Berechtigungen vergeben werden:

- Produktionsleitung, RA-Administrator ist berechtigt Zertifikate für die embedded Devices auszustellen.
- Entwicklungsleitung, CA-Administrator ist berechtigt einzelne Sub-CAs inklusive der privaten Schlüssel zu exportieren.
- Neuer Mitarbeiter, hat keine exekutiven Rechte, kann sich durch beobachten der internen Vorgänge der PKI mit dem System vertraut machen.

4.5 Use-Case 5: Automatische Benachrichtigungen

Zum einen sind automatische Benachrichtigungen an die Inhaber der bereits ausgestellten Zertifikate und an das administrative Personal im Fall eines bevorstehenden Gültigkeitsverlustes der Zertifikate umzusetzen und zum anderen ist es für Enrollment-Prozesse wie beispielsweise dem CSR notwendig, dass die User einer PKI automatisch darüber informiert werden, wenn ihr Zertifikat zum Download bereit steht.

4.6 Use-Case 6: Backup und Recovery

Beides ist nicht nur für ein Produktivsystem unerlässlich. Ein automatisiertes Backup und Recovery sollte zumindest konzeptionell erstellt werden. Auf jeden Fall sind systemrelevante Teile wie die Datenbank oder Konfigurationsdateien, welche von den Voreinstellungen abweichen, zu sichern.

4.7 Use-Case 7: Policies

Für jede PKI müssen Regeln festgelegt werden nach welchen die PKI operiert. Dabei wird im Wesentlichen zwischen einer Zertifizierungsrichtlinie (Certificate Policy - CP) und einer für jede CA erforderliche Erklärung zum Zertifizierungsbetrieb (Certification Practice Statement - CPS) unterschieden. Die entstehenden Dokumente müssen öffentlich zugänglich sein [CFS+03].

Die Policies legen somit klare Regeln für die Ausstellung und für den Umgang mit den Zertifikaten einer PKI fest. Auf ein explizites Ausformulieren dieser Regeln kann verzichtet werden, da es sich hierbei um ein Konzept handelt und noch zu wenig Informationen vorliegen, um diese Dokumente vollständig erstellen zu können.
Implizit ergeben sich einige dieser Regeln durch die in diesem Kapitel ausformulierten Use-Cases.

4.8 Use-Case 8: Umsetzung der im Sicherheitsstandard geforderten Parameter für TLS

Die in dieser Arbeit abzusichernde Client-Server Kommunikation auf Basis des TCP/IP-Protokolls erfüllt nicht vollständig die Anforderungen der MMS Spezifikationen wie sie in IEC 61850 definiert sind [IEC13a]. Da jedoch Teil 3 des Sicherheitsstandards (62351-3) hinsichtlich der Absicherung von TCP/IP basierender Kommunikation zu unklar definiert ist, werden die Vorgaben von IEC 62351-4 [IEC07b] in dieser Arbeit als Referenz herangezogen und so weit es möglich ist auf dieses Projekt angewandt.

Kapitel 5

Implementierung

In diesem Kapitel wird die praktische Umsetzung der vorher definierten Use-Cases beschrieben. Beginnend vom Austauschen der Datenbankschnittstelle über das Verteilen der Zertifikate bis zur Implementierung einer standardisierten Umsetzung einer TLS-Verbindung für die embedded Systeme, sind die Themen logisch aufeinander aufbauend strukturiert. Auf die zu diesen Themen zugehörigen Use-Cases wird an den gegebenen Stellen verwiesen.

Allgemein wird dabei vorausgesetzt, dass eine bereits installierte und konfigurierte Instanz der EJBCA sowie bereits erstellte Zertifikat- und End Entity-Profile existieren. Auf eine ausführliche Beschreibung der Installation wird in dieser Arbeit verzichtet und auf die Homepage[1] des Projektes, welche genügend Informationen enthält, verwiesen. Eine detaillierte Beschreibung des Setups sowie weitere ausführliche Informationen im Umgang mit der EJBCA im Zusammenhang mit diesem Projekt, befindet sich in [Based].

5.1 Austauschbarkeit der Komponenten

Wie im Use-Case 1 gefordert, wird an Hand der Datenbank gezeigt, wie es möglich ist, diverse zum Betrieb der EJBCA benötigten Komponenten zu ersetzen. Durch den Einsatz einer MySQL-Datenbank und einer entsprechenden Konfiguration des Jboss AS werden unterschiedlich konfigurierte und voneinander unabhängige Installationen der EJBCA betrieben.

5.1.1 Anbinden der MySQL Datenbank

Die interne Hypersonic-Datenbank (HSQLDB)[2] wird am besten während der Installation gegen eine der unterstützen Datenbanken ersetzt. Im Fall von MySQL wird

[1] www.ejbca.com/docs/installation.html (Aufgerufen: März 2014)
[2] http://hsqldb.org/ (Aufgerufen: März 2014)

unter Debian/Ubuntu die Datenbank wie folgt installiert:

```
sudo aptitude install mysql-server
```

Nach der Grundinstallation von MySQL und dem Einrichten des „Root-Accounts“ für die Datenbank müssen zumindest die Charaktersets überprüft und gegebenenfalls auf utf8 umgestellt werden. Weitere spezifische Informationen für die Konfiguration aller unterstützen Datenbanken befinden sich in der Datei „ejbca/doc/howto/HOWTO-database.txt“.
Das Einrichten eines graphischen Administrations-Interfaces für MySQL-Datenbanken stellt für ein Produktivsystem ein weiteres Sicherheitsrisiko dar. Um jedoch einen besseren Einblick in die Interna des Systemes zu bekommen wurde für dieses Projekt die Applikation phpMyAdmin verwendet.
Anschließend muss für jede installierte Instanz der EJBCA eine Datenbank angelegt werden, in der die Systemdaten abgelegt werden können. Dem DB-User „ejbca“ werden, abgesichert durch ein Passwort, die vollen Rechte über diese neue Datenbank zugewiesen:

```
mysql> create database ejbca_4_mysql_AS_5;
mysql> grant all privileges on ejbca_4_mysql_AS_5.* to
       'ejbca'@'localhost' identified by 'strengGeheim';
mysql> flush privileges;
```

Für den in Java geschriebenen Application Server müssen die MySQL-Schnittstellentreiber (JDBC connectors) entweder über die Paketverwaltung oder direkt von der MySQL-Homepage[3] heruntergeladen und installiert werden.

Die Einstellungen für die Anbindung der MySQL-Datenbank werden in der Datei „database.properties“ im Konfigurationsverzeichnis der EJBCA vorgenommen.

```
database.driver=com.mysql.jdbc.Driver
database.name=mysql
database.url=jdbc:mysql://127.0.0.1:3306/ejbca_4_mysql_AS_5    \
  ?characterEncoding=UTF-8
database.username=ejbca
database.password=strengGeheim
```

Diese Werte werden automatisch mit Hilfe des Bootstrap-Prozess vom „ant“-Tool in das Konfigurationsverzeichnis des Application-Server kopiert. Auf die selbe Art und Weise können andere Datenbanken wie beispielsweise PostgreSQL, Oracle oder der Microsoft SQL Server verwendet werden.

[3] http://dev.mysql.com/downloads/connector/j/ (Aufgerufen: März 2014)

5.1.2 Konfiguration des Application-Servers

PrimeKey empfiehlt für den Betrieb der EJBCA den JBoss Application Server (AS) zu verwenden. Darüber hinaus werden jedoch weitere Application Server unterstützt. Eine Liste möglicher Alternativen befindet sich auf der Homepage des Projektes[4]. Um eine möglichst einfache Austauschbarkeit des AS im Fall eines Updates gewährleisten zu können, empfiehlt es sich, gleich bei der Grundinstallation gewisse Vorkehrungen zu treffen. So wurde, anders als wie auf der Homepage beschrieben, sowohl der AS als auch die EJBCA nicht in das Home-Verzeichnis eines Users installiert, sondern in das bei Debian/Ubuntu dafür vorgesehene „opt"-Verzeichnis. Anschließend wurde ein symbolischer Link von einem generischen Verzeichnisnamen wie beispielsweise „jboss" auf das Verzeichnis mit der aktuellen Version des AS „jboss-5.1.0.GA" gesetzt.

Da der JBoss AS seine Dienste auch für andere Anwendungen zur Verfügung stellt, ist es sinnvoll, die Konfigurationen für die einzelnen Applikationen, wie zum Beispiel der EJBCA, in einem eigenen Unterverzeichnissen vorzuhalten. Dazu wird im Verzeichnis „/opt/jboss/server/" beispielsweise der Ordner „ejbca-4_mysql_AS_5" erstellt und mit den Standardwerten befüllt:

```
cd /opt/jboss/server/
cp -pr default ejbca-4_mysql_AS_5
```

Abschließend müssen die Startparameter des Jboss AS (jbossHome und jbossConf) sowie die EJBCA Parameter (appserver.home und jboss.config) auf die neuen Gegebenheiten angepasst werden.

Mit Hilfe dieser Methode ist es möglich mehrere von einander unabhängige Installation der EJBCA auf einem System zu betreiben.

5.2 Abbildung der PKI-Struktur

Die im Kapitel 3.1 und im Use-Case 2, Kapittel 4.2 beschriebene Struktur der PKI wird nachfolgend eingerichtet. Eine RootCA sowie weitere SubCAs werden über das Admin-Web wie folgt erstellt:

```
CA Functions -> Certificate Authorities -> Add CA ->
"Name_der_CA" eingeben -> Create
```

In der sich darauffolgend öffnenden Eingabemaske werden die Einstellungen für die jeweilige CA vorgenommen. Der Großteil der abgefragten Werte sind zum einen allgemeine X.509-Zertifikats Parameter und zum anderen X.509v3-Extension-Einstellungen. Darüber hinaus sind folgende EJBCA spezifische Parameter zu beachten:

[4]http://www.ejbca.com/docs/installation.html#Application%20servers (Aufgerufen: März 2014)

- CA Token Type: bestimmt ob die CA in ein HSM ausgelagert werden soll.
- Authentication Code: ist ein Passwort, welches beispielsweise benötigt wird, wenn die CA inklusive ihres privaten Schlüssels aus dem System exportiert werden soll.
- Certificate Profile: ist ein EJBCA spezifisches Profil-System, wie nachstehend beschrieben wird.

Abbildung 5.1 zeigt einen Ausschnitt aus dem Konfigurationsdialog zum Erstellen von CAs.

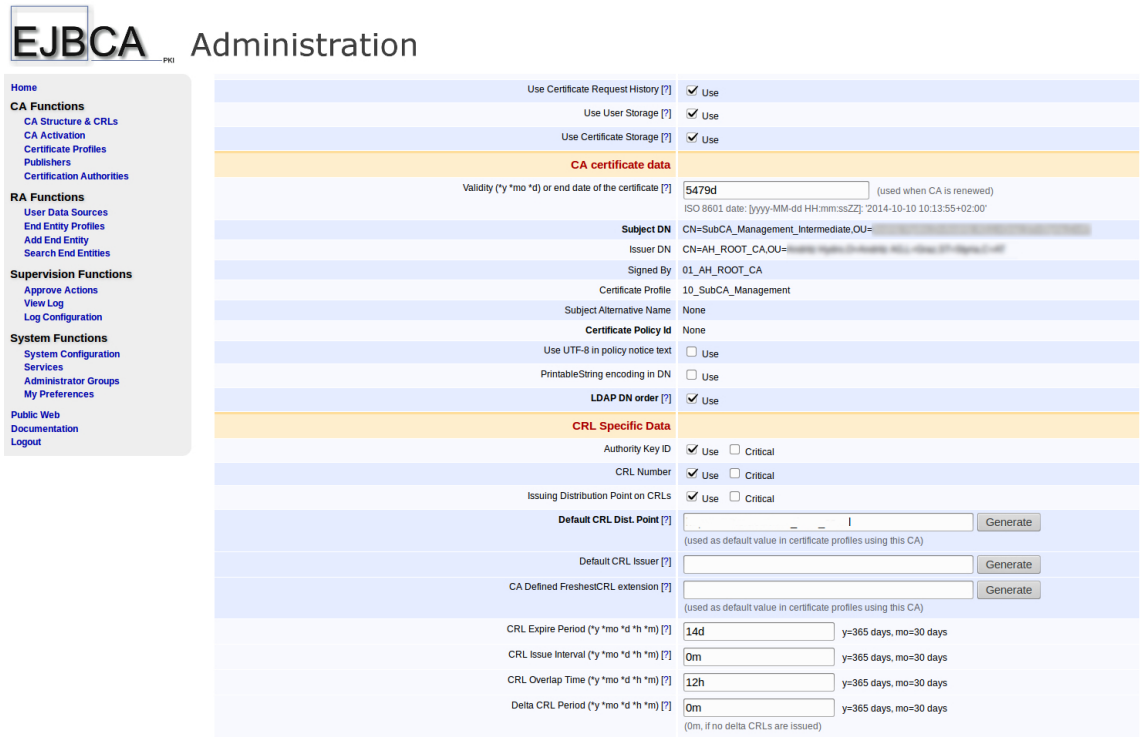

Abbildung 5.1: Webinterface für das Erstellen von CAs

5.2.1 Profil System

Das Ausstellen von Zertifikaten für CAs und EEs erfordert das Vorhandensein von sogenannten Profilen. Dabei benötigen CAs sogenannte Zertifikats-Profile und EEs darüber hinaus noch ein End Entity Profil. Vorkonfigurierte Zertifikat-Profile sowohl für die Root- und SubCAs als auch für die EE wurden während der Installation eingerichtet. End Entity Profile müssen erst vom Administrator erstellt werden. Allgemeine Informationen zu den Profilen befinden sich auf der Homepage von PrimeKey,

sowie für dieses Projekt spezifische Informationen in [Based].

Da für ein erstellen von Root- und SubCAs die bei der Installation automatisch angelegten Profile verwendet werden können, wird an dieser Stelle nicht weiter auf das Anfertigen von Zertifikat-Profilen eingegangen. End Entity Profile werden etwas später in diesem Kapitel beschrieben bzw. befinden sich die dafür notwendigen Einstellungen im Anhang B.1.

5.2.2 Erstellen der Root CA

Für die oberste Zertifizierungsstelle in der PKI-Hierarchie wird der Name „01_AH_ROOT_CA“ vergeben. Ausschließlich für den Aufbau der PKI-Struktur relevant sind nachstehende, in der Sektion „CA certificate data“ zu findende Parameter:

- Signed By: Self Signed
- Certificate Profile: Root CA

„Signed By“ legt den sogenannten „Issuer“ des CA-Zertifikates fest. Mit Hilfe dieses Parameters ist die Struktur der PKI abzubilden, sodass jede untergeordnete CA im Baum (Knoten) nur von der ihr direkt übergeordneten signiert werden darf.
Bei einer selbst signierten RootCA, wie sie in diesem Projekt verwendet wird, ist der Parameter „Signed By“ somit auf „Self Signed“ zu setzen. Diese Einstellung schränkt dabei automatisch die zur Verfügung stehenden Zertifikat-Profile auf das einer Root CA ein.

5.2.3 Erstellen der Intermediate SubCAs

Alle Intermediate SubCAs wie beispielsweise die Management Intermediate SubCA, werden direkt von der RootCA signiert, sodass der Parameter „Signed By“ in diesem Fall auf „01_AH_ROOT_CA“ zu stellen ist. Als Zertifikat-Profil ist hierbei entweder das vorinstallierte „SUBCA“ oder ein bereits selbst erstelltes spezialisierteres Profil zu wählen. Für die interne Namensgebung wurde für alles CAs die nachstehende Form gewählt:

```
\d{2}_CA-Type_CaName
```

Sodass die Management Intermediate SubCA die folgende interne Repräsentation erhält: 10_SubCA_Management_Intermediate. Jede Intermediate SubCA beginnt dabei bei einer neuen Zehnerstelle. Die ihnen untergeordneten SubCAs werden fortlaufend durchnummeriert. Diese Notation dient in erster Linie der Übersicht in der PKI und wird ausschließlich für die interne Repräsentation in der EJBCA genutzt. Für den „Subject-CN“ müssen andere den Konventionen entsprechende Namen verwendet werden.

5.2.4 Verbleibende SubCAs

Das Erstellen der weiteren SubCAs wird exemplarisch am Beispiel der „11_SubCA_Management_Srv“ gezeigt und wurde analog dazu für die verbleibenden im System befindlichen SubCAs durchgeführt.
Über die Nummerierung zu erkennen, ist die „11_SubCA_Management_Srv“ dem SubTree der Management-CA untergeordnet. Somit ist der „Signed By“-Parameter auf „10_SubCA_Management_Intermediate“ zu stellen.

5.3 Ausstellen und Exportieren von Zertifikaten

Innerhalb der EJBCA wird prinzipiell unterschieden ob, ein CA- oder ein EE-Zertifikat bearbeitet werden soll, sodass im Umgang mit CA-Zertifikaten CA- Funktionalitäten und Rechte und in der Handhabung von EE-Zertifikaten RA-Funktionalitäten und die damit verbundenen Rechte benötigt werden. Durch den Einsatz eines Superadministrators mit allumfassenden Rechten ist es nicht notwendig, diesbezüglich eigene Rollen zu vergeben.

Weiters wird sowohl bei CA als auch bei EE-Zertifikaten unterschieden, ob ein Zertifikat mit oder ohne des dazu korrespondierenden privaten Schlüssels exportiert respektive ausgestellt werden soll.
Nachstehend beschrieben sind nur jene Fälle, die für die Umsetzung der Use-Cases notwendig sind.

5.3.1 Exportieren von CA-Zertifikaten

Zertifikate für die im System befindlichen CAs werden zum Zeitpunkt des Anlegens der Zertifizierungsstelle erzeugt. Nachstehend beschrieben ist, wie ein Zertifikat mit und ohne dem korrespondierenden privaten Schlüssel aus der EJBCA exportiert werden kann.

Export ohne des privaten Schlüssels

Für die Umsetzung der rollenbasierten Authentifizierung (Use-Case 4) wird das Zertifikat der „11_SubCA_Management_Usr“ benötigt.

CA-Zertifikate können sowohl über das Admin-Web (CA Functions -> CA Structure & CRLs) als auch wie nachstehend beschrieben, über das Command Line Interface (CLI) exportiert werden:

```
cd /opt/ejbca
./bin/ejbca.sh ca listcas
```

Der „listcas“-Befehl gibt eine Liste mit Basisinformationen zu allen im System befindlichen CAs aus. Darunter auch die interne Repräsentation der CAs (CA Name).

```
./bin/ejbca.sh ca  getrootcert <CA Name> CaName.pem
openssl x509 -in CaName.pem -out  CaName.der -outform DER
```

Mit Hilfe von „getrootcert“ wird das Zertifikat der angegebenen CA (CA Name) im PEM-Format in die Datei „CaName.pem“ exportiert. Optional erfolgt eine Überführung des Zertifikates von der PEM- in die DER- Form mit Hilfe von OpenSSL.

Das Zertifikat der „11_SubCA_Management_Usr“ wird im DER-Format benötigt, um einen sogenannten „truststore“ auszubilden. In diesem, mit einem Passwort geschützten Container, werden CA-Zertifikate gesammelt, die berechtigt sind, EE-Zertifikate auszustellen, mit denen es wiederum möglich ist, sich als Administrator am Admin-Web der EJBCA anzumelden.

Export im PKCS #12-Format

Sogenannte „Soft token CAs“ also CAs, die nicht in ein HSM ausgelagert wurden, lassen sich inklusive ihres privaten Schlüssels exportieren. Dies kann zum eine für Backup-Zwecke verwendet werden und zum anderen wird diese Funktionalität genutzt um Zertifizierungsstellen für das High Assurance Boot (HAB) und der Hardwarehersteller zu exportieren, um sie dann außerhalb der EJBCA weiter verwenden zu können (Use-Case 2).

Diese Funktionalität steht wiederum im Admin-Web und CLI zur Verfügung. Für das Exportieren von CAs inklusive der privaten Schlüssel im PKCS #12-Format über das Admin-Web, benötigt es einen User mit Superadministratoren Rechten:

```
CA Functions -> Certificate Authorities -> select CA -> Edit CA
```

In der sich öffnenden Maske befindet sich am unteren Ende der Seite in der Sektion „CA life cycle“ das Eingabefeld mit dem Namen „CA export requires the token authentication code“. Der benötigte „authentication code“ wurde beim Erstellen der CA vergeben vgl.:5.2. Mit einem Click auf „Export CA keystore“ wird das PKCS #12-File über den Browser am lokalen Rechner gespeichert.

Ein automatisierter Export kann über das CLI wie folgt eingerichtet werden:

```
 cd /opt/ejbca
./bin/ejbca.sh ca listcas
./bin/ejbca.sh ca exportca <CA name> rel/path/to/bakdir/CaName.p12
```

Der Befehl „exportca“ ist somit syntaktisch gleich zu verwenden wie „getrootcert“. Die CA wird dabei in einen PKCS #12-Container gespeichert, sodass neben dem privaten Schlüssel und dem Zertifikat der CA im Falle von SubCAs auch die sogenannte CA-Chain, also alle Zertifikate in der PKI-Hierarchie aufsteigend bis zur Root-CA mit exportiert werden. Das selbe gilt auch für den Export über das Admin-Web.

Ein Umwandeln in ein anderes Format oder das Extrahieren einzelner Teile aus dem p12-File wird am besten mit dem „openssl pkcs12“-Tool bewerkstelligt.

Bevor die CA in Form der p12-Datei an einen Hardware Hersteller weiter gegeben wird, sollte sie beispielsweise zusätzlich AES verschlüsselt werden.

5.3.2 Zertifikate für EEs

Das Ausstellen von Zertifikaten für EEs ist ein zweistufiger Prozess. Zuerst muss der User im System angelegt werden, dies ist eine Funktionalität der RA, anschließend wird das Zertifikat ausgestellt.

Um ein Zertifikat für EEs ausstellen zu können wird neben einem Zertifikat-Profil noch ein sogenanntes End Entity Profil benötigt, in dem rein User bezogene Daten gespeichert werden. Exemplarisch für die TLS-Webserver Zertifikate befinden sich Informationen zum Erstellen dieser beiden Profile im Anhang B.1. Weitere Informationen zum Erstellen von Profilen befinden sich auf der Homepage[5] der EJBCA sowie für dieses Projekt spezifisch in [Based].

Für das Ausstellen von EE-Zertifikaten stehen diverse Möglichkeiten zur Verfügung. Abbildung 5.2 zeigt exemplarisch das Erstellen von EE-Zertifikaten mit Hilfe des Admin-Web.

Automatisiertes Ausstellen von TLS-Server-Zertifikaten

Ein automatisches Erstellen von EE-Zertifikaten für den Webserver auf den embedded Devices ist in Use-Case 2 definiert.

Das Anlegen von Usern respektive das Ausstellen von Zertifikaten ist über das CLI leicht automatisierbar. In einem ersten Schritt wird mit Hilfe der beiden zuvor erstellen Profile 61_Bord_SSL_Server und
61_Bord_SSL_Server_Batch_p12 (B.1) der User im System aufgenommen:

```
cd /opt/ejbca
./bin/ejbca.sh ra adduser username password dn subjectAltName caname
email type token certificateprofile endentityprofile
```

Während die Parameter „username“, „password“ und „email“ frei vergeben werden können, muss die Struktur von „dn“ und „subjectAltName“ den Einstellungen in den Profilen entsprechen. Die verbleibenden Parameter wurden entsprechend des Einsatzzweckes des Zertifikates gewählt, sodass nachstehender Befehl den neuen User anlegt:

[5] www.ejbca.com/docs/installation.html

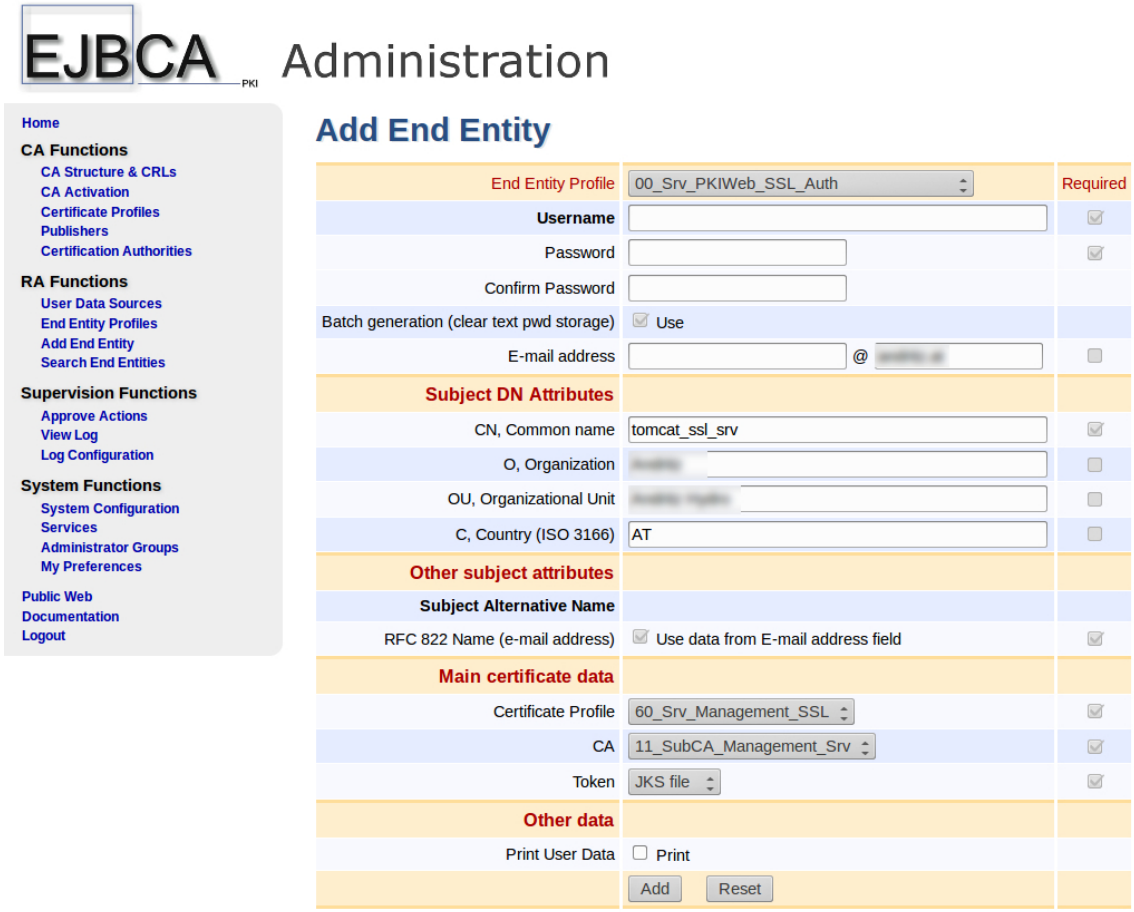

Abbildung 5.2: Webinterface für das Erstellen von EE-Zertifikaten

```
./bin/ejbca.sh ra adduser ssl_board_srv_1 passwd
"CN=ssl_board_srv_1,O=myCompany,C=AT" "ipaddress=10.0.0.123"
21_SSubCA_HW_Man_Compxy null 1 P12 61_Bord_SSL_Server
61_Bord_SSL_Server_Batch_p12
```

Da das Zertifikat für die End Entity (EE) per CLI erzeugt werden soll, muss das User-Passwort in der Datenbank im Klartext gespeichert werden, was mit nachstehendem Befehl erreicht wird:

```
./bin/ejbca.sh ra setclearpwd ssl_board_srv_1 passwd
```

Dabei müssen die beiden Passwörter nicht zwingend übereinstimmen. Nachdem der User im System angelegt ist, wird das Zertifikat schlussendlich mit dem nachstehenden Befehl erzeugt:

```
./bin/ejbca.sh batch ssl_board_srv_1 -dir p12/SSL-Server
```

Existiert der Ordner „SSL-Server“, werden die Zertifikate dort im PKCS #12-Format gespeichert.

Alle drei für das Erstellen der EEs notwendigen Befehle lassen sich per Skript verarbeiten. Als Datenquelle für die User bezogenen Daten können beispielsweise XML- oder JSON- Datenstrukturen verwendet werden, die im Zuge des Bestellprozesses generiert werden.

Automatisiertes Ausstellen von Zertifikaten per CSR

Am Beispiel der Zertifikate für die Absicherung des SSH-Zuganges soll nachstehend eine mögliche Methode gezeigt werden, wie ein Mitarbeiter ein solches Zertifikat ausgestellt bekommen kann (Use-Case 2 und 4).

Da der Mitarbeiter sein Zertifikat, in diesem Beispiel ausschließlich mittels CSR ausgestellt bekommen soll, müssen im End Entity Profile die Parameter „Default Token“ und „Available Tokens“ auf „User Generated“ gesetzt werden.
Das Anlegen des Users im System erfolgt in diesem Beispiel über das Admin-Web durch einen RA-Administrator, welcher zuvor über das Rollenmanagement der EJBCA eingerichtet werden muss [Based]. Ein solcher RA-Administrator könnte der Personalchef sein, welcher ausschließlich dazu berechtigt ist SSH-Zertifikate auszustellen.

Die Vorgehensweise dabei ist wie folgt:
Zuerst wird über das Admin-Web der User angelegt:

```
RA Functions -> Add End Entity
```

Das vorbereitete End Entity Profil muss selektiert und ausgefüllt werden. Der User wurde dem System hinzugefügt und erhält den Status „NEW“. Username und Passwort werden anschließend dem Mitarbeiter übergeben. Dieser erstellt einen CSR, beispielsweise mit OpenSSL folgendermaßen:

```
openssl req -nodes -new -newkey rsa:1024 -out csr.pem
```

Wichtig dabei ist, dass die Keylänge mit denen des Zertifikat-Profils übereinstimmt. Alle anderen Werte bzw. Abfragen beim Erstellen des CSR können auf den Voreinstellungen belassen werden. Sie werden beim Ausstellen bzw. beim Signieren des Zertifikates von der EJBCA gemäß den Profileinstellungen überschrieben. Mit Hilfe des CSR und des Public Web kann der Mitarbeiter sich sein signiertes Zertifikat wie folgt ausstellen lassen:

```
Enroll -> Certificate enrollment from a CSR
```

Das vom RA-Administrator erhaltene Password sowie der Username dienen ausschließlich der Authentifizierung der Users an der PKI. Der CSR kann entweder direkt in das dafür vorgesehene Textfeld kopiert oder per Datei hochgeladen werden.

Anschließend wird der User aufgefordert das Zertifikat abzuspeichern. Das Zertifikat ist jetzt zusammen mit dem beim CSR erzeugten privaten Schlüssel einsatzbereit. Nachdem der Enrollment-Prozess abgeschlossen ist, bekommt der User im System den Status „Generated“.

Abbildung 5.3 zeigt das öffentlich zu erreichende Webinterface zum Anfordern eines Zertifikates per CSR.

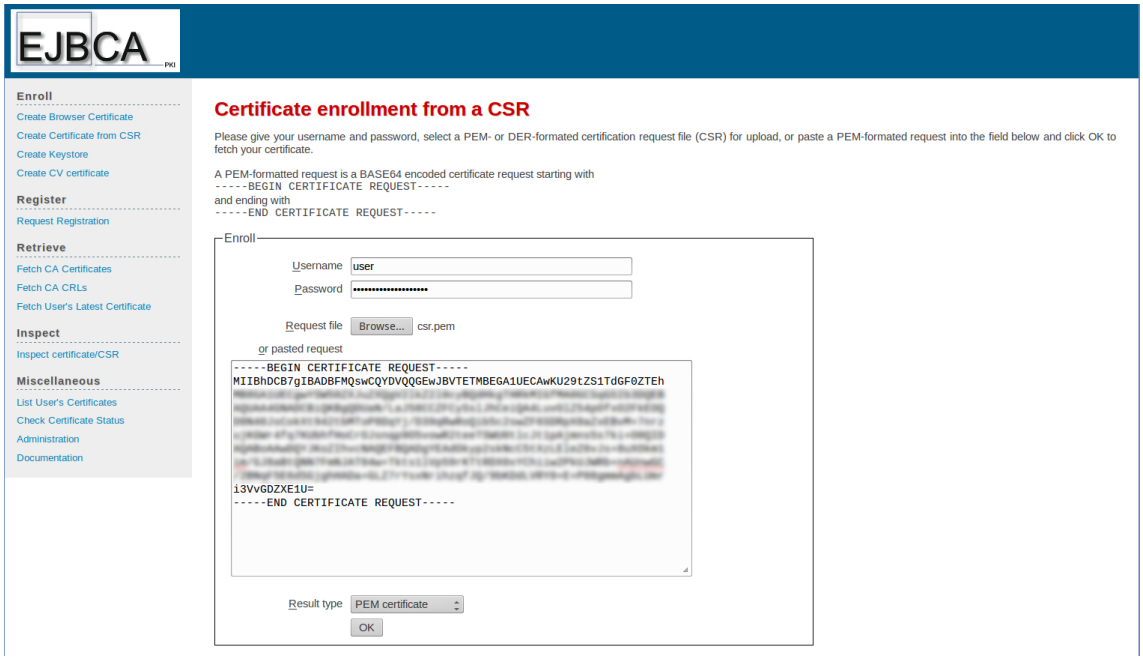

Abbildung 5.3: Zertifikat-Enrollment-Prozess per CSR

5.4 Widerrufen von CA und EE Zertifikaten

Das Widerrufen von Zertifikaten und das Veröffentlichen von Sperrlisten (CRLs) wird von den Use-Cases 2 und 3 gefordert.
Nachstehend wird zuerst das automatische Erstellen und Publizieren der Sperrlisten beschrieben. Anschließend wird über die Verwendung der Suchfunktion, beschrieben wie Zertifikate widerrufen werden können.

5.5 Erstellen und publizieren der CRLs

Um ein automatisches Erstellen der Sperrlisten einzurichten, ist es notwendig, dass die CAs dahingehend richtig konfiguriert werden. Eine detaillierte Beschreibung da-

zu befindet sich in der Dokumentation der EJBCA[6]. Eine an die Erfordernisse dieses Projektes angepasste Beschreibung befindet sich in [Based].

Sperrlisten lassen sich zumindest über zwei Methoden automatisch erstellen:

- Update Service für CRLs der EJBCA
- CLI-Befehl und Cron Job

Das CRL Update Service wird über das Admin-Web wie folgt eingerichtet:

```
System Functions -> Services -> Manage Services
```

Für das neue Service muss ein Name gefunden und über „Add“ dem System hinzugefügt werden. Anschießend werden für das neu erstellte Service die nachstehenden Parameter vergeben:

```
Select Worker: CRL Updater
CAs to Check: ALL
Active: check
Description: a usefull Description
```

Nach dem das Update Service für die CRLs gespeichert wurde, sollte es in der Liste als aktiv geführt werden. Wurden die restlichen Werte auf den Voreinstellungen belassen, so wird jetzt alle fünf Minuten geprüft ob eine CRL neu zu erstellen ist.

Mit Hilfe eines Cron Jobs und dem nachstehenden CLI-Befehls lässt sich diese Aufgabe ebenfalls lösen:

```
 cd /opt/ejbca
./bin/ejbca.sh ca createcrl
```

Abbildung 5.4 zeigt die Konfiguration des CRL-Updateservices im Admin-Web der EJBCA.

5.5.1 Veröffentlichen und Exportieren der CRLs

CRLs können über das Admin-Web und den Browser wie folgt exportiert bzw. lokal gespeichert werden:

```
CA Functions -> CA Structure & CRLs
```

Ein Script gesteuertes, automatisches Exportieren ist mit Hilfe des nachstehenden CLI-Befehls möglich:

[6] http://www.ejbca.com/docs/index.html (Aufgerufen: März 2014)

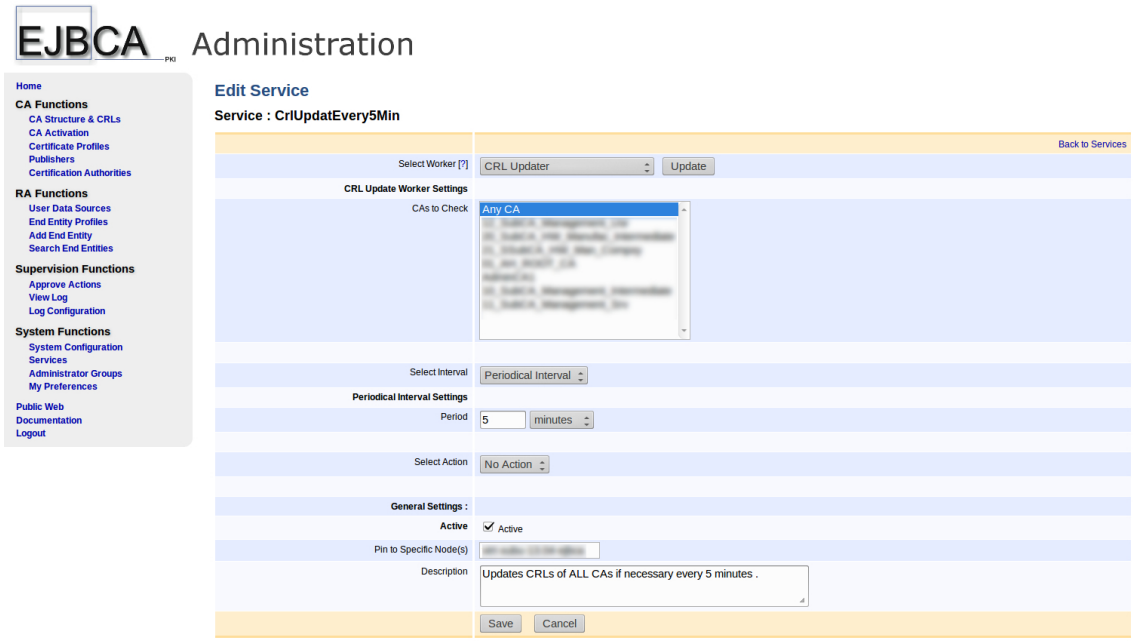

Abbildung 5.4: Konfiguration des CRL-Updateservices

```
cd /opt/ejbca
./bin/ejbca.sh ca getcrl CaName OutFile.crl
```

Die Sperrlisten der angegebenen CA werden dabei in DER-Form exportiert. Dieser Befehl wird mit Hilfe eines Cron Jobs für jede im System befindliche CA ausgeführt. Die weitere Vorgehensweise zum Veröffentlichen der CRLs wird nachfolgend für die einzelnen CAs getrennt beschrieben.

Management Srv u. Usr SubCAs: Für diese CAs werden die Sperrlisten in einem Webverzeichnis veröffentlicht. Dazu werden die CRL-Files automatisiert in das entsprechende Webserververzeichnis kopiert. Da sich, aus bereits genannten Gründen, alle Komponenten der EJBCA auf einem Rechner befinden, wird auch der zum Veröffentlichen der Sperrlisten notwendige Webserver über die Paketverwaltung auf dem selben Rechner installiert.

HAB SubCAs ausgehend vom Use-Case 2 und 3 werden hier keine Sperrlisten benötigt.

Hardware Manufacturer SubCAs und SSH CA: Die Sperrlisten dieser CAs müssen in das Linux-Image für das embedded Device integriert werden. Dazu werden die Dateien automatisiert in ein speziell dafür vorgesehenes Verzeichnis kopiert. Über einen Hook im Buildsystem und mit Hilfe des SCP-Protokolls

werden die Sperrlisten automatisch in das Linux-Image integriert. Am embedded System ist ein lighttpd[7] installiert, welcher die CRLs veröffentlicht.

5.5.2 Widerrufen von Zertifikaten

Um ein bestimmtes Zertifikat widerrufen zu können, muss es zuerst über das Admin-Web ausfindig gemacht werden:

```
RA Functions -> Search End Entities
```

Hier stehen mehrere Möglichkeiten zum Suchen von Zertifikaten zur Verfügung. Das gewünschte Zertifikat muss selektiert werden, um es anschließend mit „Revoke Selected“ zu widerrufen. Optional kann ein spezieller Grund für die Zertifikatssperrung angegeben werden.

5.6 E-Mail Benachrichtigungen

Benachrichtigungen an verantwortliche Personen im Falle eines bevorstehenden Gültigkeitsverlustes der bereits ausgestellten Zertifikate wird im Use-Case 5 gefordert. Weiters verfügt die EJBCA über ein automatisiertes Informationssystem im Umgang mit Zertifikaten [Based].

5.6.1 Benachrichtigung beim Ablauf der Gültigkeit

Um zu ermitteln, welche Zertifikate in den nächsten 100 Tagen ihre Gültigkeit verlieren, steht am CLI der nachstehende Befehl zur Verfügung:

```
cd /opt/ejbca
./bin/ejbca.sh ca listexpired 100
```

Per Cron wird der Befehl über ein Script aufgerufen und der generierte Output automatisch nach abgelaufenen Zertifikaten durchsucht. Das Ergebnis wird per E-Mail an die verantwortlichen Personen geschickt.

5.7 Rollenbasierte Authentifizierung

Für die Umsetzung einer rollenbasierten Authentifizierung, wie sie im Use-Case 4 definiert wurde, sowie für die Einrichtung des Superadministrators, wird die „Management_Usr“ SubCA genutzt. Dazu muss ihr Zertifikat extrahiert und dem Truststore hinzugefügt werden [Based].
In einem zweiten Schritt müssen die Administratoren als User (EEs) im System angelegt werden, um ihnen dann anschließend Zertifikate ausstellen zu können. Dabei

[7] http://www.lighttpd.net/ (Aufgerufen: März 2014)

ist darauf zu achten, dass die Zertifikate von der „Management_Usr“ SubCA signiert werden.
Abschließend müssen die Administratoren mit ihren Rollen verknüpft werden. Dazu wird das Admin-Web aufgerufen. Unter

```
System Functions -> Administrator Groups
```

wird eine neue Gruppe angelegt, der anschließend eine Rolle zugewiesen wird. Nachdem die Gruppe erstellt wurde, können ihr die User über die Seriennummer der Zertifikate zugeordnet werden. Auch hier ist zu beachten, dass in der drop-down List „CA“ wieder die „Management_Usr“ SubCA selektiert wird.

5.8 Backup und Recovery

Eine Backup und Recovery Funktionalität, wie es im Use Case 6 gefordert ist, kann aufgrund des Konzeptes der EJBCA mit einigen wenigen Befehlen auch automatisiert durchgeführt werden.

Zum einen muss die verwendete MySQL-Datenbank, beispielsweise mit dem Tool mysqldump, gesichert werden. Des weiteren müssen die Dateien aus dem Konfigurationsverzeichnis (ejbca/conf), die Key- und Truststores für den Webserver und die Administratoren, in regelmäßigen Abständen gesichert werden.
Darüber hinaus können auch alle selbst erstellten Profile und CAs exportiert werden, sodass es möglich ist, die komplette Instanz wieder herzustellen oder auch einen Teil der PKI auf ein anderes System umzuziehen.

Alle für das Backup benötigten Funktionen lassen sich in einem Script zusammenfassen und per cron in regelmäßigen Abständen aufrufen. Das Recovery wird sinnvollerweise per Hand durchgeführt.

5.9 Konfiguration der embedded Devices

Die in Use-Case 8 geforderte Implementierung der Zertifikate auf dem embedded System wird nachstehend beschrieben.

Da in die bereits existierende proprietäre Client-Server-Software nicht eingegriffen werden kann, wurde der SSL/TLS Warpper Stunnel[8] eingesetzt, um die bisherige unverschlüsselte Kommunikation in eine nach FIPS 140-2 standardisierte Datenübertragung überzuführen [Bas14]. Abbildung 5.5 verdeutlicht die Funktionsweise der Verschlüsselung unter zuhilfenahme von Stunnel.
Das Bereitstellen der CRLs auf den embedded Devices wird von einem schlanken

[8] https://www.stunnel.org/ (Aufgerufen: März 2014)

Websever übernommen. Sowohl die dafür eingesetzte Software (lighttpd) als auch der SSL/TLS Warpper müssen bereits im Linux-Image integriet sein [Bas14].
Im Anhang diesee Arbeit C befindet sich eine Zusammenfassung aller Parameter, welche von den Standardeinstellungen abweichen. Nachstehende Abbildung verdeutlicht die Funktionsweise der Verschlüsselung mit Stunnel

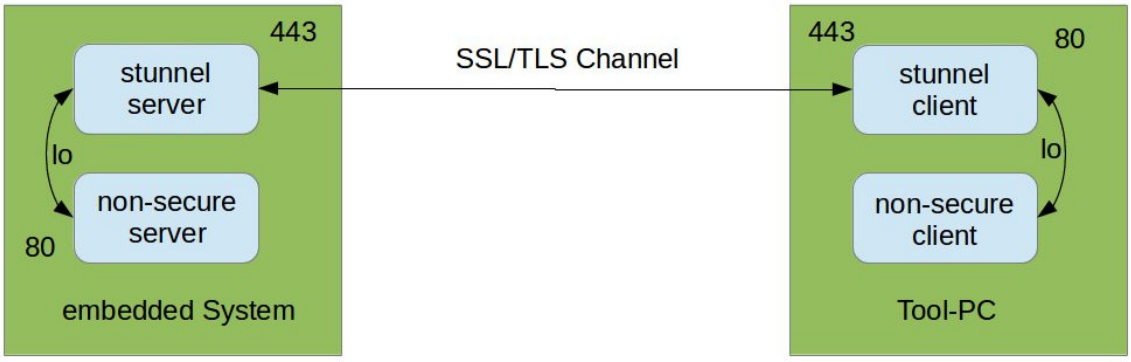

Abbildung 5.5: TLS-Verschlüsselung mit Hilfe von stunnel

5.9.1 Vorbereiten der Zertifikate für stunnel

Bevor das von der EJBCA im PKCS #12-Format ausgestellte TLS-Server-Zertifikat im stunnel verwendet werden kann, müssen einzelne Teile extrahiert und in das PEM-Format übergeführt werden.
Unter dem Konfigurationsparameter „cert =“ erwartet sich der TLS-Wrapper stunnel das Serverzertifikat und den dazu korrespondierenden privaten Schlüssel, zusammengefasst in einer Datei. Der nachstehende OpenSSL-Befehl erzeugt diese Datei:

```
openssl pkcs12 -in tls-server.p12 -passin pass:1234 -clcerts
  -nodes -out cert-key.pem
```

Die CA-Chain für das verwendetet TLS-Server-Zertifikat wird mit Hilfe des Parameters „CAfile =“ dem Server übergeben. Die CA-Chain wird wie nachstehend beschrieben ebenfalls aus der p12-Datei extrahiert:

```
openssl pkcs12 -in tls-server.p12 -passin pass:1234 -cacerts -nokeys
  -nodes -out ca-chain.pem
```

5.9.2 Einstellen der TLS-Parameter

Im IEC 62351 Teil 4 und 6 wird zumindest die Unterstützung der beiden nachstehenden Cipher Suites gefordert:

- „TLS_DH_DSS_WITH_AES_256_SHA“
- „TLS_DH_RSA_WITH_AES_128_SHA“

Darüber hinaus bleibt es den Herstellern überlassen, welche alternativen Konfigurationen angeboten werden [IEC13b].

Stunnel verwendet OpenSSL, sodass über den Parameter „ciphers =“ die zu verwendenden kryptographischen Funktionen, in der von OpenSSL bekannten Notation, übergeben werden können.

Nachstehenden Konfiguration erlaubt beispielsweise nur starke kryptographische Funktionalität aus dem TLS-Protokollspezifikationen, sowie die vom IEC 62351 geforderten und in SSL v3 spezifizierte Cipher Suites.

```
ciphers = !SSLv2 -SSLv3 EECDH+ECDSA+AESGCM EECDH+aRSA+AESGCM
  DH+aRSA+AESGCM DH+DSS+AES256+SHA DH+aRSA+AES128+SHA
```

Je nach clientseitiger Unterstützung wird beim Verbindungsaufbau eine der zur Verfügung stehenden Cipher Suites ausgewählt.

Kapitel 6

Ergebnisse

Nachfolgend werden die wichtigsten Ergebnisse dieser Arbeit in Hinblick auf den Einsatz der EJBCA als ausgewählte PKI-Implementierung sowie die Erkenntnisse in der Umsetzung des Sicherheitsstandards IEC 62351 im Bereich von embedded Devices besprochen.

6.1 EJBCA und PKI

Die in der Anforderungsanalyse aufgestellten grundlegenden und erweiterten Funktionen wurden durch die EJBCA vollkommen abgedeckt. Darüber hinaus wird von der PKI-Implementierung zusätzliche Funktionalität angeboten, welche deutlich über den Erfordernissen diese Projektes liegen.

Im Allgemeinen ist der Betrieb einer PKI sehr ressourcenintensiv und steigt überproportional mit den zu verwaltenden Zertifikaten an. Neben der PKI (EJBCA) und dem Jboss Application Server müssen im Wesentlichen Datenbanken, Webserver, Verzeichnisdienste und ein Mail Transfer Agent ordentlich konfiguriert und ständig aktuell gehalten werden. Neben diesen systemadministrativen Anforderungen werden auch Kenntnisse im Bereich der Netzwerktechnik (Firewall, Load-Balancing), ein detailliertes Wissen um X.509-Zertifikate und ein sicherer Umgang mit den diversen OpenSSL-Tools und dem Java-Keytool vorausgesetzt. Ein funktionierendes Backup und Recovery sowie Sicherheitsupdates sind darüber hinaus nicht zu vernachlässigen.

Beide Systeme, sowohl die EJBCA als auch der als Backend verwendetet Jboss Application Server werden seit mehr als zehn Jahren aktiv entwickelt, sodass sie sowohl Stabilität aufweisen, als auch gut durchdacht konzipiert sind. Die Installation des Systemes ist durch die gegliederten und gut dokumentieren Konfigurationsdateien sowie mit Hilfe des Apache Ant Tools nach einer notwendigen Einarbeitungszeit, leicht realisierbar. Dabei ist ein Mindestmaß an Java-Kenntnissen Voraussetzung.
Die Integration einer externen Datenbank wie sie in dieser Arbeit am Beispiel von

MySQL gezeigt wurde, ist problemlos möglich und hinsichtlich der Reaktionszeit der Anwendungen konnten im Vergleich zur internen Hypersonic-Datenbank (HSQLDB) keine Performance-Unterschiede festgestellt werden. Mit Hilfe des Jboss Application Server war es möglich, unterschiedliche Instanzen der EJBCA auf einem System zu betreiben. Dabei wurde zumindest ein Produktiv-System und ein Testsystem eingesetzt, um Beispielsweise CAs oder Profile von der einen Instanz in die andere zu transferieren oder das Backup und Recovery zu testen.

Nahezu alle in den Use-Cases definierten Aufgabenstellungen konnten sowohl über das graphische Webinterface, als auch auf der Command-line mit Hilfe des CLIs umgesetzt werden. Je nach Aufgabenstellung ist die eine oder andere Variante besser geeignet. Über das CLI lassen sich Aufgaben unter der Zuhilfenahme von cron einfach automatisieren. Auch das skriptgesteuerte Ausstellen von mehreren Zertifikaten ist problemlos möglich. Das Admin-Web eignet sich besser zum Erstellen der Profile oder zum Suchen nach bereits ausgestellten oder widerrufenen Zertifikaten. Dank der guten Dokumentation auf der Homepage des Projektes, aber auch durch die direkt im Admin-Web verlinkten Hilfeseiten in der lokalen Installation und den Syntax-Erklärungen zu den CLI-Befehlen, wird der Einstieg in das System erleichtert.

Das Profil-System (Zertifikat- und EE-Profile) stellt am Beginn, beim Einrichten der PKI einen Mehraufwand da, erleichtert jedoch das Hinzufügen von weiteren CAs und das Ausstellen von End User Zertifikaten ungemein. Besonders die Möglichkeit des Ableitens von Profilen ist dabei ein Vorteil und erinnert an das Ableiten von Klassen in der objektorientierten Programmierung. Beispielsweise wird ein abstraktes Profil für einen User erstellt, in welchen ausschließlich die Werte für O, OU, C des DNs spezifiziert werden. Ausgehend von diesem Profil werden dann zwei allgemeine Profile für Server und Client Anwendungen spezifiziert, die wiederum als Vorlage für speziellere Anwendungsfälle dienen können.
Neben dem einfachen Erstellen von Zertifikaten auf Grund dieser Profile ist es mit ihnen möglich, Teile der Policies von Zertifizierungsstellen abzubilden. Indem einzelne Parameter als verpflichtend oder für den User als nicht mehr änderbar markiert werden, ist es dem RA Administrator oder auch einem User, der ein Zertifikat per CSR beantragt, nicht mehr möglich, diese Werte zu ändern oder nicht zu befüllen.

Somit kann abschließend festgehalten werden, dass die EJBCA den Umgang mit Zertifikaten deutlich erleichtert. Der Einsatz einer vollwertigen PKI bringt darüber hinaus weitere Vorteile mit sich. Dem gegenüber steht ein nicht zu unterschätzender Aufwand im Bereich der Systemadministration. Durch das modulare Konzept der EJBCA können viele der Komponenten bei Bedarf zu einen späteren Zeitpunkt eingerichtet respektive ausgelagert werden. Der Einsatz der EJBCA als PKI-Implementierung eignet sich nicht nur in großen Umgebungen, sondern ist auch für das Verwalten von Zertifikaten in kleineren Organisationen anwendbar.

6.2 X.509 Zertifikate im embedded Bereich

Nachstehende Ergebnisse müssen immer vor dem Hintergrund betrachtet werden, dass einerseits die Fertigung der embedded Devices von Firmen übernommen wird, welche keine direkte Verbindung zur PKI haben. Zum anderen werden die Geräte in Einrichtungen zu elektrischen Energiegewinnung eingesetzt, welche ebenfalls keine Anbindung zu einem öffentlichen Datennetz besitzen. Aufgrund dieser Vorgaben kann unter Anderem eine Verifikation der Zertifikate nicht mit Hilfe des OCSP erfolgen. Weiters ist es nicht möglich die Zertifikate mit Hilfe bekannter Techniken wie Beispielsweise dem SCEP zu verteilen.

Ein generisches Zertifikat für alle embedded Devices kann einfach über das Buildsystem in das Linux Image automatisiert integriert werden.

Im Use-Case 2 (4.2.1) werden zwei Varianten vorgestellt, die es ermöglichen, jedes embedded Devices mit einem eigenen Zertifikat zu versehen.
In der ersten Variante, in welcher eine ganze SubCA inklusive des privaten Schlüssels exportiert wird, ist zwar vom Aufwand her einfacher umzusetzen, hat jedoch den großen Nachteil, dass die Kontrolle über die erzeugten Zertifikate vollständig verloren geht. Es ist somit im Nachhinein nicht nachvollziehbar, wie viele Zertifikate ausgestellt worden sind und auf welchen Geräten diese im Einsatz sind. Das Widerrufen einzelner Zertifikate ist somit nicht mehr möglich. Im schlechtesten Fall bleibt nur die Möglichkeit, die komplette SubCA zu widerrufen.
Die zweite vorgestellte Variante ist technisch aufwendiger, jedoch bleibt die Kontrolle über die ausgestellten Zertifikate nahezu vollständig erhalten.
Für beide Varianten wurde gezeigt, dass sich sowohl CAs als auch Zertifikate manuell und automatisiert exportieren lassen, um sie anschließend entweder in das Linux-Image zu integrieren oder gesammelt in einen Archiv verpackt und verschlüsselt dem Hardware-Hersteller zu übergeben.
Das Einbinden der von der EJBCA erstellten TLS-Server-Zertifikate in den TLS-Wrapper Stunnel kann dabei automatisiert abgewickelt werden.

Analog zu den Zertifikaten wurden Möglichkeiten zur Erstellung von CRLs erarbeitet. Die Sperrlisten können ebenfalls automatisiert generiert und in das Linux-Image integriert oder dem Fermentierungsprozess zugeführt werden.

Um eine Übersicht über abgelaufene oder widerrufen Zertifikate zu waren wurden bereits vorhandene Techniken der EJBCA erweitert. In diesen Fällen werden die Verantwortlichen der PKI per Email darüber verständigt.

6.2.1 IEC 62351

Die im IEC 62351 geforderten Maßnahmen zur Absicherung respektive der Verschlüsselung des TCP/IP-Protokolls konnten soweit sie für diese Projekt zutreffend waren, umgesetzt werden.
Dabei ist der Sicherheitsstandard IEC 62351, der aus elf Teilen besteht zum einen noch unvollständig, weil zumindest zwei Teile zur Zeit nicht veröffentlicht sind. Darunter befindet sich auch der für diese Arbeit nicht unwichtige Teil 9 „Cyber security key management for power system equipment“.
Zum anderen entsprechen einige Teile des Standards nicht mehr den aktuellen Anforderungen. Als Beispiel sei hier die im Standard als verpflichtend zu implementierende Cipher Suite „TLS_DH_RSA_WITH_AES_128_SHA“ angeführt. 2007, als dieser Teil des Standards veröffentlicht wurde galten diese kryptographischen Funktionen als sicher. Darüber hinaus forderte der Standard mit dieser Cipher Suite, welche Perfect Forward Secrecy erzwingt, deutlich mehr an Sicherheit als zur damaligen Zeit üblich war. Aktuell jedoch gilt SHA-1 als zumindest theoretisch gebrochen und sollte nicht mehr verwendet werden[1]. Für den verwendeten symmetrischen Verschlüsselungsalgorithmus AES stehen neue und sicherere Betriebsmodi, wie beispielsweise der Galois Counter Mode, zur Verfügung. Der Diffie–Hellman-Merkle Algorithmus (DH) ermöglicht einen sicheren Austausch des Schlüsselmateriales (Perfect Forward Secrecy) und garantiert auch dann die Vertraulichkeit der Daten, wenn der private Schlüssel der Zertifikate zu einem späteren Zeitpunkt bekannt wird. Die Verwendung von Perfect Forward Secrecy benötigt jedoch deutlich mehr Ressourcen[2]. Diesbezüglich wäre gerade bei embedded Devices der Einsatz von Elliptic Curve Cryptography (ECC) zu präferieren. In Kapitel 5.9 werden somit sowohl die vom IEC 62351 geforderten Cipher Suites aber auch neuere den derzeitigen Stand der Technik entsprechende kryptographische Funktionen konfiguriert.

6.3 Erbebnisse für die Use-Cases im Detail

6.3.1 Use-Case 1: System Setup

Anforderung:

Neben der EJBCA selber sind auch alle weiteren für den Betrieb der PKI notwendigen Services einzurichten. Die von der EJBCA bereits voreingestellt verwendetet HSQL-Datenbank ist gegen eine externe Datenbank zu ersetzen. Der Betrieb voneinander unabhängigen Instanzen soll für Testzwecke ermöglicht werden.

Umsetzung:

[1] https://www.pki.dfn.de/faqpki/faqpki-sha2/ (Aufgerufen: März 2014)
[2] http://vincent.bernat.im/en/blog/2011-ssl-perfect-forward-secrecy.html (Aufgerufen: März 2014

Sowohl die EJBCA als auch der Jboss Application Server wurden in unterschiedlichen Versionen installiert und konnten nebeneinander betrieben werden. Das Anbinden einer MySQL-Datenbank wurde mit der Hilfe von JDBC-Konnektoren bewerkstelligt, sodass es auch möglich ist, verschiedene Instanzen der PKI auf einen Host zu betreiben.

6.3.2 Use-Case 2: Abbildung der CA-Strukturen

Anforderung::

Die für dieses Projekt notwendige Root CA sowie 4 weitere benötigte SubCA-Trees sind einzurichten. Bei der Ausbildung der Struktur ist darauf zu achten, dass sie jederzeit erweitert werden kann. Des weiteren sind die für diese Projekt spezifischen Anforderungen hinsichtlich des Exportierens der einzelnen SubCAs sowie das Automatisierte erstellen von Zertifikaten umzusetzen.

Umsetzung:

Die Root CA wurde mit einem selbst signierten Zertifikat erstellt. Alle unmittelbar darunterliegenden SubCAs wurden von der Root CA signiert. Jede dieser SubCAs bildet dabei einen Einsatzbereich der PKI ab und stellt entweder direkt Zertifikate für den Enduser aus oder dient als Container (Intermediate-CA) für weitere SubCAs. Die für die Erstellung der CAs notwendigen Profile wurden allesamt von den bereits vorhanden Profilen abgeleitet und erweitert.

Der Export einzelner SubCAs wurde sowohl auf der Command-Line, als auch im Admin-Web durchgeführt. Dabei wurde unterschieden, ob von der CA nur das Zertifikat inklusive des öffentlichen Schlüssels oder ob auch zusätzliche der private Schlüssel mit exportiert werden soll. In weiterer Folge wurde gezeigt, wie die exportierten Daten in andere gebräuchliche Formate übergeführt werden können um sie anschließend in andere Systeme wie beispielsweise das der Hardware-Hersteller, integrieren zu können.

Anders als bei den CAs mussten die für das Ausstellen von EE-Zertifikaten benötigten Profile alle neu erstellt werden. Dabei wurde darauf geachtet, dass die Vorlagen am Beginn möglichst allgemein gehalten werden, um dann in weiterer Folge genauer ausgearbeitet zu werden. Das automatische Erstellen von EE-Zertifikaten für die TLS-Verschlüsselungen auf den embedded Devices wurde per bash-Script und mit Hilfe des Cron-Dienstes realisiert. Einzelnen Zertifikate für End User (EEs) wurden über das Admin-Web erstellt. Für das Verteilen der Zertifikate wurde zum einen der CSR-Enrollment-Prozess im öffentlichen Webinterface der EJBCA freigeschaltet und zum anderen wurden Zertifikate und der dazu korrespondierende private

Schlüssel innerhalb der PKI erzeugt, um diese dann auf alternativem Wege automatisiert verteilen zu können.

6.3.3 Use-Case 3: Zertifikatssperrung

Anforderung::

Allgemeine, als auch für dieses Projekt spezielle Anwendungsfälle zum Widerrufen von Zertifikaten sollen auf ihre Umsetzbarkeit innerhalb der EJBCA geprüft und wenn notwendig implementiert werden. Aufgrund der Vorgaben für dieses Projekt soll die Veröffentlichung der gesperrten und widerrufenen Zertifikate per CRL für jede im System befindliche CA getrennt und automatisiert erfolgen. Die Sperrlisten für die SubCAs der Hardware-Hersteller sind auf die embedded Systeme zu übertragen.

Umsetzung:

Die EJBCA bietet als PKI-Implementierung im Unternehmensumfeld sowohl Konzepte für das Sperren von Zertifikaten als auch Lösungen zum Veröffentlichen von CRLs sowie die Möglichkeit der Validierung von Zertifikaten per OCSP. Die bereits vorinstallierten Konzepte für das vorübergehende und auch das endgültige Sperren von Zertifikaten sind für dieses Projekt ausreichend. Ein oder mehrere Zertifikate könne sowohl im Admin-Web als auch mit Hilfe des CLI gleichzeitig widerrufen werden. Um CRLs automatisiert und in periodischen Abständen veröffentlichen zu können wird gezeigt, welche Konfigurationsparameter diesbezüglich beim Erstellen der CAs zu setzen sind und wie ein CRL-Update-Service sowohl über das Admin-Web als auch am CLI mit Hilfe von Cron zu realisieren ist. Das Exportieren dieser Sperrlisten ist ebenfalls sowohl über das Admin-Web als auch auf der Befehlszeile möglich. Somit konnten die CRLs per Script in das Linux-Image integriert werden.

6.3.4 Use-Case 4: Rollenbasierte Authentifizierung

Anforderung::

Für den zukünftigen Betrieb der PKI ist zu erwarten, dass Personen mit unterschiedlichen Aufgabenbereichen und damit einhergehenden unterschiedlichen Berechtigungen eine oder mehrere CAs administrieren werden. Auf Grund dieser Annahme sollen zu der Rolle des sogenannten „Superadministrators“ weitere Rollen der PKI hinzugefügt werden. Diese Rollen sollen die Rechte des Users auf bestimmte CAs beschränken.

Umsetzung:

Für die rollenbasierte Authentifizierung wurde die „Management_Usr“ SubCA dem System hinzugefügt. Mit ihr ist es nur dem „Superadministrators“ möglich Zertifikate für weitere Administratoren auszustellen. Über das Benutzermanagement der EJBCA wurden neue Administratorengruppen mit eingeschränkten Rechen erstellt. Die dabei entstehenden Rollen werden über die Seriennummer der ausgestellten Zertifikate den entsprechenden Personen zugeordnet. Das Verteilen dieser Zertifikate erfolgt nicht wie üblich über einen CSR. Vielmehr werden diese Zertifikate und der dazu korrespondierende private Schlüssel dem neuen Administrator persönlich übergeben.

6.3.5 Use-Case 5: Automatische Benachrichtigungen

Anforderung::

Zum einen sollen Inhaber von bereits ausgestellten Zertifikaten und die PKI-Administratoren automatisch über eine Statusänderung eines Zertifikates informiert werden. Weiters ist es für die Verteilungsprozesse von Zertifikaten wie beispielsweise dem CSR von Vorteil, wenn der User per E-Mail automatisch darüber informiert wird, dass sein Zertifikat jetzt angefordert werden kann. Im Fall eines bevorstehenden Gültigkeitsverlustes von Zertifikaten sollen die verantwortlichen Personen mit einer frei einstellbaren Vorlaufzeit benachrichtigt werden.

Umsetzung:

Für die Umsetzung der automatischen Benachrichtigungen wurde ein MTA am System installiert und konfiguriert. Für das Versenden von E-Mails im Falle von Statusänderungen der Zertifikate wurden dementsprechend die sogenannten End-Entity-Profile adaptiert, sodass derzeit Benachrichtigungen sowohl an den User als auch an den zuständigen Administrator verschickt werden. Für das ermitteln von Zertifikaten, bei welchen ein Gültigkeitsverlust bevorsteht, werden CLI-Befehle der EJBCA in einem Script, welches ein mal am Tag per Cron ausgeführt wird, abgearbeitet. Derzeit wird der Administrator 100 Tage im voraus darüber informiert, dass ein bestimmtes Zertifikat seine Gültigkeit verliert.

6.3.6 Use-Case 6: Backup und Recovery

Anforderung::

Es soll sichergestellt sein, dass zumindest für den Testbetrieb relevante von den Voreinstellungen abweichende Dateien gesichert werden. Zusätzlich soll auch die MySQL-Datenbank in regelmäßigen Abständen gesichert werden. Ein vollständiges

Backup- und Recovery-Konzept für alle Systemrelevanten Teile würde den Rahmen dieser Arbeit übersteigen.

Umsetzung:

Das System wird in einer KVM-Umgebung auf einen LVM-Raid 1 Festplatten-Verbund betrieben. Dieses Setup ist für den experimentellen Betrieb der PKI mehr als ausreichend. Darüber hinaus wurde mit Bordmitteln sowohl die MySQL-Datenbank als auch die sich ändernden Daten gesichert.

6.3.7 Use-Case 7: Policies

Anforderung::

Da es sich bei diesem Projekt um ein Konzept handelt und es zum gegenwärtigen Zeitpunkt noch nicht klar ist wie und in welcher Form die PKI eingesetzt wird, kann auf ein explizites niederschreiben der Policies verzichtet werden.

Umsetzung:

Im Umgang mit den EE-Profilen wurde in dieser Arbeit gezeigt, wie etwaige Policies umsetzbar sind und wie sie eingehalten und kontrolliert werden können. So wird beispielsweise beschrieben wie die Vollständigkeit des DNs eingehalten werden kann oder wie über das Setzen von vorgegebenen und verpflichtend auszufüllenden Feldern die User dazu angehalten werden können, ausschließlich vollständige Angaben zu machen oder wie bei der Auswahl der Passwörter gewisse Regeln einzuhalten sind.

6.3.8 Use-Case 8: Umsetzung der im Sicherheitsstandard geforderten Parameter für TLS

Anforderung::

Die im Sicherheitsstandard IEC 62351-Standard vorgeschriebenen, verpflichtenden und optionalen Maßnahmen müssen, soweit sie für dieses Projekt zutreffend sind, umgesetzt werden.

Umsetzung:

Die auf TCP-IP basierende Kommunikation wird mit Hilfe des TLS-Wrappers „stunnel“ verschlüsselt. Die dafür benötigten Zertifikate werden von der EJBCA erzeugt und automatisiert in das Linux-Image integriert. Die ebenfalls von der EJBCA in regelmäßigen Abständen erstellten CRLs werden auf den embedded Systemen mit

Hilfe eines schlanken Webservers bereitgestellt. Hinsichtlich der zu verwendenden kryptographischen Funktionen wurden die vom Sicherheitsstandard (IEC 62351) empfohlenen Konfigurationen um sicherere Cipher Suites erweitert.

Kapitel 7

Zusammenfassung

Der Einsatz von Verschlüsselung und die Authentifizierung mit Hilfe von X.509-Zertifikaten in der elektronischen Datenübertragung muss nicht zwangsläufig eine sichere Kommunikation gewährleisten. Aufgrund der sich ständig weiterentwickelnden Angriffsvektoren und immer wieder auftretenden „Software-Bugs“ ist das Absichern einer elektronischen Datenübertragung ein andauernder und fortwährender Prozess.

Die Verwendung von modernen kryptographischen Funktionen, wie die Verschlüsselung der Daten mit Hilfe von AES-GCM oder erweiterten Sicherheitsmechanismen wie Perfect Forward Secrecy, kann dabei die Leistungsfähigkeit des Prozessors eines embedded Systems übersteigen. Aus diesem Grund ist bereits bei der Hardware-Auswahl solcher Systeme darauf zu achten, dass die CPU bereits kryptographische Operationen in Hardware unterstützt.
Des weiteren können sich aus den speziellen Einsatzbereichen der embedded System weitere Einschränkungen für die Anwendung von standardisierten und bereits erprobten Methoden ergeben.

Durch die Verwendung einer PKI für den embedded Bereich konnte in dieser Arbeit gezeigt werden, dass mit Hilfe der EJBCA und den selbst hinzugefügten Erweiterungen, die an dieses Projekt gestellten Anforderungen erfüllt werden können. Das Einrichten der PKI-Infrastruktur sowie der Aufwand für den ständigen Betrieb ist beträchtlich und darf nicht unterschätzt werden. Dem gegenüber stehen die Erleichterungen im Ausstellen und Verwalten der Zertifikate.
Für das Exportieren der Zertifikate und CRLs auf die embedded Systeme wurden Konzepte ausgearbeitet und implementiert. Die bestehende Datenübertragung auf den embedded Systemen wurde mit Hilfe des TLS-Wrapper „stunnel“ unter der Verwendung von X.509 Zertifikaten verschlüsselt.

7.1 Weiterführende Arbeiten

In einem nächsten Schritt ist die PKI in die Unternehmensstruktur einzuführen. Hinsichtlich der Erstellung und Verteilung der Zertifikate für die embedded Systems sind die beiden vorgestellten Methoden auf ihre praktische Umsetzbarkeit zu evaluieren. Diesbezüglich wird zur Zeit ein weiteres Projekt am Institut für Technische Informatik der Technische Universität Graz durchgeführt. In diesem Projekt soll eine sogenannte „Programmier- und Testeinheit (Pute)“ inklusive Software bereitgestellt werden, welcher unter anderem das Linux-Image inklusive der benötigten Zertifikate auf den embedded Systemen installiert und automatisiert Testroutinen durchführt.

Für den weiteren Betrieb der PKI ist es notwendig, dass die Policies entsprechend der Nutzung der CAs niedergeschrieben werden und dass Prozesse entwickelt werden, mit welchen die Einhaltung der Policies kontrolliert werden können. Des weiteren sollte sowohl ein Security als auch ein Backup und Recovery Konzept für alle zusätzlichen Dienste, die für den Betrieb der PKI notwendig sind, erstellt werden.

Im Bereich der embedded Systeme sollte die TLS-Verschlüsselung, die derzeit mit Hilfe des TLS-Wrapper „stunnel“ umgesetzt ist, nativ in die bestehende proprietäre Client-Server-Software integriert werden.

Anhang A

IEC 62351

A.1 IEC 62351-4: Empfohlene Cipher Suites

Neben der verpflichtenden zu implementierenden Cipher Suite *„TLS_DH_DSS_WITH_AES_256_SHA“* wird von Standard empfohlen nachstehende kryptographische Funktionen anzubieten.

Key exchange		Encryption	Hash
Algorithm	**Signature**		
TLS_RSA_		WITH_RC4_128_	SHA
TLS_RSA_		WITH_3DES_EDE_CBC_	SHA
TLS_DH_	DSS_	WITH_3DES_EDE_CBC_	SHA
TLS_DH_	RSA_	WITH_3DES_EDE_CBC_	SHA
TLS_DHE_	DSS_	WITH_3DES_EDE_CBC_	SHA
TLS_DHE_	RSA_	WITH_3DES_EDE_CBC_	SHA
TLS_DH_	DSS_	WITH_AES_128_	SHA
TLS_DH_	DSS_	WITH_AES_256_	SHA
TLS_DH_		WITH_AES_128_	SHA
TLS_DH_		WITH_AES_256_	SHA

Tabelle A.1: Empfohlenen Cipher Suites nach IEC 62351-4 - Quelle: [IEC07b]

Anhang B

EJBCA

B.1 Profile

B.1.1 Zertifikat Profile

Exemplarische Zertifikat Profil für einen SSL/TLS Webserver. Die nachstehende Konfiguration eignent sich nicht für eien Produktivbetrieb.

Profilname: 61_Bord_SSL_Server

```
Type: select End Entity
Available bit lengths: 2048 and higher
Validity: 1y
Allow Key Usage Override: deselect
Key usage: select Digital Signature and Key encipherment
Extended Key Usage:select  Server Authentication
CRL Distribution Points: select Use
Use CA defined CRL Dist. Point: select Use
Available CAs: select all 2x_SSubCA_HA_Man_xxxxx
```

B.1.2 End Entity Profil

Exemplarische End Entity Profil für einen SSL/TLS Webserver für ein automatische Generieren der Zertifikate per CLI. Die nachstehende Konfiguration eignent sich nicht für eien Produktivbetrieb.

Profilname: 61_Bord_SSL_Server_Batch_p12

```
E-mail Address: select Use, Required and Modifiable; fill in the domain part
CN, Common name: select Required and Modifiable; leave it blank
O, Organization: select Required and Modifiable; enter the Organization
C, Country (ISO 3166): select Required and Modifiable; enter the Country
```

```
IP Address: select Required and Modifiable; leave it blank
Available Certificate Profiles: 61_Bord_SSL_Server
Default Token: P12 file
Available Tokens: P12 file
```

Anhang C

Stunnel

C.1 stunnel.conf

Nachstehend aufgelistet sind jene Parameter, welche für einen standardkonformen Betrieb des stunnels notwendig sind.

```
cert = /etc/stunnel/cert-key.pem
CAfile = /etc/stunnel/ca-chain.pem
options = NO_SSLv2
ciphers = !SSLv2 -SSLv3 EECDH+ECDSA+AESGCM EECDH+aRSA+AESGCM
  DH+aRSA+AESGCM DH+DSS+AES256+SHA  DH+aRSA+AES256+SHA
```

Acronyms

1CD 1st Committee Draft. 22

AES Advanced Encryption Standard. 72

ANW Approved New Work. 22

AS Application Server. 65–67, 82, 83

CA Certification Authority. 13, 15, 17, 27, 29, 32, 34, 36–38, 40, 41, 48, 49, 53, 55, 57, 63, 68, 83, 92

CAF Consider All Facts. 35

CLI Command Line Interface. 36, 38, 42, 49, 50, 70–73, 76, 78, 83, 87, 88, 94

CPS Cyber Physical Systems. 10

CRL Certificate Revocation List. 27, 36, 38, 48, 56, 58, 60, 61, 75–79, 84, 87, 89, 91

CSR Certificate Signing Request. 48, 63, 74, 75, 83, 86, 88

DBAL Database Abstraction Layer. 40

DNS Domain Name System. 19

DSS Digital Signature Standard. 26

ECC Elliptic Curve Cryptography. 85, 101

EE End Entity. 48, 53, 54, 57, 59, 68, 70–74

FIPS Federal Information Processing Standards. 79

GCM Galois Counter Mode. 85

GUI Graphical User Interface. 37, 38, 49

HAB High Assurance Boot. 10, 32, 56, 71

HTTP Hypertext Transfer Protocol. 19, 31

HTTPS Hypertext Transfer Protocol Secure. 15

IMAPS Internet Message Access Protocol Secure. 15

IP Internet Protocol. 19, 20, 24, 25, 27, 64, 85

IPSec Internet Protocol Security. 12

JSS Network Security Services for Java. 43

MAC Message Authentication Code. 25

MERGED Merged project. 22

MITM Man-in-the-middle. 25

NSS Network Security Services. 43

OCSP Online Certificate Status Protocol. 39, 40, 45, 48–50, 61, 62, 84

OSI Open Systems Interconnection model. 27, 28

PGP Pretty Good Privacy. 14, 16

PKI Public-Key-Infrastruktur. 10, 11, 13–15, 17, 19, 23, 31, 33, 34, 37–45, 47–53, 55–58, 60, 62–64, 67, 69, 71

PKIX Public-Key-Infrastruktur (X.509). 14–16, 19

PPUB Publication issued. 22

PRQP Resource Query Protocol. 39

RA Registration Authority. 36, 37, 40, 48, 49, 72, 83

RBAC Role-Based Access Control. 27

RFC Request for Comments. 14, 24, 25, 27, 28, 37, 50

S/MIME Secure/Multipurpose Internet Mail Extensions. 12

SCADA Supervisory Control and Data Acquisition. 24

SCEP Simple Certificate Enrollment Protocol. 39–41, 45, 54, 55, 84

SCVP Server-based Certificate Validation Protocol. 61, 62

SDSI Simple Distributed Security Infrastructure. 18

SMTPS Simple Mail Transfer Protocol Secure. 15

SPKI Simple Public Key Infrastructur. 18

SSL Secure Sockets Layer. 12, 15

SubCA subordinate Certification Authority. 37, 41, 48, 53–56, 60, 62

TCP Transmission Control Protocol. 19, 20, 24, 25, 27–29, 64, 85

TLS Transport Layer Security. 10, 12, 15, 19, 23–29, 31, 32, 37, 38, 53, 54, 56, 57, 59, 72, 91, 92, 101

TPDU Trtansport Protocol Data Unit. 28

WOT Web of Trust. 17

Glossary

Admin-Web Administrations-Web-Interface der EJBCA, erreichbar unter https://<IP.des.PKI.Rechners>:8443/ejbca/adminweb/index.jsp. Je nach vergebenen Rechten wird beispielsweise über das Adminweb die PKI konfiguriert, neue Services oder Profile angelegt oder Zertifikate ausgestellt. 68, 71–73, 75, 77, 79, 80, 84, 87, 88

Cipher Suite Eine Cipher Suite ist eine mögliche Kombination aus diversen kryptographischen Funktionen. Als Beispiel dient: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x39). Dieser Sting beinhaltet Angaben zum Austausch des öffentlichen Schlüssel, verwendete asymmetrisches und symmetrischer Verschlüsselungsalgorithmus, Betriebsmodus des Block Ciphers und verwendete Hashfunktion für die Authentifizierung im TLS Protokoll.. 25, 26, 29, 81, 82, 86, 94

HSM Ein Hardware Security Module dient zum ausführen von kryptographischen Funktionen in dafür speziell vorgesehener Hardware. Dies erhöht nicht nur die Sicherheit sondern entlastet auch die CPU bei den rechenintensiven Prozessen. Des weiteren werden auf HSMs auch kryptographischen Schlüssel gespeichert. Als Schnittstelle zur Software dient oft der Cryptographic Token Interface Standard (PKCS #11).. 40, 43, 45, 69, 72

IEC (International Electrotechnical Commission) Die Internationale Elektrotechnische Kommission, mit ihren Hauptsitz in Genf, ist eines von mehreren internationalen Normungsinstituten für die Bereiche Elektrotechnik und Elektronik. In ihr sind ca. 70 Nationalstaaten durch die sogenannten nationalen Komitees (NC) vertreten. Am Beispiel Österreich, durch den Österreichischer Verband für Elektrotechnik (OVE). 19–21, 23–26, 29, 32, 36, 65

IETF Die Internet Engineering Task Force (IETF) ist eine offenen internationale Freiwilligenvereinigung. Sie besteht aus NetzwerktechnikerInnen, Herstellern, Netzbetreibern, ForscherInnen und AnwenderInnen. Ihre Mitglieder sind sowohl privater Natur aber auch Personen welche in einen Angestelltenverhältnis stehen. Ihre Aufgabe ist die Erstellung technischer Dokumente, den sogenannten RFCs, in welchen die Weiterentwicklung, Benutzung und Verwaltung des Internets spezifiziert sind.. 12, 14, 19, 24

ISO on TCP/IP ISO Transport Service on top of the TCP (ISO on TCP/IP) portiert das ISO TP0 [ISO8072] Protokoll für TCP/IP Netze, sodass auch alle höheren Klassen (TP1 - TP4) des ISO8072 Protokolls transparent in Verbindung mit TCP/IP genutzt werden können. Das ISO8072 Protokoll besitzt keine Routing-Funktionalität, ist jedoch im Automatisierungsbereich weit verbreitet und wird zum Beispiel von den speicherprogrammierbaren Steuerungen der Firma Siemens unterstützt.. 28, 29

MMS Manufacturing Messaging Specification ist ein Standard zum objektorientierten Austausch von Daten in Automatisierungssystemen. Druch Schaffen von einheitlichen Kommunikationsschnittstellen oder das verwenden von Standardnetzwerkkomponenten und der gelichen, soll eine Integration verschiedener Geräte in einen kompletten System ermöglichen.. 20, 27, 65

Nonce aus dem englischen kommen und steht sinngemäß für „used only once“. d.h. ein Zufälliger Wert welcher nur einmal verwendet werden darf um z.B.: Replay-Angriffe zu verhindern.. 25

Perfect Forward Secrecy PFS verhindert, dass bei bekanntwerden des geheimen Langzeitschlüssels, auf die verwendeten Session Keys zurückgeschossen werden kann. Und somit auch, dass von einem Angreifer aufgezeichnete Kommunikationsdaten im Nachhinein entschlüsselt werden können. PFS wird von TLS ab Version 1.1 [DR06] unterstützt und ist unter Experten seit langem ein Thema. PFS wird bei TLS durch den Diffie–Hellman-Merkle Algorithmus gewährleistet. Dabei wird ein Verfahren mit und eins ohne ECC angeboten.. 26, 86, 92

PKCS #11 Der Cryptographic Token Interface Standard ist ein Public Key Cryptography Standard und beschreibt eine API für Security-Hardware wie zum Beispiel für Smart Cards oder andere HSMs. Eine PKCS #11 Bibliothek stellt somit diverser Funktionen im Umgang mit Objekten auf den HSMs (=kryptografischen Token) zur Verfügung.. 38, 43

PKCS #12 Public-Key Cryptography Standards #12 ist ein zumeist mit einem Passwort geschütztes Archiv. Eine PKCS #12 Datei kann dabei X.509 Zertifikate und den dazu korrespondierenden privaten Schlüssel, sowie weitere benötigte Zertifikate (CA-Chain) oder CRLs enthalten.. 72, 75, 81

Public Web Die öffentliche Webseite der EJBCA ist erreichbar unter https://<IP.des.PKI.Rechners>:8443/ejbca/. Sie dient unter anderen dazu den User der PKI die Zertifikate der CAs und die dazugehörigen Sperrlisten anzubieten. Weiters werden über dieses Interface der Zertifik Enrollment-Prozess abgewickelt.. 75

TC 57 WG 15 Die IEC ist Unterteilt in Technischen Komitees (TC), Unterkomitees (SC; subcommittees) und zumeist mehren Arbeitsgruppen (WG; working groups). In diesen Gruppen werden die einzelnen Standards ausgearbeitet und überprüft. Das TC 57 ist verantwortlich für Standards im Bereich von Management und Informationsaustausch in Anlagen zur Gewinnung und Verteilung von elektrischer Energie. Die WG 15, im speziellen befasst sich mit dem Standard IEC 62351 im Bereich von Daten- und Kommunikationssicherheit.. 23, 27

Literaturverzeichnis

[AMM06] Klink Alexander, Bartosch Martin, and Belll Michael. Building an Open Source PKI using OpenXPKI. In *Papers of the 23rd Chaos Communication Congress*, pages 68–72. Chaos Computer Club, December 2006.

[And14] Hechl Andreas. Secure remote access to embedded systems with OpenSSH and X.509v3-cerificates. Bachelor thesis, Institute for Technical Informatics, Graz University of Technology, August 2014.

[ASZ96] Derek Atkins, William Stallings, and Philip Zimmermann. PGP Message Exchange Formats. RFC 1991, Internet Engineering Task Force, August 1996. Obsoleted by RFC 4880, http://www.rfc-editor.org/rfc/rfc1991.txt.

[AW12] Hans-Ferdinand Angel and Reinhard Willfort. Die Systematik hinter „Bauchentscheidungen“. *Wissen im Dialog. Beiträge zu den Kremser Wissensmanagement-Tagen 2012*, page 21, 2012.

[Bar05] Martin Bartosch. *OpenXPKI: White Paper - Architecture Overview*, November 2005. www.openxpki.org/docs/OpenXPKI-Architecture-Overview.pdf, Accessed: November 2013.

[Bas14] Dünhofen Bastian. Implementierung einer SSL/TLS Verbindung im embedded System Bereich unter Berücksichtigung von aktuellen Bedrohungsszenarien. *Institute for Technical Informatics, Graz University of Technology*, January 2014.

[Based] Dünhofen Bastian. Anleitungen zur Installation und der Verwendung der EJBCA als Public-Key-Infrastruktur im embedded System Bereich. *Institute for Technical Informatics, Graz University of Technology*, to be published.

[BS11] Markus Braendle and Ragnar Schierholz. Cyber security for power systems #x2014; A closer look at the drivers and how to best approach the new challenges. In *Protective Relay Engineers, 2011 64th Annual Conference for*, pages 322–327, April 2011.

[CDF+07] J. Callas, L. Donnerhacke, H. Finney, D. Shaw, and R. Thayer. OpenPGP Message Format. RFC 4880, Internet Engineering Task Force, November 2007. Updated by RFC 5581, http://www.ietf.org/rfc/rfc4880.txt.

[CDFT98] J. Callas, L. Donnerhacke, H. Finney, and R. Thayer. OpenPGP Message Format. RFC 2440, Internet Engineering Task Force, November 1998. Obsoleted by RFC 4880, http://www.ietf.org/rfc/rfc2440.txt.

[CEE+01] Dwaine E. Clarke, Jean-Emile Elien, Carl M. Ellison, Matt Fredette, Alexander Morcos, and Ronald L. Rivest. Certificate chain discovery in SPKI/SDSI. *Journal of Computer Security*, 9:285–322, 2001.

[CFS+03] S. Chokhani, W. Ford, R. Sabett, C. Merrill, and S. Wu. Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework. Technical Report 3647, Internet Engineering Task Force, November 2003.

[Cho02] P. Chown. Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS). RFC 3268, Internet Engineering Task Force, June 2002. Obsoleted by RFC 5246, https://datatracker.ietf.org/doc/rfc3268.txt.

[CSF+08] D. Cooper, S. Santesson, S. Farrell, S. Boeyen, R. Housley, and W. Polk. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, Internet Engineering Task Force, May 2008. Updated by RFC 6818, http://www.ietf.org/rfc/rfc5280.txt.

[DA99] T. Dierks and C. Allen. The TLS Protocol Version 1.0. RFC 2246, Internet Engineering Task Force, January 1999. Obsoleted by RFC 4346, https://datatracker.ietf.org/doc/rfc2246/.

[DB04] E. De Bono. *Edward de Bono's Thinking Course*. BBC Worldwide, 2004.

[DH76] W. Diffie and M. Hellman. New Directions in Cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654, November 1976.

[DR06] Tim Dierks and Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.1 - RFC 4346. *Internet Engineering Task Force*, 2006.

[DR08] Tim Dierks and Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.2 - RFC 5246. *Internet Engineering Task Force*, August 2008.

[EFL+99] C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas, and T. Ylonen. SPKI Certificate Theory. RFC 2693, Internet Engineering Task Force, September 1999. http://www.ietf.org/rfc/rfc2693.txt.

[Ell99] C. Ellison. SPKI Requirements. RFC 2692, Internet Engineering Task Force, September 1999. http://www.ietf.org/rfc/rfc2692.txt.

[FAAM11] S. Fuloria, R. Anderson, F. Alvarez, and K. McGrath. Key management for substations: Symmetric keys, public keys or no keys? In *Power Systems Conference and Exposition (PSCE), 2011 IEEE/PES*, pages 1–6, March 2011.

[GP06] Ayesha Ishrath Ghori and Asra Parveen. PKI Administration Using EJBCA and OpenCA. *George Mason University*, 2006.

[HBA10] Frank Hohlbaum, Markus Braendle, and Fernando Alvarez. Cyber Security - Practical considerations for implementing IEC 62351, 2010.

[HPFS02] R. Housley, W. Polk, W. Ford, and D. Solo. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 3280, Internet Engineering Task Force, April 2002. Obsoleted by RFC 5280, https://datatracker.ietf.org/doc/rfc3280.txt.

[IEC07a] IEC TC57 WG15. Power systems management and associated information exchange - Data and communications security - Part 3: Communication network and system security – Profiles including TCP/IP. IEC 62351-3, Edition 1.0, The International Electrotechnical Commission, 3, rue de Varembe, CH-1211 Geneva 20, Switzerland, June 2007.

[IEC07b] IEC TC57 WG15. Power systems management and associated information exchange - Data and communications security - Part 4: Profiles including MMS. IEC 62351-4, Edition 1.0, The International Electrotechnical Commission, 3, rue de Varembe, CH-1211 Geneva 20, Switzerland, June 2007.

[IEC13a] IEC TC57 WG. Communication networks and systems in substations - ALL PARTS. IEC 61850-SER, Edition 1.0, The International Electrotechnical Commission, 3, rue de Varembe, CH-1211 Geneva 20, Switzerland, December 2013.

[IEC13b] IEC TC57 WG15. Power systems management and associated information exchange - Data and communications security - ALL PARTS. IEC 62351-SER, Edition 1.0, The International Electrotechnical Commission, 3, rue de Varembe, CH-1211 Geneva 20, Switzerland, April 2013.

[Koh78] Loren M Kohnfelder. *Towards a practical public-key cryptosystem.* PhD thesis, Massachusetts Institute of Technology, May 1978.

[KS05] S. Kent and K. Seo. Security Architecture for the Internet Protocol. RFC 4301, Internet Engineering Task Force, December 2005. https://www.ietf.org/rfc/rfc4301.txt.

[MA11] Marian Marius and Pirvan Andrei. Implementing a Public-Key Infrastructure for the Academic Environment. *University of Craiova*, August 2011.

[Mat13] Vierthaler Matthias. Implementing High Assurance Boot Function on the Freescale i.mx28 Processor. Bachelor thesis, Institute for Technical Informatics, Graz University of Technology, 8010 Graz, Austria, September 2013.

[MH99] A. Medvinsky and M. Hur. Addition of Kerberos Cipher Suites to Transport Layer Security (TLS). RFC 2712, Internet Engineering Task Force, October 1999. https://datatracker.ietf.org/doc/rfc2712/.

[Miced] Gissing Michael. Pute, Programmier- und Testeinheit. Master thesis, Institute for Technical Informatics, Graz University of Technology, to be published.

[MSF06] J.E. Mazur and E. Steinweg-Fleckner. *Lernen und Verhalten*. Allgemeine Psychologie. Pearson Studium, 2006.

[Pfo77] Hans-Christian Pfohl. *Problemorientierte Entscheidungsfindung in Organisationen*. Mensch und Organisation 5. Walter De Gruyter, 1977.

[RC87] M.T. Rose and D.E. Cass. ISO Transport Service on top of the TCP Version: 3. RFC 1006, Internet Engineering Task Force, May 1987. Updated by RFC RFC 2126, https://datatracker.ietf.org/doc/rfc1006/.

[RSC01] Ronald L. Rivest, Arthur C. Smith, and Dwaine Clarke. *SPKI/SDSI HTTP Server / Certificate Chain Discovery in SPKI/SDSI*. PhD thesis, Massachusetts Institute of Technology, September 2001.

[RT10] B. Ramsdell and S. Turner. Secure/Multipurpose Internet Mail Extensions (S/MIME) Version 3.2 Message Specification. RFC 5751, Internet Engineering Task Force, January 2010. ISSN 2070-1721, https://www.ietf.org/rfc/rfc5751.txt.

[Sch09] Thorsten Scherf. Gute hüter - public-key-infrastruktur mit dogtag. *Linux-Magazin Admin*, 01, January 2009.

[SMA+13] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP. Technical Report 6960, Internet Engineering Task Force, June 2013.

[Str99] Michael Ströder. *Einführung kryptographischer Techniken zur gesicherten Nutzung des Internet bei der Propack Data GmbH*. PhD thesis, Institut für Telematik der Universität Karlsruhe, January 1999.

[SYSkB10] Bin Song, In-Kwan Yu, Jiseong Son, and Doo kwon Baik. An effective access control mechanism in home network environment based on spki certificates. In *Information Theory and Information Security (ICITIS), 2010 IEEE International Conference on*, pages 592–595, Dec 2010.

[Ter00] Tero Hasu and Yki Kortesniemi. Implementing an spki certificate repository within the dns. In *3rd International Workshop on Practice and Theory in Public Key Cryptography (PKC 2000)*, volume 1751 of *Lecture Notes in Computer Science*, pages 44–63. Springer-Verlag, January 2000.