PKI-INFRASTRUKTUURIN SUUNNITTELU JA TOTEUTUS JYVSECTEC-TOIMINTAYMPÄRISTÖÖN

Ville-Veikko Helminen

Opinnäytetyö
Lokakuu 2012

Tietotekniikan koulutusohjelma
Tekniikan ja liikenteen ala

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖN
KUVAILULEHTI

Tekijä(t) HELMINEN, Ville-Veikko	Julkaisun laji Opinnäytetyö	Päivämäärä 04.10.2012
	Sivumäärä 100 + 13	Julkaisun kieli Suomi
	Luottamuksellisuus ( ) saakka	Verkkojulkaisulupa myönnetty (X)

Työn nimi
PKI-INFRASTRUKTUURIN SUUNNITTELU JA TOTEUTUS JYVSECTEC-TOIMINTAYMPÄRISTÖÖN

Koulutusohjelma
Tietotekniikan koulutusohjelma

Työn ohjaaja(t)
NARIKKA, Jorma

Toimeksiantaja(t)
Jyväskylän ammattikorkeakoulu Oy
VATANEN, Marko

Tiivistelmä

Työ toteutettiin Jyväskylän ammattikorkeakoululle JYVSECTEC-tietoturvahankkeen toimintaympäristöön. Ympäristössä tuotetaan testaus-, kehitys- ja koulutuspalveluita yhteistyöverkoston käyttöön.

Opinnäytetyön tavoitteena oli tuottaa JYVSECTEC-toimintaympäristöön varmennepalveluita tarjoava PKI-infrastruktuuri. Työ toteutettiin toimeksiantajan pyynnöstä ilmaisilla työkaluilla ja ohjelmistoilla. Järjestelmän tuli olla myös kevyt resurssivaatimuksiltaan. Työstä oli tarkoitus tehdä mahdollisimman tietoturvallinen ja todellisuutta vastaava järjestelmä. Huomioon otettiin kuitenkin, että järjestelmä tuli suljetun simulaatioympäristön käyttöön rajoittuneilla resursseilla, joten järjestelmä ei täysin vastaa oikeaa käytettävyyden parantamiseksi.
Työssä vertailtiin useita eri toteutusvaihtoehtoja, joista lopulta päädyttiin hyödyntämään vapaan lähdekoodin OpenCA ohjelmistoa, Linuxia ja useita virtuaalikoneita. Apuna käytettiin lukuisia ilmaisia Linuxiin saatavia työkaluja ja ohjelmistoja, kuten MySQL, Apache ja SSH Server.

Työ jaettiin luontevasti eri osiin. Aluksi käytiin aiheeseen liittyvä teoria läpi, jonka jälkeen suunniteltiin toteutus ja tehtiin itse käytäntö. Teoriaosuus koostuu kryptografiasta ja itse PKI:n teoriasta.

Lopputuloksena saatiin toimiva kokonaisuus JYVSECTEC-hankkeen käyttöön, jonka toiminta todettiin luomalla ja testaamalla eri prosessit muun muassa sertifikaattien myöntämiselle ja sulkemiselle. Toiminta testattiin ja todettiin myös tietoturvan osalta.

Avainsanat (asiasanat)
PKI, OpenCA, JYVSECTEC, CA, RA, Sertifikaatti, Varmenne, Tietoturva, Julkinen avain

Muut tiedot

JYVÄSKYLÄN AMMATTIKORKEAKOULU
JAMK UNIVERSITY OF APPLIED SCIENCES

DESCRIPTION

Author(s)	Type of publication	Date
HELMINEN, Ville-Veikko	Bachelor´s / Master's Thesis	04102012
	Pages 100 + 13	Language Finnish
	Confidential ( ) Until	Permission for web publication (X)

Title
PUBLIC KEY INFRASTRUCTURE DESIGN AND IMPLEMENTATION IN JYVSECTEC OPERATIONAL ENVIRONMENT

Degree Programme
Data Network Technology

Tutor(s)
NARIKKA, Jorma

Assigned by
JAMK University of Applied Sciences
VATANEN, Marko

Abstract

The work was carried out at JAMK University of Applied Sciences in the operational environment of JYVSECTEC security development project. The purpose for the environment is to produce development, testing and education services for the use of a collaboration network.

The purpose for this thesis was to produce a full scale PKI infrastructure for the use of the JYVSECTEC environment. The infrastructure offers certificate services within the environment. The assigner requested that the work should be carried out with free tools and software. The system should also be light on resource demands, secure and as realistic as possible. Though it was taken into account that the system worked in closed simulation environment with limited resources, some steps had to be made to improve the usability which reduced the realisticness of the system.
Several different implementation designs were evaluated in this study. In the end, an open source software named OpenCA, Linux and several virtual computers were used. Several free tools and software available to Linux, like MySQL, Apache and SSH Server were also used.

The thesis was divided in to three parts. First part presents theory regarding the subject. After the theory part the implementation was planned carefully and the last part was the implementation. The theory part consists of cryptography and the PKI theory itself.

As a result of the thesis a functional PKI system was built for the use of JYVSECTEC project. The functionality was proven, among other things by creating and testing different processes for certificate issuing and revoking. The security functionality was also proven.

Keywords
PKI, OpenCA, JYVSECTEC, CA, RA, Cetificate, Security, Public key

Miscellaneous

SISÄLTÖ

KUVIOT

TAULUKOT

LYHENTEET

ACL	Access Control List
AES	Advanced Encryption Standard
ASN.1	Abstract Syntax Notation One
BER	Basic Encoding Rules
CA	Certificate Authority
CDP	Certificate Distribution Point
CER	Canonical Encoding Rules
CP	Certificate Policy
CPS	Certificate Practice Statement
CRL	Certificate Revocation List
CRR	Certificate Revocation Request
CSR	Certificate Signing Request
DER	Distinguished Encoding Rules
DES	Data Encryption Standard
DNS	Domain Name System
FQDN	Fully Qualified Domain Name
HSM	Hardware Security Module
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
LDAP	Lightweight Directory Access Protocol
MD5	Message-Digest 5
OCSP	Online Certificate Status Protocol

OID	Object Identifier
PEM	Privacy Enhanced Mail
PIN	Personal Identification Number
PKCS #12	Public-Key Cryptography Standards #12
PKCS #7	Public-Key Cryptography Standards #7
PKI	Public-Key Infrastructure
RA	Registration Authority
RSA	Rivest, Shamir, Adleman
SCEP	Simple Certificate Enrollment protocol
SHA	Secure Hash Algorithm
SSL	Secure Sockets Layer
Sub CA	Subordinate Certificate Authority
URI	Uniform Resource Identifier
VPN	Virtual Private Network
WEB	World Wide Web

1 TYÖN LÄHTÖKOHDAT

1.1 Toimeksiantaja

1.1.1 Jyväskylän ammattikorkeakoulu

Työn toimeksiantajana toimi Jyväskylän ammattikorkeakoulu (JAMK), joka tarjoaa ylempään ja alempaan korkeakoulututkintoon johtavaa koulutusta nuorille ja aikuisille. Tämän lisäksi on mahdollisuus saada ammatillista opettajakoulutusta ja avoimia korkeakouluopintoja. (Tutustu JAMKiin 2012a.)

Jyväskylän ammattikorkeakoulu on vetovoimainen ja kansainvälinen korkeakoulu. Sen toimipisteet sijaitsevat Jyväskylässä ja Saarijärven Tarvaalassa. Opiskelijoita ammattikorkeakoulussa on yli 8000 ja sillä on vahva asema Jyväskylän seudun ja Keski-Suomen kehittäjien joukossa. JAMKilla on kiinteät suhteet alueen yrityksiin ja yhteisöihin. Työelämä vaikuttaa koulutuksen suuntaamiseen ja opetussuunnitelmien sisältöihin. (Tutustu JAMKiin 2012a.)

1.1.2 Tietotekniikan koulutusohjelma

Tietotekniikan koulutusohjelma keskittyy pääsääntöisesti tietoverkkotekniikan osa-alueelle. Painopisteenä koulutuksessa korostuvat operaattoritason internet-teknologiat, palvelinympäristöjen virtualisointi, pilvipalvelujen toteuttaminen, verkkojen ylläpito ja suunnittelu sekä tietoturvallisuuden hallinta. Koulutusohjelman laajuus on 240 opintopistettä. (Tietotekniikan koulutusohjelma 2012b.)

Opintojen sisältöön kuuluu muun muassa käytännön suunnittelua ja toteutusta lähiverkoista, kampusverkoista ja runkoverkoista oikeassa laboratorioympäristössä. Lisäksi opetellaan suunnittelemaan, kuinka asiakas kytketään verkkoon (ADSL, WiMAX, langaton LAN) ja ylläpitämään verkkoja. Tietoturvaan on myös suuresti panostettu hanketoiminnan kautta ja se on merkittävässä osassa opiskelua. Jyväskylän ammattikorkeakoulussa käynnistetään tammikuussa 2013 Informaatioteknologian koulutusohjelma, jonka sisältönä on kyberturvallisuus ja se tähtää ylempään AMK-tutkintoon. (Tietotekniikan koulutusohjelma 2012b.)

Koulutuksen aikana on mahdollisuus erikoistua tiettyyn erityisosaamisalueeseen, kolmantena vuonna on valittavissa neljästä eri osaamisalueesta kaksi. Tällä hetkellä valittavissa ovat verkkopalveluiden laadun hallinnan osaaminen, järjestelmien hallinnan ja tietoturvan osaaminen, CCNP osaaminen ja langattomien tietoliikennejärjestelmien osaaminen. Laadun hallinta kattaa keskeisimmät reaaliaikasovellukset, IP-QoS-mekanismien toiminnat ja runkoverkkojen palvelunlaatuun vaikuttavat tekniikat. Järjestelmien hallinta ja tietoturvaopinnoissa pyritään antamaan opiskelijalle valmiudet vastata yrityksen verkkojen ja palveluiden tietoturvasta, hallinnasta sekä ylläpidosta. Cisco Certified Network Professional (CCNP) on sertifikaattiin tähtäävä koulutus, ja se on osa Cisco Network Academyä, kansainvälistä opintokokonaisuutta, jonka on laatinut Cisco Systems. Koulutus antaa opiskelijalle verkkotekniikan ammattilliseen soveltamiseen vaadittavat erityistaidot. Langattomien tekniikoiden opintokokonaisuus antaa opiskelijalle valmiudet soveltaa tietojaan lisensoimattomien lyhyen kantaman ja lisensoitujen laajan kantaman langattomien tietoliikennejärjestelmien suunnittelussa ja toteutuksessa. (Tietotekniikan koulutusohjelma 2012b.)

1.1.3 JYVSECTEC

”JYVSECTEC - Jyväskylä Security Technology - on turvallisuusteknologian kehittämishanke, jonka osarahoittajina toimivat Keski-Suomen Liitto ja Euroopan aluekehitysrahasto. Hanke käynnistyi syyskuussa 2011 ja jatkuu vuoden 2013 loppuun.” (Jyvsectec 2012.)

Hankkeen tarkoituksena on kehittää kyberturvallisuuden kehittämisympäristö, jossa tuotetaan kehitys-, testaus- ja koulutuspalveluita yhteistyöverkoston käyttöön. Kehitysympäristöä tullaan hyödyntämään tiiviisti 2013 tammikuussa alkavassa kyberturvallisuuden ylemmässä ammattikorkeakoulututkinnossa. (Jyvsectec 2012.)

Kehitysympäristössä voidaan simuloida organisaatioiden prosesseja ja toimintamalleja, tarjota tietoturvan seuranta ja ylläpitopalveluja SecaaS-palveluna (Security as a Service) ja tehdä tietoturvavaikuttamisen tilannesimulointia. Hankkeen yhteydessä toteutetaan myös tilannekuvakeskus, joka mahdollistaa eri organisaatioiden yhteistyön ja toimintaprosessien testauksen ja simuloinnin. (Jyvsectec 2012.)

Ympäristössä olevilla työkaluilla voidaan myös toteuttaa yksittäisiä tietoturvaan liittyviä laite- ja ohjelmistotestauksia tai kokonaisia järjestelmätestauksia. Testit perustuvat standardimenetelmiin ja kriteeristöihin. Menetelmillä voidaan toteuttaa muun muassa haavoittuvuus-, penetraatio-, stressi- ja vaikuttamistestausta. Organisaatiolle voidaan toteuttaa myös tietoturvan hallintajärjestelmän evaluointeja eri osa-alueille. Lisäksi tuotetaan auditointipalveluja (laitteet, sovellukset, prosessit) mahdollistaen myös sertifioinnin eri osa-alueille. (Jyvsectec 2012.)

1.2 Työn tavoitteet

Opinnäytetyön tavoitteena oli tuottaa Jyväskylän ammattikorkeakoululle JYVSECTEC-hankkeelle PKI-infrastruktuuri, joka tarjoaa varmennepalveluja JYVSECTEC-toimintaympäristöön. Työssä käytettiin ilmaista Ubuntu Linux -käyttöjärjestelmää ja useita eri virtuaalikoneita. Työssä vertailtiin useita eri toteutustapoja, kuten OpenCA, Windows Server Certificate Services, EJBCA ja TinyCA. Vertailun pohjalta suunniteltiin ja toteutettiin ympäristö. Toimeksiantajan toivomuksesta suunniteltiin ja toteutettiin ilmainen ja kevyt järjestelmä, joka ei vie suurta määrää resursseja. Järjestelmästä tehtiin myös turvallinen ja tietyiltä osilta oikeaa tuotantoympäristöä vastaava kokonaisuus. Kyseessä oli kuitenkin simulaatioympäristöön tuleva vapaan lähdekoodin ohjelmistolla toteutettu järjestelmä, joka ei tuota ”oikeille” käyttäjille sertifikaatteja. Tästä syystä järjestelmää jouduttiin tietyiltä osin karsimaan käytettävyyden parantamiseksi.

Työn lopputuotteena tuli syntyä siis toimiva PKI-järjestelmä, joka tuottaa sertifikaatteja eri käyttötarkoituksiin. Järjestelmä dokumentoitiin perusteellisesti ja luotiin ohjeistukset sen käyttöön. Työn teoriaosuudessa käydään läpi työn kannalta olennaiset aiheet, joihin kuuluvat muun muassa kryptografia ja PKI.

Lopullinen työn dokumentaatio sisältää kaksi eri dokumenttia. Ainoastaan toinen dokumentti julkaistaan kaikkien saataville, joka on sanitoitu versio työstä. Kattavampi dokumentti tulee JYVSECTEC-hankkeen sisäiseen käyttöön.

2 KRYPTOGRAFIA JA SALAUS

2.1 Yleistä

Kommunikaatio on jo aikojen alusta ollut suuri osa ihmiskuntaa ja se on kehittynyt nopeasti. Nykypäivän kommunikaatiotapoja on monia, näistä esimerkkinä verkon ja mobiililaitteiden avulla tapahtuva kommunikaatio. Nopeasti kehittyvät tekniikat kasvattavat kuitenkin tietoturvariskiä sekä yksityisille käyttäjille että organisaatioille. Nykyisin onkin jo kehitetty runsaasti eri tapoja ja palveluja pienentää tätä riskiä. Eri keinoilla on kuitenkin tietyt perusvaatimukset, ja ne koostuvat seuraavista:

- **Luottamuksellisuus (confidentiality)**, prosessi, jolla pidetään tieto salaisena ja yksityisenä, jotta tiedon voi ymmärtää vain se henkilö, jolle se on tarkoitettu.
- **Eheys (integrity)**, tapa, jolla varmistetaan, että tieto pysyy muuttumattomana lähettäjältä vastaanottajalle. Kukaan ei siis voi muokata viestiä matkalla siten, että viestin peukalointia ei voitaisi todeta.
- **Saatavuus (availability)**, tapa, jolla varmistetaan, että tieto on saatavissa ja käytettävissä kohtuullisessa vasteajassa vain niille henkilöille, joille se on tarkoitettu.
- **Todennus (authentication)**, prosessi, joka tarjoaa todisteet lähettäjän henkilöllisyydestä vastaanottajalle, joten vastaanottaja pystyy todentamaan, että lähettäjä on se henkilö, joka hän väittää olevansa.
- **Kiistämättömyys (non-repudiation)**, tapa, joka varmistaa, että lähettäjä ei voi kiistää lähettäneensä viestiä. Esim. Alice lähettää viestin Bobille, täten hän ei voi myöhemmin kiistää lähettäneensä viestiä. (Choudhury 2002.)

2.2 Symmetrinen salaus

2.2.1 Yleistä

Symmetrinen salaus perustuu jaettuun salaiseen avaimeen jota käytetään sekä tiedon salaukseen että purkuun. Ongelmaksi muodostuu se, että lähettäjän ja vastaanottajan täytyy hyväksyä ja sopia yhteinen salainen avain. Avainten vaihtoon vaadi-

taan myös turvallinen kanava. Tämä hoidetaankin yleensä käyttäen hyväksi asymmetristä salausta. Symmetrisen salauksen etuna on sen nopeus, joten sitä suositellaan käytettäväksi suurien tietomäärien salaamiseen. (Choudhury 2002.)

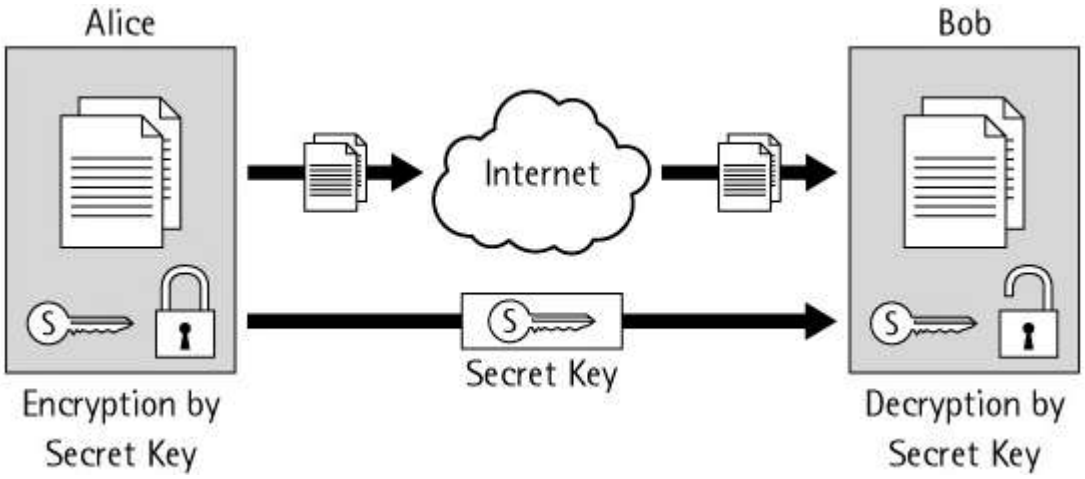
2.2.2 Algoritmit

Tunnetuimpiin ja käytetyimpiin symmetrisiin salausalgoritmeihin kuuluvat seuraavat:

- **Data Encryption Standard (DES)** on lohkosalaaja, jonka on kehittänyt IBM:n tiimi 1970-luvulla, ja se muodostuu 16 ns. Feistel-kierroksesta. Avaimen pituus on 56 bittiä ja lohkon koko 64 bittiä.
- **Triple-DES (3DES)** on yksinkertainen muunnos DES:stä, mutta se on huomattavasti tätä varmempi. Algoritmissa salataan kolmesti käyttäen DES:n algoritmia ja jokaiseen salaukseen voidaan käyttää eri avainta. Lohkon koko on sama kuin DES:llä, mutta avaimen pituus vaihtelee 112 ja 168 bitin välillä
- **Rivest Cipher 4 (RC4)** on jonosalaaja, jonka on kehittänyt Ron Rivest. Sen avaimen pituus voi olla jopa 2048 bittinen, ja se on tunnetusti erittäin nopea ja vahva salausalgoritmi. Algoritmin heikko kohta on, että käytetään samaa avainta kahden eri viestin salaamiseen. Joten sen paras hyöty saadaan, kun voidaan käyttää eri avainta jokaiselle viestille.
- **Advanced Encryption Standard (AES)** on DES:n seuraaja, jonka kehityksen tavoitteena oli saada aikaan julkinen ja kaikille saatavilla oleva tehokas lohkosalaaja. Avaimen pituus ja lohkon koko vaihtelee 128, 192 ja 256 bitin välillä. Sen etuna on suuri nopeusero ja vahvuus verrattuna DES:iin. (Choudhury 2002.)

2.2.3 Toimintaperiaate

Alice haluaa lähettää viestin Bobille ja varmistua siitä, että viestin voi lukea vain Bob. Turvatakseen tiedonsiirron, Alice loi salaisen avaimen ja salaa viestin tällä avaimella, jonka jälkeen hän lähettää viestin Bobille. Lukeakseen viestin, Bob tarvitsee salaisen avaimen, jonka Alice loi. Alicen täytyy siis vain toimittaa luotu avain Bobille valitsemallaan tavalla, jonka jälkeen viesti voidaan purkaa Bobin toimesta (ks. kuvio 1).



KUVIO 1. Symmetrinen salaus (Choudhury 2002)

2.3 Asymmetrinen salaus

2.3.1 Yleistä

Asymmetrinen salaus, jota kutsutaan myös nimellä epäsymmetrinen salaus, perustuu avainpariin: julkiseen avaimeen ja salaiseen avaimeen. PKI onkin eräänlainen toimintamalli julkisten avainten ja varmenteiden hallinnointiin, sillä menetelmää voidaan käyttää myös ilman PKI -infrastruktuuria taustalla. Etuna symmetrisiin algoritmeihin voidaankin pitää avaintenhallinnan yksinkertaisuutta, mutta toisaalta taas haittana on salauksen hitaus, joten se ei sovi kovin hyvin suurten tietomäärien salaamiseen. Asymmetristä salausta hyödynnetään usein symmetrisen salauksen tukena avainten vaihdossa. (Choudhury 2002.)

2.3.2 Algoritmit

Asymmetrisessä salauksessa käytetään pääasiassa seuraavia algoritmeja:

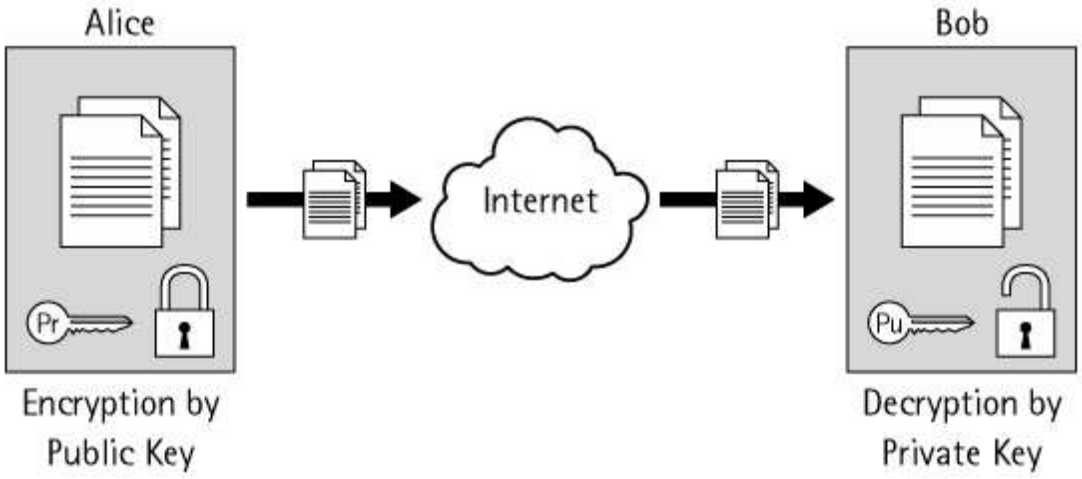
- **RSA** on yleisin ja käytetyin asymmetrinen salausalgoritmi, jonka kehittivät Ron Rivest, Adi Shamir ja Len Adleman vuonna 1977, ja sen nimi on peräisin heidän sukunimiensä alkukirjaimista. Julkinen ja salainen avain muodostetaan suurista alkuluvuista yksisuuntaisella modulaarifunktiolla. Turvallisuus perustuu siihen, että kyseinen funktio on erittäin vaikea ja aikaa vievä laskea toisinpäin johtuen suurten lukujen tekijöihin jakamisen vaikeudesta. Algoritmis-

sa avaimen pituus vaihtelee ja sen pituus ei ole suoraan verrannollinen symmetrisessä salauksessa käytettäviin pituuksiin, esim. 512-bittinen RSA on erittäin heikko.

- **Diffie-Hellman** on Whitfield Diffien ja Martin Hellmanin avaimenvaihtoon kehitetty algoritmi, jonka tarkoituksena on taata turvallinen symmetrisen avaimen jakelu. Se perustuu matemaattiseen funktioon, jonka avulla voidaan generoida yhteinen salaisuus ja tätä kautta salattu avain tiedon salausta varten. Diffie-Hellman on ensimmäinen julkisen avaimen salausmenetelmä. Algoritmi tarjoaa siis vain avaimen jakelun, mutta ei salausta tai digitaalista allekirjoitusta eli todennusta. Sen heikkous piileekin ns. ”man-in-the-middle”-tyyppisissä hyökkäyksissä.
- **El Gamal** on oikeastaan laajennus Diffie-Hellmanille. Sitä voidaan käyttää digitaalisiin allekirjoituksiin, salaukseen ja avaintenvaihtoon. Se perustuu diskreetin logaritmin ongelmaan. Vaikka El Gamal tarjoaa samat toiminnallisuudet kuin muut saman tyypin asymmetriset algoritmit, on sen suurin haitta hidas suorituskyky.
- **Elliptinen kurvi (ECC)** tarjoaa samat toiminnallisuudet kuin RSA: digitaalinen allekirjoitus, turvallinen avainten jakelu ja salaus. Algoritmi laskee diskreetit logaritmit elliptisistä kurveista. Elliptiset kurvit käyttävät vähän resursseja, koska ne toimivat hyvin lyhyilläkin avaimilla. Tästä syystä ne sopivat käytettäväksi hyvin pienillä laitteilla, joissa on rajoittuneet resurssit.
- **Digital Signature Algorithm (DSA)** on algoritmi, jota käytetään vain todentamiseen, sillä ei voi salata tietoa. (Harris 2010, 713 – 717.)

2.3.3 Toimintaperiaate

Molemmilla tietoa vaihtavalla osapuolella, Alice ja Bob, on hallussaan oma salainen avain ja julkinen avain. Julkinen avain voidaan jakaa vapaasti toiselle osapuolelle, mutta salainen avain pidetään tallessa. Kun Alice haluaa lähettää viestin Bobille, salaa hän viestin Bobin julkisella avaimella. Viestin voi purkaa vain Bobin salaisella avaimella, joten tiedonvaihto on turvallista eikä viestiä pysty tulkitsemaan muut kuin Bob (ks. kuvio 2). Lähettäessä viestiä toisesta suunnasta, tehdään vain sama toisinpäin Alicen avainparia käyttäen.



KUVIO 2. Asymmetrinen salaus (Choudhury 2002)

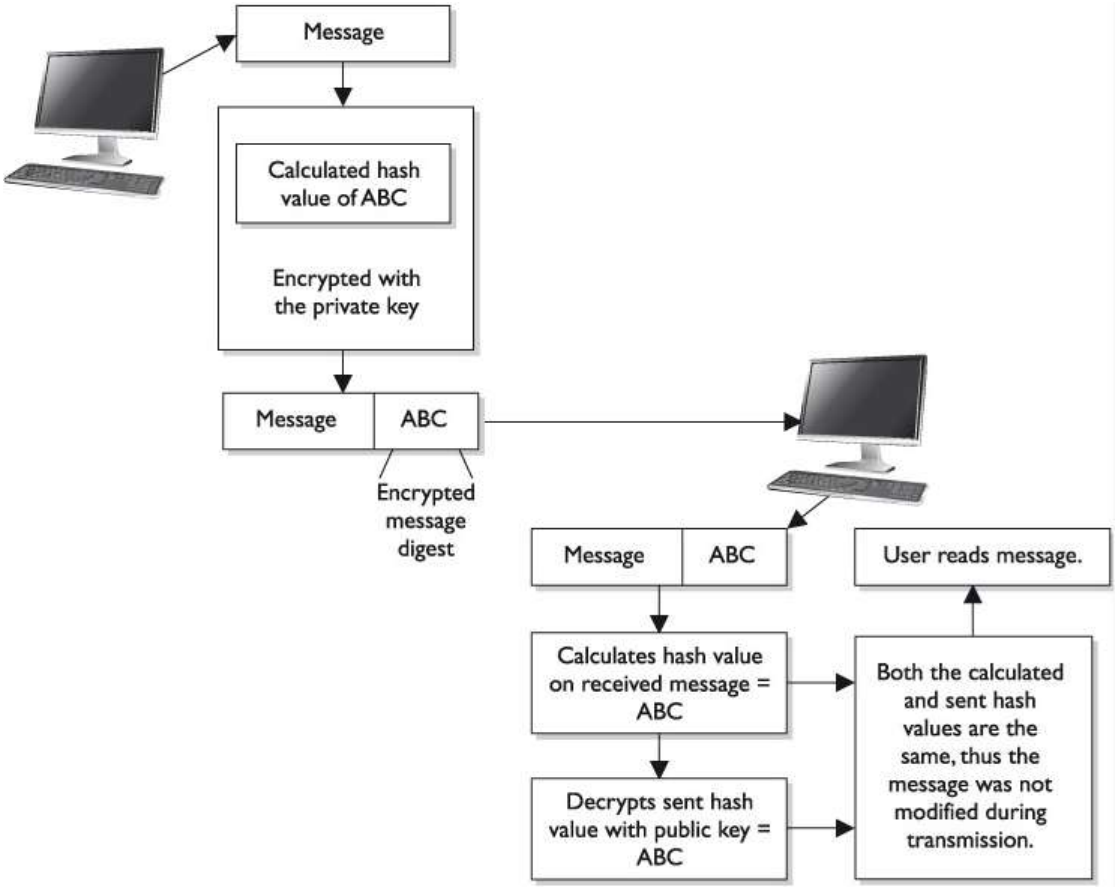
2.4 Tiiviste ja digitaalinen allekirjoitus

2.4.1 Tiivistefunktio

Tiiviste on tietyn pituinen merkkijono, joka on laskettu yksisuuntaisella funktiolla vaihtuvan pituisesta merkkijonosta. Sen turvallisuus perustuu siihen, että funktion laskeminen toisinpäin on erittäin vaikeaa ja aikaa vievää. Tiivistefunktiota hyödynnetään muun muassa salasanojen säilömisessä ja digitaalisessa allekirjoituksessa. (Harris 2010, 731.)

2.4.2 Digitaalinen allekirjoitus

Digitaalinen allekirjoitus on tiiviste, joka on laskettu lähetettävästä viestistä, salattu lähettäjän salaisella avaimella ja liitetty mukaan lähetettävään viestiin (ks. kuvio 3). Vastaanottaja purkaa salatun tiivisteen lähettäjän julkisella avaimella ja laskee itse tiivisteen viestistä, tämän jälkeen hän vertaa tiivisteitä keskenään ja mikäli tulos on sama, on viesti pysynyt muuttumattomana. Allekirjoitus vaatii siis julkisen avaimen menetelmän toimiakseen. Mikäli viestiin lasketaan pelkkä tiiviste mukaan, voidaan viestiä matkalla muuttaa ja laskea uusi tiiviste. Tässä tapauksessa ei vastaanottaja voi tietää, onko mukana oleva tiiviste alkuperäisen lähettäjän laskema. Allekirjoitus tarjoaa siis todentamisen, kiistämättömyyden ja eheyden. (Harris 2010, 731.)



KUVIO 3. Digitaalinen allekirjoitus (Harris 2010, 731)

2.4.3 Algoritmit

Tiivisteiden laskennassa käytetään pääasiassa seuraavia funktiota:

- **Message Digest 5 (MD5)** on Ron Rivestin kehittämä yksisuuntainen funktio, joka tuottaa 128 bittisen tiivisteen. Nykyään MD5 käyttöä ei suositella, koska se on todettu heikoksi ja alttiiksi törmäyshyökkäyksille.
- **SHA-1, SHA-256, SHA-384 ja SHA-512** on kehittänyt NSA, nämä ovat huomattavasti MD5:sta vahvempia. SHA-1 tuottaa 160 bittisen, ja muut versiot 256, 384 sekä 512 bittisen tiivisteen. (Harris 2010, 727 - 728.)

3 JULKISEN AVAIMEN INFRASTRUKTUURI

3.1 Yleistä

Julkisen avaimen infrastruktuuri, eng. Public key infrastructure (PKI), tarjoaa suojaamattoman julkisen verkon, kuten Internetin käyttäjille suojatun ja yksityisen tavan vaihtaa dataa. Tämä mahdollistetaan käyttäen julkista ja salaista kryptografista avainparia, jotka vastaanotetaan ja jaetaan luotetun kolmannen osapuolen kautta. Menetelmä tarjoaa digitaalisia sertifikaatteja eli varmenteita, joiden avulla voidaan tunnistaa muun muassa henkilöitä, organisaatioita, laitteita, ohjelmakoodia ja palveluja. PKI on teknologia, jota käytetään turvaamaan esimerkiksi elektroninen kaupankäynti julkisen Internetin yli käyttäen Secure Sockets Layer (SSL) ja Hypertext Transfer Protocol Secure Sockets (HTTPS) -protokollia. (Choudhury 2002.)

Julkisen avaimen infrastruktuurin toteutuksessa ei ole yhtä oikeaa ja parasta tapaa, eikä se määritä esimerkiksi, mitä salausalgoritmia pitää käyttää. PKI on eräänlainen kehys tai kokonaisuus, joka koostuu laitteista, ohjelmista, käytänteistä ja toimintamalleista, joita käytetään hallinnoimaan avaimia ja sertifikaatteja. (Choudhury 2002.)

3.2 Sertifikaatit

3.2.1 Yleistä

PKI hyödyntää sertifikaattipohjaista todennusta. Jotta varmistettaisiin globaali yhteensopivuus sertifikaattien välillä, täytyy niiden pohjautua standardiin. X.509 on ITU-T -standardi, ja se määrittää, mitkä kentät ovat pakollisia oikein määritetyssä sertifikaatissa, mitkä kentät ovat vaihtoehtoisia lisätietoja varten, miten kentät tulee allekirjoittaa ja miten allekirjoitettu data on koodattu. X.509 on tuettu monien aiheeseen liittyvien protokollien toimesta, kuten PEM, PKCS, S-HTTP ja SSL. (Vacca 2009, 444.)

Standardista on olemassa kolme eri versiota (ks. taulukko 1), joista yleisimmin käytetään versio kolmosta. Versioita 1 ja 2 käytetään lähinnä organisaatioiden intranetissä, koska ne ovat huomattavasti versio kolmosta rajoittuneempia. (Vacca 2009, 445.)

TAULUKKO 1. X.509-sertifikaatti

X.509-sertifikaatti	Ver. 1	Ver. 2	Ver. 3
Versio	x	x	x
Sarjanumero	x	x	x
Allekirjoitusalgoritmi	x	x	x
Myöntäjän nimi	x	x	x
Voimassaoloaika	x	x	x
Kohteen nimi	x	x	x
Kohteen julkinen avain ja algoritmi	x	x	x
Myöntäjän uniikki tunniste ID		x	x
Kohteen uniikki tunniste ID		x	x
Laajennokset			x
Myöntäjän digitaalinen allekirjoitus	x	x	x

3.2.2 Versio 1 ja 2

Versio 1 julkaistiin vuonna 1988 osana laajempaa X.500 hakemistostandardia. Versio 1 sisältää tarvittavat tiedot, jotka tunnistavat kohteen, mutta se ei takaa yksikäsitteisyyttä kohteen ja myöntäjän nimelle. Tällöin voi tulla tilanne, jossa kahdella eri kohteella voi olla sama nimi. Tästä syystä kehitettiin versio 2, johon lisättiin myöntäjän ja kohteen uniikki tunniste ID. Versio 2 ei kuitenkaan sisällä laajennoksia, jotka lisättiin seuraavaan versioon. Ensimmäisessä versiossa luottoketju muodostetaan siis myöntäjän nimi ja kohteen nimi -kentistä. Luottoketjun avulla yhdistetään myöntäjä sertifikaattiin. Toisessa versiossa luottoketjun muodostus helpottui lisättyjen tunnisteiden avulla ja kentät myös helpottivat sertifikaatin uusimista. (Vacca 2009, 445.)

3.2.3 Versio 3

Vuonna 1996 standardia uudistettiin ja julkaistiin versio 3, johon lisättiin vaihtoehtoinen laajennoskenttä, joka mahdollisti lisätietojen lisäämisen sertifikaattiin. Muutos oli merkittävä, ja se toi huomattavasti joustavuutta lisää sertifikaattien käyttöön. Ei ole olemassa yhtä sääntöjen mukaista hakemistoa laajennoksista, mutta yleisim-

min käytettyjä ovat S/MIME- ja SSL/TLS–toteutuksissa esiintyvät laajennokset. Näihin kuuluvat myöntäjän avaimen tunniste, kohteen avaimen tunniste, avaimen käyttötarkoitus ja kohteen vaihtoehtoinen nimi. (Vacca 2009, 445.)

Avaimen tunnisteen avulla voidaan tunnistaa esimerkiksi, mitä tiettyä avainta sertifikaatin myöntäjä on käyttänyt allekirjoittaessaan sertifikaatin. Näin ollen myöntäjä voi käyttää useaa eri salaista avainta allekirjoittaessaan sertifikaatteja. Tämä helpottaa tilannetta, jossa myöntäjän avain katoaa tai vaarantuu. (Vacca 2009, 445.)

Kohteen tunnisteen avulla on helppo tunnistaa, mitkä sertifikaatit kuuluvat tietylle avaimelle, jonka kohde omistaa. Kohteen tunnisteen avulla voidaan myös rakentaa luottoketjuja yhdistämällä se vastaavan myöntäjätunnisteen kanssa. (Vacca 2009, 445.)

Avaimen käyttötarkoitus voidaan myös määrittää laajennoskenttään, esimerkiksi eri avain salaukseen ja allekirjoitukseen. (Vacca 2009, 446.)

Kohteen vaihtoehtoiseksi nimeksi voidaan määrittää useita eri formaatteja, kuten sähköpostiosoite, DNS-nimi, URI tai IP-osoite. (Vacca 2009, 446.)

Laajennokset voivat sisältää myös sertifikaattikäytännöt, myöntäjän sertifikaattien julkaisupaikan ja sulkulistan sijainnin sekä mahdolliset rajoitukset sertifikaattien käyttöön. (Vacca 2009, 446 - 447.)

3.3 Arkkitehtuuri ja hierarkia

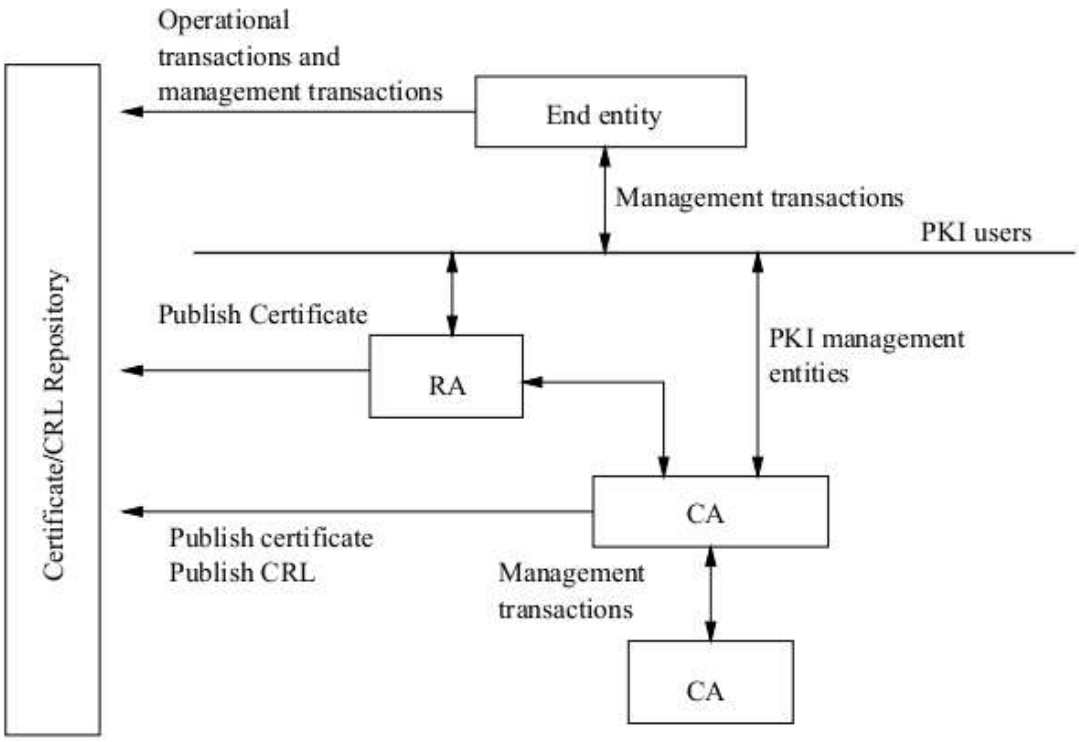
3.3.1 Yleistä

Julkisen avaimen infrastruktuuri on kehys, joka koostuu laitteista, ohjelmista, käytänteistä ja toimintamalleista, joita käytetään hallinnoimaan avaimia ja sertifikaatteja. Jotta tämä kehys on toimiva, tarvitsee se pohjalle ainakin seuraavat komponentit (Choudhury 2002):

- Certificate Authority (CA), eli sertifiointielin
- Registration Authority (RA), eli rekisteröintielin
- PKI asiakkaat/käyttäjät

- Digitaaliset sertifikaatit
- Sertifikaattien jakelujärjestelmä tai säilö (repository)

Kuviossa 4 on esitetty mahdolliset tiedonvaihdot eri elimien välillä riippumatta käyttöönotetusta rakenteesta. Loppukäyttäjä lähettää sertifikaattipyynnön RA:lle hyväksyttäväksi. Mikäli pyyntö hyväksytään, välitetään se CA:lle allekirjoitettavaksi. CA varmistaa pyynnön, ja mikäli varmistus menee läpi, tuotoksena syntyy allekirjoitettu sertifikaatti. CA lähettää allekirjoitetun sertifikaatin suoraan tai RA:n kautta säilöön, josta loppukäyttäjä voi sen noutaa. Kuviossa ilmenee, että loppukäyttäjän on mahdollista myös kommunikoida suoraan CA:n kanssa ilman RA:ta rajapintana. Tätä ei kuitenkaan suositella, mikäli sertifikaattipyyntöjä käsitellään paljon. (Xenitellis 2000, 35 - 36.)



KUVIO 4. PKI elimet (Xenitellis 2000, 36)

3.3.2 Certificate Authority (CA)

Sertifiointielin, eli sertifikaatin myöntäjä (CA) on luotettu kolmas osapuoli, joka todentaa ja asettaa luotetuksi eri osapuolet PKI-järjestelmässä. CA vastaanottaa sertifikaattipyyntöjä rekisteröintielimeltä (RA), joka on ensin hyväksynyt pyynnöt. Myön-

tääkseen sertifikaatin, täytyy CA:n allekirjoittaa vastaanotetut sertifikaattipyynnöt sen salaisella avaimella. (Choudhury 2002.)

Root CA, eli juurivarmentaja on korkeimmalla hierarkiassa. Juuri CA myöntää ja allekirjoittaa sertifikaatin itselleen, eli ns. self-signed sertifikaatin. Jokaisen PKI-järjestelmän luottoketju päättyy juurivarmentajaan. Juuri CA ei yleensä myönnä suoraan käyttäjille sertifikaatteja, vaan Juuri CA myöntää ainoastaan alemmille CA -tasoille sertifikaatit. Näitä alempia tasoja kutsutaan alivarmentajiksi (Sub CA). Usein Juuri CA on turvallisuussyistä ns. offline-tilassa, eli sillä ei ole Internet-yhteyttä. Mikäli Juuri CA:n salainen avain vaarantuu, on koko hierarkia vaarantunut ja jokaiselle joudutaan myöntämään uusi sertifikaatti. Jos puolestaan alivarmentajan avain on vaarantunut, Juuri CA voi evätä vain Ali CA:n sertifikaatin, jolloin pelkästään kyseisen Ali CA:n myöntämät sertifikaatit on evätty ja muut Ali CA:t pysyvät edelleen toiminnassa. (Choudhury 2002.)

Subordinate CA (Sub CA), eli alivarmentaja toimii aina alemmalla tasolla hierarkiassa kuin Juuri CA. Tasoja voi olla useampia, esimerkiksi 2- tai 3-tasoinen malli. 3-tasoisessa mallissa on siis ylimpänä Juuri CA, seuraavana alivarmentaja ja tämän alla vielä alivarmentaja. Alivarmentaja myöntää ja allekirjoittaa käyttäjille sertifikaatit, mutta kommunikoi yleensä rekisteröintielimen välityksellä. (Choudhury 2002.)

3.3.3 Registration Authority (RA)

Rekisteröintielin on vastuussa käyttäjien ja sertifiointielimen vuorovaikutuksesta. Se toimiikin eräänlaisena rajapintana näiden välillä, mutta joissain toteutuksissa käyttäjien on mahdollista kommunikoida myös suoraan CA:n kanssa. RA vastaanottaa sertifikaattipyyntöjä käyttäjiltä ja varmistaa hakijan henkilöllisyyden organisaation politiikan ja toimintatapojen mukaisesti, esimerkiksi käyttäjän täytyy käydä esittämässä kuvallinen henkilökortti, jonka jälkeen pyyntö joko hyväksytään tai hylätään. Mikäli pyyntö hyväksytään, välitetään se eteenpäin allekirjoitettavaksi CA:lle. RA myös vastaanottaa allekirjoitetut sertifikaatit CA:lta ja välittää ne edelleen oikealle kohteelle. (Choudhury 2002.)

RA helpottaa huomattavasti CA:n työtä, joten tätä kokoonpanoa suositellaan, mikäli pyyntöjä käsitellään paljon. RA:t mahdollistavat PKI sovelluksille hyvän skaalautuvuu-

den esimerkiksi eri maantieteellisiin osiin, sillä CA pystyy delegoimaan sen vastuita eri RA:lle eri alueilla. (Choudhury 2002.)

3.3.4 PKI asiakkaat/käyttäjät

Osapuolta, joka tekee sertifikaattipyynnön, kutsutaan usein PKI-asiakkaaksi tai käyttäjäksi (PKI Client). Kommunikaatio asiakkaan ja CA:n välillä täytyy olla salaista, eli esimerkiksi liikennöidään HTTPS:n yli. Asiakkaan vastuulla on pitää huolta omasta salaisesta avaimesta, koska avaimen kadotessa sillä ei voida enää purkaa salattuja viestejä ja pahimmassa tapauksessa vihamielinen käyttäjä voi vakoilla salattua liikennettä. Salaisen avaimen turvaaminen voidaan hoitaa monella eri tapaa, esimerkiksi käyttämällä rautatason ratkaisuja kuten älykorttia tai tokenia. (Choudhury 2002.)

3.3.5 Keskitetty ja hajautettu infrastruktuuri

PKI-infrastruktuurissa käytetyt avaimet, joita käytetään salaukseen ja todentamiseen, voidaan generoida ja säilöä joko keskitetysti (centralized) tai hajautetusti (decentralized). Hajautetussa mallissa avaimen generoinnin ja säilömisen hoitaa paikallisesti käyttäjän tietokoneella oleva sovellus, esimerkiksi selain. Keskitetyssä mallissa avaimet generoidaan ja säilötään keskitetysti palvelimella, josta avaimet lähetetään käyttäjille tarvittaessa. (Conklin, White, Williams, Davis & Cothren 2011, 132 - 133.)

Molemmissa malleissa on omat hyvät ja huonot puolensa. Jos organisaatio käyttää paljon resursseja vaativaa asymmetristä algoritmia ja suurta avaimen kokoa, tällöin käyttäjien omat tietokoneet eivät välttämättä ole tarpeeksi tehokkaita generoimaan avainta hyväksytyllä tavalla. Tässä tapauksessa kannattaa organisaation käyttää keskitettyä tehokkaampaa palvelinta avaimien prosessointiin, joka usein myös hyödyntää rautapohjaista satunnaislukugeneraattoria. (Conklin, White, Williams, Davis & Cothren 2011, 132 - 133.)

Keskitetty infrastruktuuri helpottaa myös avaimien varmuuskopiointia ja palautusprosessien käyttöönottoa. Palautusprosessin käyttöönotto erikseen jokaisella avainpareja säilövällä tietokoneella voi olla haastavaa. Mikäli työntekijä poistuu organisaatiosta, organisaatio ei välttämättä pääse enää käsiksi kyseisen työntekijän salaamiin organisaatiolle kuuluviin tietoihin. Keskitetyssä mallissa ongelmaksi muodostuu pal-

velimen saatavuus ja avaimien turvallinen lähetys niitä vaativille käyttäjille. Organisaation täytyy tehdä avaimia säilövästä palvelimesta tietoturvallinen ja vikasietoinen. Jos palvelin ei ole käytettävissä, käyttäjät eivät pääse avaimiinsa käsiksi ja tämä estää heiltä esimerkiksi todentamisen verkkoon, resursseihin ja sovelluksiin. Koska avaimet ovat yhdessä paikassa, palvelin on houkuttava kohde hyökkääjille. (Conklin, White, Williams, Davis & Cothren 2011, 132 - 133.)

Mikäli avainparia käytetään digitaaliseen allekirjoitukseen ja halutaan varmistaa kiistämättömyys, parempi vaihtoehto tälle on generoida avaimet hajautetusti käyttäjän tietokoneella. Tällöin varmistutaan siitä, että allekirjoitukseen käytetty avain on vain käyttäjän hallussa. (Conklin, White, Williams, Davis & Cothren 2011, 132 - 133.)

3.3.6 Yksitasoinen malli

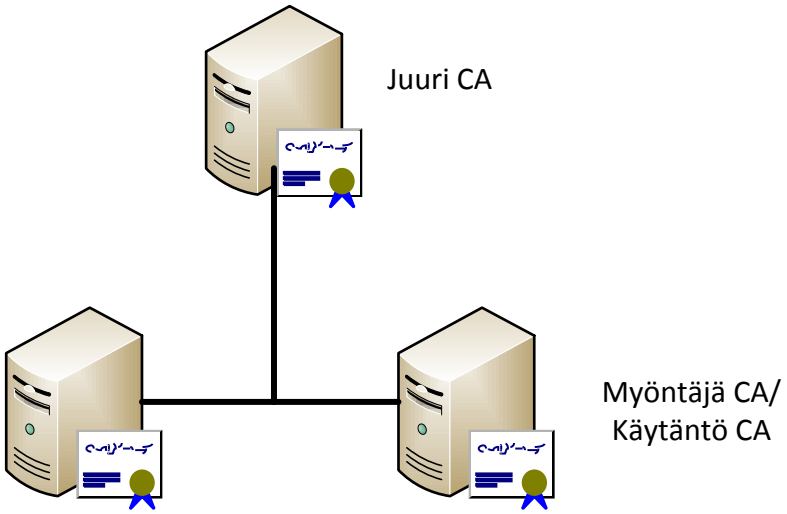
Organisaatiossa voidaan harkita yksitasoista hierarkiaa, mikäli halutaan ottaa käyttöön vain välttämättömimmät perustason PKI-palvelut. Tyypillisesti yksitasoista mallia käyttävillä organisaatioilla on vain alle 300 käyttäjää. Ratkaisussa on vain yksi Juuri CA, joka hoitaa kaikki PKI-palvelut. (Komar 2008.)

Järjestely ei ole kovin vikasietoinen eikä turvallinen. Mikäli CA jostain syystä vikaantuu, se ei pääse käsittelemään sertifikaattipyyntöjä ja allekirjoittamaan sertifikaatteja, eikä myöskään jakamaan sertifikaatteja ja sulkulistoja. Kyseistä tilannetta kutsutaan myös nimellä ”single point of failure”, eli jos yks piste vikaantuu, koko PKI-järjestelmän toiminta lamaantuu. Yksitasoisessa mallissa CA:n tulee myös olla jatkuvasti yhteydellisessä (online) tilassa, tämä on huono turvallisuuden kannalta ja vaikeuttaa salaisen avaimen turvaamista. Järjestelmän ainut CA on myös altis palvelunestohyökkäyksille. (Komar 2008.)

Yksitasoinen malli on kuitenkin helppo hallita ja ylläpitää, lisäksi sen kustannukset saadaan todella pieneksi. Eli todellisuudessa mallia kannattaa harkita, mikäli halutaan helppo hallittavuus ja alhaiset kustannukset, eikä organisaation tietoturvapolitiikka vaadi yhteydettömän (offline) CA:n käyttöönottoa. (Komar 2008.)

3.3.7 Kaksitasoinen malli

Kaksitasoinen hierarkia kostuu yhdestä yhteydettömässä tilassa olevasta juurivarmentajasta ja yhdestä tai useammasta yhteydellisessä tilassa olevasta alivarmentajasta (ks. kuvio 5). Alivarmentajat toimivat sekä myöntäjä CA:n että käytäntö CA:n roolissa. (Komar 2008.)



KUVIO 5. 2-tasoinen malli

Kaksitasoisen hierarkian tietoturva paranee huomattavasti verrattuna yksitasoiseen hierarkiaan. Juuri CA:n ollessa yhteydettömässä tilassa se on immuuni verkosta tuleville hyökkäyksille, joten fyysisen tietoturvan merkitys korostuu. Korkean tietoturvatason toteutuksissa Juuri CA:ta ei koskaan kytketä verkkoon, vaan tarvittaessa tiedonvaihto toteutetaan esimerkiksi optisen median avulla. (Komar 2008.)

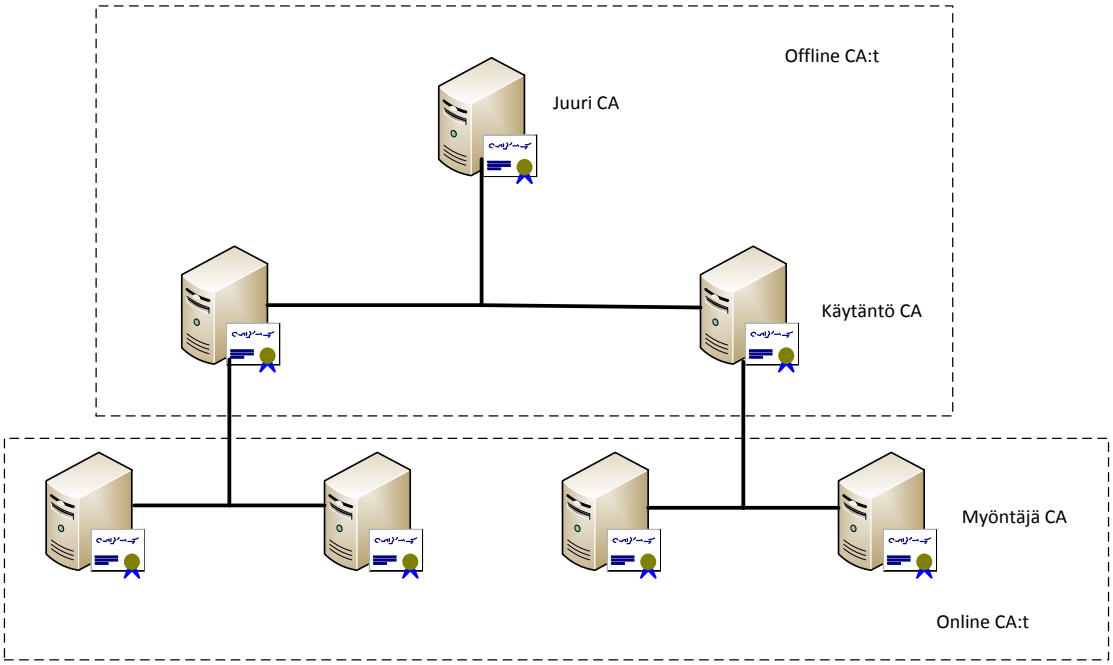
Monitasoisessa mallissa ei ole väliä vaikka toisen tason varmentaja myöntää sertifikaatteja tietokoneille, käyttäjille, palveluille ja verkkolaitteille. Tärkeintä on, että toisen tason varmentaja linkittyy luotettuun juurivarmentajaan, joka tässä tapauksessa on yhteydettömässä tilassa oleva juuri CA hierarkian yläpäässä. (Komar 2008.)

Vikasietoisuuden vuoksi toisella tasolla on hyvä olla vähintään kaksi alivarmentajaa. Toisen tason varmentajien määrä riippuu organisaation vaatimuksista, esimerkiksi voidaan luoda kaksi CA:ta samasta mallista varmistamaan palvelun toiminta vikatilanteessa. (Komar 2008.)

3.3.8 Kolmitasoinen malli

Kolmitasoinen hierarkia tarjoaa parhaan joustavuuden ja tietoturvan. Kolmitasoinen malli koostuu yhteydettömässä tilassa olevasta juurivarmentajasta ja yhdestä tai useammasta käytäntö CA:sta sekä yhteydellisessä tilassa olevasta yhdestä tai useammasta myöntäjä CA:sta (ks. kuvio 6). Myöntäjä CA:t jakavat sertifikaatit loppukäyttäjille. Yhteydetön tila hierarkian ylätasoilla tuo samat edut tietoturvan kannalta kuin kaksitasoisessa mallissa. Hierarkian ansiosta järjestelmä voidaan myös helposti jakaa maantieteellisten osien tai yrityksen rakenteen perusteella. (Komar 2008.)

Sertifikaatit myönnetään eri varmuustasojen (Assurance Level) mukaan, jotka vaativat erilaisia käytäntöjä. Mikäli vaaditaan eri menetelmiä sertifikaatin hakijan varmentamiseen, tarvitaan eri käytäntö CA:t 2-tasolle. Tässä tapauksessa vaaditaan myös eri varmennekäytäntö lausumat (CPS) esimerkiksi yrityksen työntekijöille ja asiakkaille. Eli jokainen käytäntö CA ottaa käyttöön oman varmennekäytäntö lausuman ja siihen liittyvät käytänteet ja varmuustasot. (Komar 2008.)



KUVIO 6. 3-tasoinen malli

3.3.9 Nelitasoinen malli ja siitä suuremmat

Joskus organisaatiossa vaaditaan enemmän kuin 3 tasoa. Tätä ei kuitenkaan suositella, koska rakenteen hallittavuus vaikeutuu. Nelitasoisessa mallissa kakkostasolla toimii käytäntö CA ja kolmostasolla järjestelmä on jaettu eri alueisiin. Nelostasolla on loppukäyttäjille tarkoitetut myöntäjä CA:t, jotka voivat olla jaettu vielä erikseen esimerkiksi työntekijöille ja aliurakoitsijoille. (Komar 2008.)

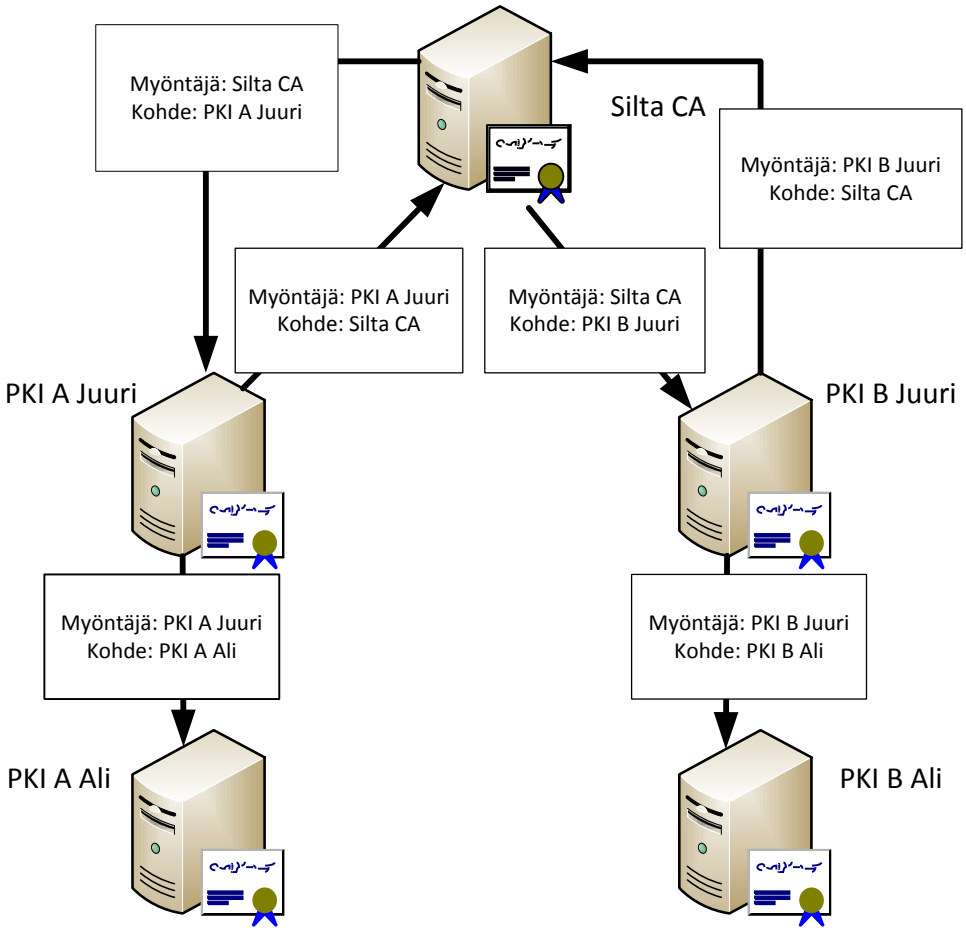
3.3.10 Ristiinsertifiointi

Ristiinsertifioinnin avulla voidaan luoda luottosuhde kahden eri organisaation välille. Eli toisen organisaation CA-rakenne liitetään osaksi omaa organisaatiota. Rakenne liittyy sen CA:n alapuolelle, joka sertifioinnin suorittaa. Ristiinsertifioinniksi määritetään kaikki, jossa sertifioinnin kohteena on toinen CA. Jos molemmat CA:t kuuluvat samaan toimialueeseen, esimerkiksi organisaation sisäisessä CA hierarkiassa, tätä kutsutaan toimialueen sisäiseksi ristiinsertifioinniksi (intradomain cross-certification). Eri toimialueiden välistä sertifiointia, jossa esimerkiksi organisaatio A:n CA sertifioi organisaatio B:n CA:n, kutsutaan nimellä toimialueiden väliseksi ristiinsertifioinniksi (interdomain cross-certification). (Adams & Lloyd 2003, 143 - 144.)

Ristiinsertifioinnin etuna on se, että jokaiselle käyttäjälle ei tarvitse myöntää uutta sertifikaattia. Uudelle järjestelmään liittyvälle organisaatiolle valitaan vain sopiva CA nykyisestä hierarkiasta, joka suorittaa ristiinsertifioinnin. Tämän jälkeen kaikki sertifikaatit myöntäjä CA:n alapuolella ovat luotettuja. Ristiinsertifiointi voi myös tapahtua kahteen suuntaan. Myöntävä osapuoli voi myös lisätä rajoitteita luottosuhteeseen. Esimerkiksi luotetaan vain tiettyihin käyttäjiin, ryhmiin ja käyttötarkoituksiin toisessa organisaatiossa. Sertifikaattilaajennoksien avulla voidaan määrittää, että luotetaan vain tiettyihin kohteisiin tietyn nimiavaruuden sisällä. Esimerkiksi luotetaan vain naapuriorganisaation myyntiyksikön sertifikaatteihin. Käytäntörajoitteiden (policy constraints) avulla voidaan hyväksyä vain tietyt käyttötarkoitukset sertifikaateille. Sertifikaattipolun pituusrajoitteilla (path-length constraints) voidaan rajoittaa ristiinsertifiointien määrää polun varrella, eli luottoketjun pituutta. Esimerkiksi CA 1 voi määrittää, että CA 2:n loppukäyttäjille myöntämät sertifikaatit ovat hyväksyttyjä, mutta CA2:n muut ristiinsertifioinnit eivät. (Adams & Lloyd 2003, 144 - 145.)

3.3.11 Silta CA

Suuren PKI-järjestelmän rakentamisessa saattaa ongelmaksi muodostua tavanomainen hierarkinen X.509-pohjainen PKI-malli. Tässä tapauksessa yksi ratkaisu voi olla silta CA:n käyttöönotto ilman hierarkista rakennetta, eli rakenne perustuu solmuihin (Mesh PKI). Solmupohjainen PKI-järjestelmä hyödyntää ristiinsertifiointia, jossa välittäjänä toimii erillinen silta CA. Käytännössä ristiinsertifioinnin jälkeen silta CA:sta muodostuu organisaatiolle ali CA. Ja koska ristiinsertifiointi tehdään kahteen suuntaan, toimii silta CA myös korkeimman tason CA:na organisaatiolle. Ratkaisun lopputuloksena kaksi eri organisaatiota liitetään yhteen ja ne voivat muodostaa luottoketjun silta CA:n välityksellä. Kuviosta 7 selviää miten luottoketju rakentuu organisaatioiden A ja B välillä. (Vacca 2009, 443.)



KUVIO 7. Kaksi eri PKI-järjestelmää liitetty yhteen silta CA:n avulla

3.4 Sertifikaattien jakelu ja sulkeminen

3.4.1 Sertifikaattien voimassaoloaika

Avaimilla ja sertifikaateilla tulee olla määritetty elinikä. Sertifikaatin tiedoista täytyy löytyä tarkka alku- ja loppuaika millä välillä sertifikaatti on voimassa. Sertifikaatti täytyy siis uusia tietyn ajan jälkeen. Mitä pidempi voimassaoloaika, sitä enemmän on aikaa vihamielisillä käyttäjillä murtaa avain ja mitä lyhyempi voimassaoloaika, sitä enemmän on hallinnointityötä. Voimassaoloaikaan ja avaimen murtamiseen vaikuttaa myös avaimen pituus, joten oikean voimassaoloajan määrittäminen on mietittävä tarkkaan. Hienostuneemmat PKI järjestelmät hyödyntävät automaattista ja usein käyttäjille näkymätöntä avaimen päivittämistä säästääkseen käyttäjien aikaa. (Conklin, White, Williams, Davis & Cothren 2011, 126.)

Kun sertifikaatin voimassaolo päättyy, täytyy huolehtia asianmukaisesta avaimien hävityksestä. Tarkoituksena on estää vanhojen avaimien väärinkäyttö, vihamielinen käyttäjä saattaa yrittää allekirjoittaa tai salata viestejä vanhalla avaimella ja täten varastaa jonkun toisen identiteetin. Vihamielinen käyttäjä saattaa myös yrittää murtaa järjestelmän avaimia ja täten saada lisätietoa käytetyistä menetelmistä vanhojen avaimien perusteella. (Conklin, White, Williams, Davis & Cothren 2011, 131 - 132.)

Moderneissa PKI-järjestelmissä avaimia tulee säilyttää myös niiden voimassaolon päättymisen jälkeen. Säilytyksen avulla varmistetaan, että käyttäjät voivat edelleen purkaa tiedon, joka on salattu vanhalla avaimella. (Conklin, White, Williams, Davis & Cothren 2011, 132.)

3.4.2 Sertifikaattien sulkeminen (Revocation)

Sertifikaattien sulkeminen (revocation) tarkoittaa eri asiaa kuin voimassaolon päättyminen. Sertifikaatti suljetaan, koska sen voimassaolo halutaan evätä ennen siihen merkattua virallista päivämäärää. Sertifikaatin sulkemiseen voi olla monta syytä, esimerkiksi käyttäjän salainen avain on murrettu, käyttäjä ei enää työskentele kyseiselle organisaatiolle, käyttäjä voi olla hukannut kannettavan tai älykortin jossa salainen avain oli säilössä tai käyttäjä on joutunut sosiaalisen hakkeroinnin (social en-

gineering) kohteeksi ja vahingossa luovuttanut salaisen avaimensa vihamieliselle käyttäjälle. (Conklin, White, Williams, Davis & Cothren 2011, 128.)

Sertifiointielin pitää suljetuista sertifikaateista yllä sulkulistaa (Certificate Revocation List, CRL), joka listaa kaikki suljetut sertifikaatit sarjanumeron perusteella. Listasta löytyy myös tieto miksi kyseinen sertifikaatti on suljettu ja sulkemisen tarkka ajankohta. CA on siis vastuussa luomiensa sertifikaattien tilasta. CA:lle täytyy kertoa sertifikaatin sulkemisesta ja sen täytyy tarjota tieto muille käyttäjille. Sulkulistan sijainti liitetään yhdeksi kentäksi laajennoksiin CA:n myöntämiin sertifikaatteihin. Sulkulistan laajennoskenttää kutsutaan nimellä CRL distribution point, eli sulkulistan jakelupiste. Jakelupiste sisältää esimerkiksi URL- tai LDAP-osoitteen josta löytyy hakemisto yhdelle tai useammalle sulkulistalle tietyn PKI järjestelmän sisällä. (Conklin, White, Williams, Davis & Cothren 2011, 129-130.)

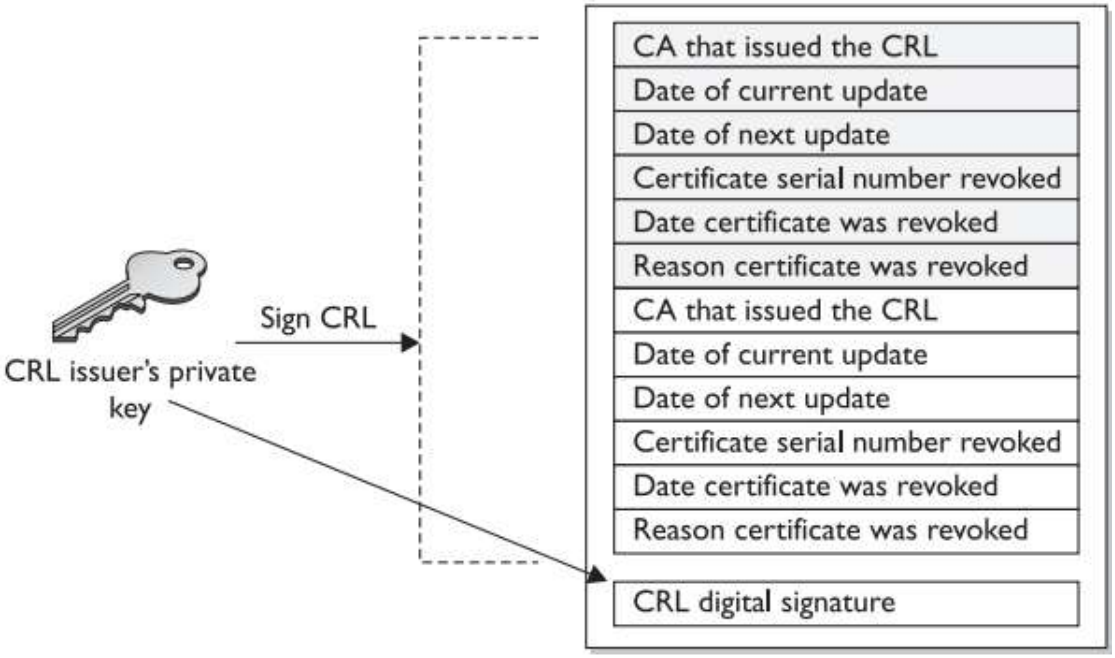
CRL tiedostoja voidaan joko hakea käyttäjien toimesta tai lähettää lista käyttäjille tietyin väliajoin. Sulkulista voi kasvaa isokokoiseksi, joten koko listan lähettäminen ei aina ole järkevää vaadittujen resurssien takia. Tästä syystä on mahdollista ensin ladata koko lista, ja seuraavilla kerroilla vain listaan tulleet muutokset. Tätä vajaata listaa kutsutaan myös nimellä delta CRL ja se vähentää huomattavasti vaadittua kaistaa päivityksissä. Suljettujen sertifikaattien tarkistusta helpottamaan on myös kehitetty verkkoprotokolla OCSP (Online Certificate Status Protocol), jonka avulla voidaan tehdä kyselyitä sertifikaatin tilasta sarjanumeron perusteella. (Conklin, White, Williams, Davis & Cothren 2011, 130 - 131.)

Jotta käyttäjät eivät voisi sulkea toinen toistensa sertifikaatteja ilman syytä ja näin aiheuttaa palvelunestohyökkäystä, täytyy sulkupyynnön tekijä todentaa. Todennus voidaan hoitaa esimerkiksi salasanalla, joka sovitaan rekisteröinnin yhteydessä. Todennus ei saa kuitenkaan perustua käyttäjän salaiseen avaimeen, koska tämä voi olla varastettu ja todellisuudessa CA todentaakin vihamielisen käyttäjän. (Conklin, White, Williams, Davis & Cothren 2011, 130 - 131.)

Sulkulistan eheys tulee myös varmistaa. Tällä taataan, että listaa voi muokata ainoastaan sitä ylläpitävä CA. Ilman eheyden varmistamista, vihamielinen käyttäjä voisi tehdä omia poistoja tai lisäyksiä listaan ja näin ollen jatkaa varastamansa salaisen avaimen käyttämistä tai aiheuttaa palvelunestohyökkäyksen. Eheys varmistetaan

CA:n digitaalisella allekirjoituksella (ks. kuvio 8). (Conklin, White, Williams, Davis & Cothren 2011, 129.)

Olennaisinta sulkulistan ylläpidossa on se, että kuinka ajantasaista siinä oleva tieto on, kuinka usein listaa päivitetään ja sisältääkö se kaikki suljetut sertifikaatit. Sulkulistan päivitysväli riippuu CA:sta ja sen varmennuskäytännöstä. (Conklin, White, Williams, Davis & Cothren 2011, 130.)



KUVIO 8. Sulkulistan allekirjoitus (Conklin, White, Williams, Davis & Cothren 2011, 130)

Sertifikaatin sulkeminen on lopullista. Tämä tarkoittaa sitä, että suljettua sertifikaattia ei ole enää mahdollista avata uudelleen, vaan tässä tapauksessa täytyy myöntää kokonaan uusi sertifikaatti. Joissain tapauksissa sulkemisen sijasta sertifikaatti voidaan ns. väliaikaisesti hyllyttää (suspend). Esimerkiksi käyttäjä lähtee pidennetylle lomalle ja haluaa varmistua siitä, että hänen sertifikaattia ei voida käyttää tänä aikana. Käyttäjä voi tehdä hyllytyspyynnön CA:lle ja CA listaa sertifikaatin sarjanumeroineen CRL-listalle, tällöin sulkemisen syy -kenttä ilmaisee syyksi väliaikainen hyllytys. Käyttäjän palatessa lomalta, hän ilmoittaa tästä CA:lle ja sertifikaatti poistetaan CRL-listalta. Toinen syy väliaikaiseen hyllyttämiseen voi olla, mikäli epäillään käyttäjän salaisen avaimen vaarantuneen. Tutkimusten ollessa käynnissä varmistetaan, että

sertifikaattia ei voida käyttää tänä aikana. (Conklin, White, Williams, Davis & Cothren 2011, 131.)

3.4.3 Online Certificate Status Protocol (OCSP)

Online Certificate Status Protocol on protokolla, joka kehitettiin vähentämään resurssien tarvetta ja tuottamaan ajantasaisempaa tietoa sertifikaattien tilasta. Koko sulkulistan lataaminen aina sertifikaatin tilan tarkistuksen yhteydessä ei ole resurssien kannalta järkevää. (Vacca 2009, 441)

OCSP:n idea ja toteutus ovat melko yksinkertaisia. Sertifiointielimen sertifikaatti sisältää viitteen OCSP-palvelimelle. Käyttäjä, joka haluaa tarkistaa tietyn sertifikaatin, lähettää sertifikaatin sarjanumeron, myöntäjän nimen tiivisteen ja kohteen nimen tiivisteen OCSP-palvelimelle. Palvelin tarkistaa sertifikaatin tilan ja palauttaa tästä tiedon käyttäjälle. Vastaus sisältää perustiedot, kuten sertifikaatin tilan, joka voi olla hyvä (good), suljettu (revoked) tai tuntematon (unknown). Vastaus sisältää myös usein nextUpdate -ajan, joka ilmaisee kuinka kauan käyttäjä voi pitää palvelimen vastausta kelvollisena. Mikäli sertifikaatti on suljettu, ilmoitetaan vastauksessa myös sulkemisen syy. (Vacca 2009, 441 - 442.)

Menetelmä siis poistaa tarpeen ladata koko sulkulista tietyn sertifikaatin tilan tarkistusta varten ja mahdollistaa lähes välittömän sertifikaattien sulkemisen sekä tiedon välittämisen käyttäjille. Menetelmän huonona puolena on, että se vaatii jatkuvan verkkoyhteyden käyttäjille kommunikoidakseen palvelimen kanssa. (Vacca 2009, 441 - 442.)

3.4.4 Sertifikaattisäilö (Repository)

Kun sertifikaattipyyntö on käsitelty ja sertifikaatti on myönnetty, täytyy se julkaista kaikkien saataville. PKI-ympäristön sisällä kommunikointi ei onnistu ilman osapuolten julkista avainta. Avaimet ja näille kuuluvat sertifikaatit julkaistaan yleensä keskistetyssä säilössä (repository). Säilö on tarkoitettu vain tietyn PKI-ympäristön sertifikaattien ja julkisten avainten säilyttämiseen. Säilö voi olla esimerkiksi HTML-käyttöliittymä, josta sertifikaatit on mahdollista noutaa ja se on yleensä myös LDAP-hakemistoprotokollan kanssa yhteensopiva. Kun käyttäjä kommunikoi toisen osapuo-

len kanssa, voi lähettäjä lähettää oman sertifikaattinsa julkisen avaimen kera vastaanottajalle, tämä mahdollistaa sen, että sertifikaatteja ei aina tarvitse etsiä ja hakea säilöstä. Jos lähettäjä haluaa salata ensimmäisen viestin vastaanottajan julkisella avaimella, täytyy sertifikaatti hakea ensin säilöstä. (Conklin, White, Williams, Davis & Cothren 2011, 118.)

Säilössä ei ole yhtä suuret turvallisuusvaatimukset kuin itse CA:lla. Säilössä ei säilytetä salaisia avaimia ja kun sertifikaatit on allekirjoitettu CA:n toimesta, näihin tehdyt muutokset voidaan helposti havaita ja hylätä peukaloitu varmenne. (Conklin, White, Williams, Davis & Cothren 2011, 118.)

3.4.5 Simple Certificate Enrollment Protocol (SCEP)

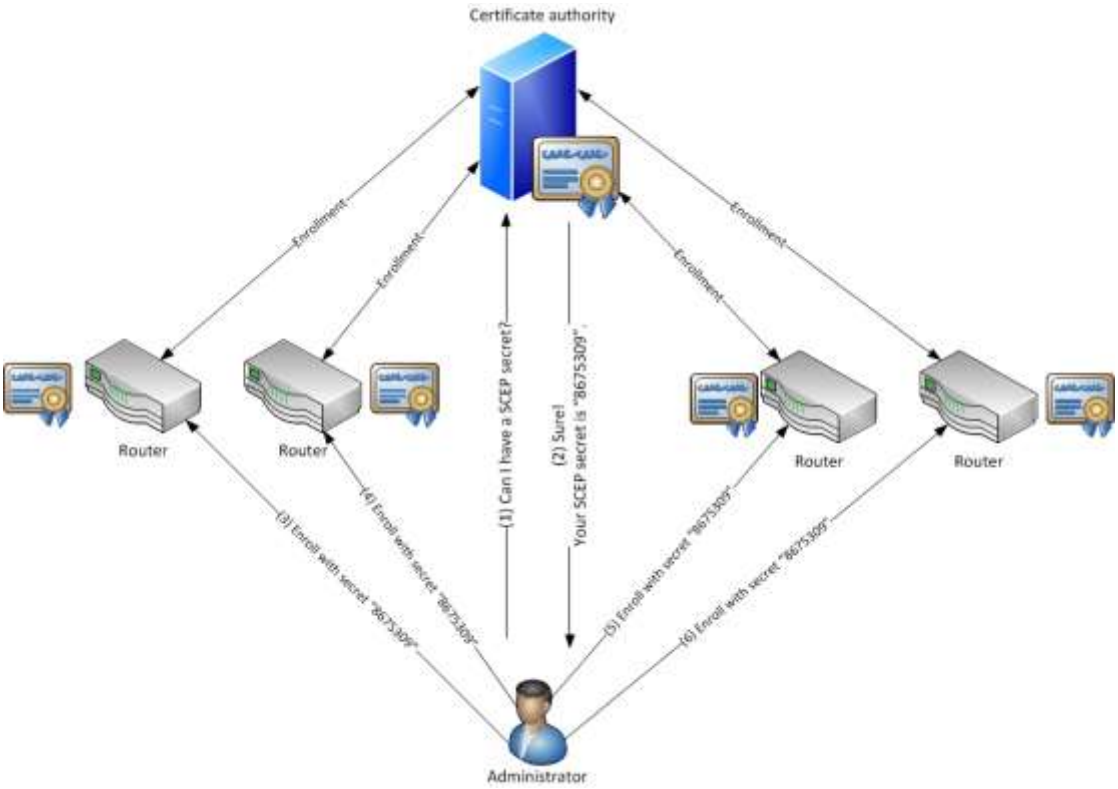
Simple Certificate Enrollment Protocol (SCEP) on Ciscon kehittämä protokolla yksinkertaisempaan laitesertifikaattien enrollaukseen. Tarkoituksena on turvallinen ja skaalautuva sertifikaattien myöntäminen esimerkiksi verkkolaitteille. Protokolla tukee seuraavia toimintoja (Liu, Madson, Nourse & Vilhuber 2009, 4 - 5.):

- CA:n ja RA:n julkisen avaimen levitys
- Sertifikaatin enrollaus
- Sertifikaattikysely (Certificate query)
- Sulkulistakysely (CRL query)

SCEP on de-factor standardi sertifikaattien myöntämiseen mobiililaitteille ja se on tärkeä osa kun organisaatio ottaa käyttöön BYOD (Bring Your Own Device)-ympäristöä. Monet organisaatiot käyttävät sertifikaattipohjaista todennnusta mobiililaitteille. Sertifikaattien avulla on helppo kontrolloida laitteiden pääsyä organisaation resursseihin, kuten sisäverkkoon, sähköpostiin, SharePoint:iin, virtuaalityöpöytiin ja web-sovelluksiin. (Diodati 2012.)

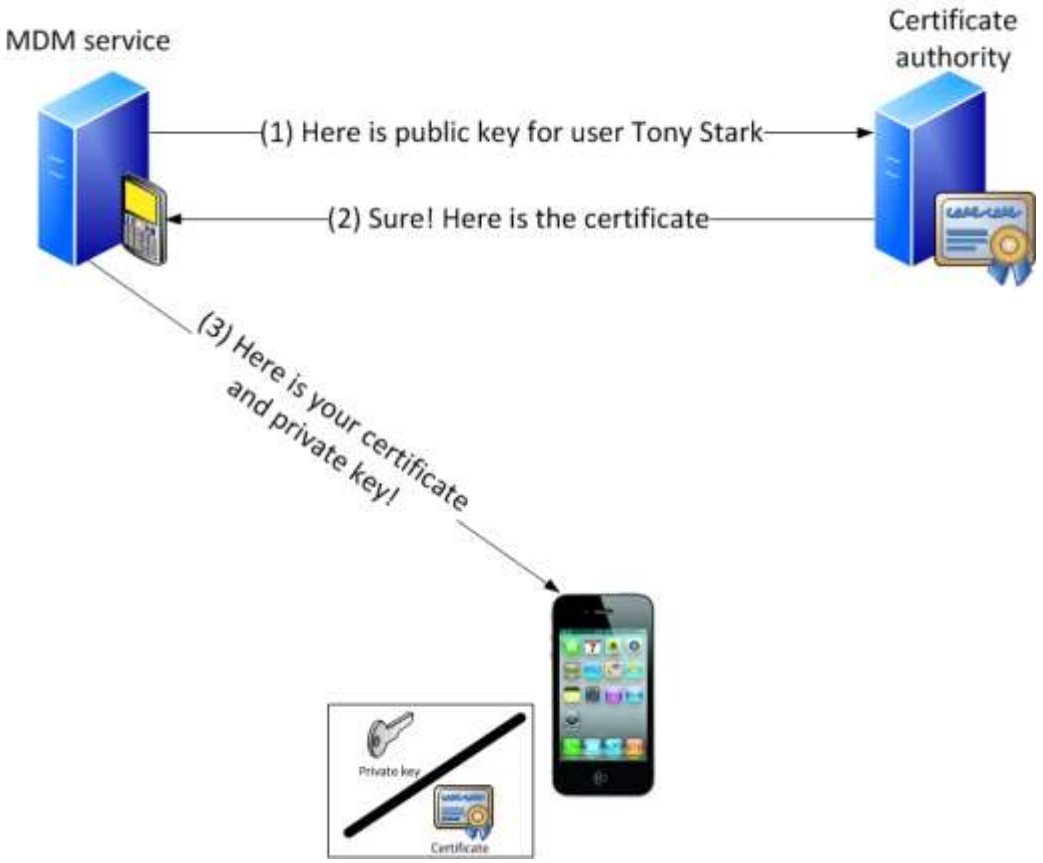
Kuviosta 9 selviää SCEP-protokollan toimintaperiaate. Järjestelmänvalvoja pyytää SCEP:iä tukevalta CA:lta jaetun salaisuuden. Sertifikaatin hakija, joka tässä tapauksessa on reititin, käyttää jaettua salaisuutta valtuutuksena hakuprosessissa, jotta uusi sertifikaatti saadaan enrollattua onnistuneesti. Toimintaperiaate voi vaihdella mobiililaiteympäristössä, jossa on usein mukana myös mobiililaitteiden hallintapalvelin

(MDM) ja/tai identiteettitinhallintajärjestelmä, jotka voivat toimia myös ns. SCEP-proxynä. (Diodati 2012.)



KUVIO 9. SCEP-toimintaperiaate (Diodati 2012.)

SCEP:n käyttöönotossa on syytä ottaa huomioon tietoturva. Samaa jaettua salaisuutta ei kannata käyttää kaikille laitteille, koska tämä mahdollistaa sertifikaatin hakemisen väärällä nimellä. Erityisesti tällä on vaikutusta kun ollaan tekemisissä mobiililaitteiden kanssa. Paras tapa on käyttää esimerkiksi MDM-järjestelmää SCEP-proxynä (ks. kuvio 10). (Diodati 2012.)



KUVIO 10. MDM-järjestelmä SCEP-proxynä (Diodati 2012.)

3.5 Varmennepolitiikka (CP)

Certificate policy (CP), eli varmennepolitiikka on dokumentoitu joukko sertifiointielimen laatimia sääntöjä ja sitoumuksia, jotka ilmaisevat sertifikaattien sopivuutta tietyille sovelluksille tai käyttäjäryhmille. Eli toisin sanoen voidaan arvioida varmenteen soveltuvuus tiettyyn käyttötarkoitukseen. (Choudhury 2002.)

Varmennepolitiikan päätarkoitus on kuitenkin määrittää tietoturvapolitiikka, jota kyseinen sertifiointiorganisaatio noudattaa. Sitä voidaan käyttää myös referenssinä muille organisaatioille luottosuhteen muodostuksen yhteydessä. Varmennepolitiikka ei kuvaa tarkasti miten se on otettu käyttöön, vaan se on eräänlainen organisaation sitoutuminen varmenteiden tietoturvan parantamiseen. Koska varmennepolitiikka on vain pintapuolinen kuvaus, se on sovellusriippumaton ja sitä voidaan käyttää laajasti sekä pidemmän aikaa eri sovelluksiin. Varmennepolitiikka voi kattaa yhden tai useamman CA:n, jotka myöntävät sertifikaatteja kyseistä politiikkaa noudattaen. (Choudhury 2002.)

Käytetty varmennepolitiikka liitetään sertifikaattiin OID-tunnistekentän (Object Identifier) avulla. OID on yksinkertainen tietotyyppi, jonka määrittää ASN.1-standardi. OID koostuu sarjasta kokonaislukuja, jotka on eroteltu pisteellä, esimerkiksi 2.3.4.5.2. Tietty tunniste viittaa tiettyyn varmennepolitiikkaan, joka puolestaan liittyy yhteen tai useampaan varmennekäytäntö lausumaan. Kun politiikka liitetään tunnisteeseen, siihen ei voi tehdä enää muutoksia lisäämättä uutta tunnistetta. Tunnisteiden avulla on helppo erotella eri politiikat toisistaan. Varmennekäytäntö lausuma (CPS) liittyy epäsuorasti OID-tunnisteeseen. Esimerksiksi organisaatiolla on kolme eri osastoa: myynti-, valmistus- ja lakiosasto. Lakiosasto ei millään lailla liity kahden muun toimintaan, joten sille on luotu oma varmennepolitiikka ja varmennekäytäntö lausuma. Myynti- ja valmistusosaston toiminnot ovat osittain samanlaisia, joten näillä on yhteinen varmennepolitiikka. Jokainen osasto suorittaa kuitenkin tiettyjä erikoistoimintoja, joten jokaiselle osastolle täytyy luoda oma varmennekäytäntö lausuma. Myynti- ja valmistusosaston lausuma luodaan saman varmennepolitiikan pohjalta, mutta lakiosastolle eri politiikasta. Organisaatio tarvitsee siis kaksi eri OID-tunnistetta kahdelle varmennepolitiikalle, joihin erilliset lausumat ovat liitetty. (Choudhury 2002.)

3.6 Varmennekäytäntö lausuma (CPS)

Certification Practice Statement (CPS), eli varmennekäytäntö lausuma on sertifiointielimen tarjoama dokumentti, joka määrittää elimen käytänteet sertifikaattien myöntämiseen ja hallinnointiin. Varmennekäytäntö lausuma voi sisältää käytänteet muun muassa sertifikaattien myöntämisestä, uusimisesta, sulkemisesta, arkistoinnista ja julkaisusta. CPS:n avulla eri elimien ja käyttäjien on helpompi arvioida CA:n luotettavuutta. Mitä tarkemmin eri toimintamallit on dokumentoitu, sitä helpompi luotettavuutta on arvioida. (Choudhury 2002.)

Varmennekäytäntö lausuma eroaa varmennepolitiikasta (Certificate Policy) siten, että varmennekäytäntö lausuma määrittää tarkan kuvauksen kuinka varmennepolitiikassa määritetyt prosessit ja toiminnot otetaan käyttöön sekä kuinka niitä valvotaan. Kuvaus sisältää sovelluskohtaisia lisätietoja, joita organisaatio pitää oleellisena kyseiseen dokumenttiin liittyen. (Choudhury 2002.)

CPS:n tulisi noudattaa varmennepolitiikkoja, joita kyseinen CA tukee. Usein käytetäänkin samaa mallipohjaa varmennepolitiikalle ja varmennekäytäntö lausumalle. Varmennekäytäntö lausuma on kuitenkin huomattavasti tarkempi ja yksityiskohtaisempi kuin varmennepolitiikka. Varmennekäytäntö lausuma voi sisältää muun muassa seuraavia tietoja (Choudhury 2002):

- CA, jolle kyseinen CPS kuuluu
- CA:n määrittelemät varmennepolitiikat
- Politiikka sertifikaattien myöntämiselle ja uusimiselle
- CA:n sertifikaatin voimassaoloaika
- Ehdot, joiden perusteella CA voi sulkea käyttäjän sertifikaatin
- Sulkulistaan liittyvät politiikat, jotka määrittävät muun muassa julkaisuvälin ja jakelupisteen
- CA:n sertifikaatin käyttämät algoritmit.

RFC 3647:n määrittelemä rakenne (Chokhani, Ford, Sabett, Merrill & Wu 2003):

1. Esittely
2. Julkaisu ja säilö
3. Tunnistaminen ja autentikointi
4. Sertifikaatin elinkaaren toiminnalliset vaatimukset
5. Fyysiset, hallinnolliset ja toimintaa koskevat vaatimukset
6. Tekniset turvallisuusvaatimukset
7. Sertifikaatin, sulkulistan ja OCSP:n profiili
8. Vaatimustenmukaisuuden arviointi (Compliance Audit)
9. Muut liiketoiminnalliset ja oikeudelliset asiat

3.7 Salaisen avaimen varmistus ja suojaus

3.7.1 Yleistä

Monet PKI toteutukset ovat monimutkaisia ja koostuvat useista eri komponenteista. Yksi tärkeimmistä komponenteista on käyttäjien salaiset avaimet, jotka tulee pitää salassa aina. Kun salainen avain luodaan ensimmäisen kerran, tulee se säilöä johon-

kin myöhempää käyttöä varten. Säilöä kutsutaan usein avainsäilöksi (key store). Avainsäilö voi olla esimerkiksi internet-selain, älykortin sovellus tai jokin muu sovellus. Usein sovellus pyytää käyttäjältä salasanan, jolla avainsäilö salataan ja suojataan. Esimerkiksi Alice tekee sertifikaattipyynnön internet-selaimella ja generoi salaisen avaimen selaimen avainsäilöön. Selain pyytää Alicelta salasanan, jota sovellus käyttää avainsäilön salaamiseen. Kun Alicen tarvitsee käyttää salaista avaintaan, häneltä kysytään määritettyä salasanaa avainsäilön salauksen purkuun. Avaimien käsittelyssä on syytä ottaa huomioon myös suojaus, varmuuskopiointi ja palautus. (Conklin, White, Williams, Davis & Cothren 2011, 134.)

3.7.2 Suojaaminen

Avainsäilön heikkoutena on usein käyttäjän määrittämä heikko salasana. Jotkin sovellukset eivät välttämättä vaadi salasanaa lainkaan. Tässä tapauksessa vihamielisen käyttäjän tarvitsee vain päästä fyysisesti käyttäjän tietokoneelle väärinkäyttääkseen salaista avainta. Paras tapa on suojata avainsäilö vahvan salasanan lisäksi esimerkiksi älykortilla ja PIN-koodilla. (Conklin, White, Williams, Davis & Cothren 2011, 134.)

Salaisen avaimen suojauksessa tulee ottaa huomioon myös avaimen pituus, jonka tulee tarjota vaadittava suojaustaso ympäristöön. Avaimen elinikää määrittäessä tulisi ottaa huomioon, että kuinka usein sitä käytetään ja kuinka arkaluontoista tietoa se suojaa. Avainta ei tulisi myöskään käyttää sen voimassaoloajan jälkeen. Avain tulee myös tuhota oikeaoppisesti voimassaoloajan jälkeen, mutta tässä tulee ottaa huomioon sen tarpeellisuus vanhojen salauksien purkamisessa. Avainta ei saa koskaan paljastaa selväkielisenä, vaan se tulee säilyttää turvallisesti salattuna avainsäilössä ja avainta säilövien sovelluksien tietoturvan taso tulee varmistaa. Kiistämättömyyden takaamiseksi, avaimesta ei saa tehdä kopioita, eikä sitä saa jakaa muille. (Conklin, White, Williams, Davis & Cothren 2011, 134 - 135.)

3.7.3 Varmuuskopiointi ja palautus

Henkilöllä voi olla yksi tai useampia avainpareja käytössään. Nämä avainparit ovat sidottu henkilön identiteettiin ja niillä voi olla useita eri vaatimuksia ja käyttötarkoituksia. Sertifikaatteihin on mahdollista liittää attribuutteja ja käyttörajoitteita, jotka määrittävät mihin tarkoitukseen sertifikaattia voidaan käyttää. Esimerkiksi käyttäjällä

voi olla erillinen avainpari allekirjoitukseen, tiedon salaamiseen ja avaintenvaihtoon. Käyttäjällä voi olla myös erilliset allekirjoitukseen tarkoitetut avainparit sekä työ käyttöön että henkilökohtaiseen käyttöön. (Conklin, White, Williams, Davis & Cothren 2011, 135.)

Mikäli organisaatio varmuuskopioi käyttäjien avaimia, tulisi tämä tehdä ainoastaan avainpareille, joita käytetään tiedon salaamiseen. Kun allekirjoitukseen tarkoitettuja avainpareja ei varmuuskopioida organisaation toimesta, varmistetaan kiistämättömyys. Varmuuskopioimalla salaukseen tarkoitetut avainparit, organisaatio turvaa tärkeän tiedon saatavuuden, vaikka avainparin omistaja jostain syystä katoaisi. (Conklin, White, Williams, Davis & Cothren 2011, 135.)

Avaimen varmuuskopiointiin ja palautukseen liittyy kaksi tärkeää järjestelmää: avaimien arkistointijärjestelmä ja avaimien palautusprosessi. Arkistointijärjestelmä on keino varmuuskopioida ja säilöä avaimet turvallisesti talletuspaikkaan. Mikäli avaimet säilötään keskitettyyn talletuspaikkaan, tulee tämä suojata asianmukaisesti. Jos hyökkääjä onnistuu pääsemään käsiksi talletuspaikkaan, on koko infrastruktuuri vaarantunut. Avaimien palautusta ei pidä myöskään luottaa yhden henkilön käsiin. Palautusprosessi on järkevä määrittää siten, että avaimen palauttamiseen vaaditaan ainakin kaksi henkilöä. Henkilöt tunnistautuvat palautussovellukselle ennen toimenpiteen suorittamista. Tätä menetelmää kutsutaan nimellä usean henkilön läsnäolo (dual control), joka tarkoittaa, että vaaditaan kaksi henkilöä toimenpiteen suorittamiseen. Palautusprosessi voidaan myös määrittää siten, että toimenpiteeseen vaaditaan m henkilöä n:stä, jossa m on pienempi kuin n. Tämä estää yksittäisten henkilöiden tai pienen ryhmän oikeuksien väärinkäytön. Toimenpiteeseen ei kuitenkaan tarvita kaikkia henkilöitä (n), koska kaikki ovat harvoin saatavilla samaan aikaan. (Conklin, White, Williams, Davis & Cothren 2011, 136.)

Kaikkia avaimen palautukseen liittyviä toimenpiteitä tulee myös valvoa asianmukaisesti. Valvontatapahtumat tulee sisältää ainakin palautetut avaimet, mukana olleet henkilöt ja aikaleimat. (Conklin, White, Williams, Davis & Cothren 2011, 136.)

3.7.4 Avaimen turvatalletus (Key Escrow)

Avaimen turvatalletus on prosessi, jossa avaimet luovutetaan luotetulle kolmannelle osapuolelle, joka voi tarvittaessa purkaa ja lukea salattua tietoa. Kolmas osapuoli voi olla esimerkiksi poliisi, tai jokin muu korkean tason elin. Avaimia voidaan esimerkiksi käyttää todisteiden hankkimiseen rikosoikeudellisissa tutkimuksissa. (Conklin, White, Williams, Davis & Cothren 2011, 137.)

Varmasti kuuluisin turvatalletushanke on 90-luvun loppupuolella Yhdysvalloissa ajettu Clipper Chip-hanke. Tarkoituksena oli, että kaikki Yhdysvalloissa valmistetut kommunikaatiolaitteet sisältävät salaamiseen tarkoitetun rautatason sirun. Sirua olisi voitu käyttää tiedon salaamiseen kahden henkilön välillä. Mikäli hallituksen osasto haluaisi salakuunnella keskustelua, sen tarvitsisi hakea vain oikeuden määräys. Jos määräys hyväksytään, lainvalvoja hankkii tarvittavat palaset avaimesta kolmansilta osapuolilta. Yhdistämällä avaimen palaset voidaan avainta käyttää salatun yhteyden salakuunteluun. Standardi ei koskaan yleistynyt, koska sitä pidettiin liian tehokkaana valvontakeinona amerikkalaisille. (Conklin, White, Williams, Davis & Cothren 2011, 137.)

4 ERI TOTEUTUSVAIHTOEHTOJEN ESITTELY

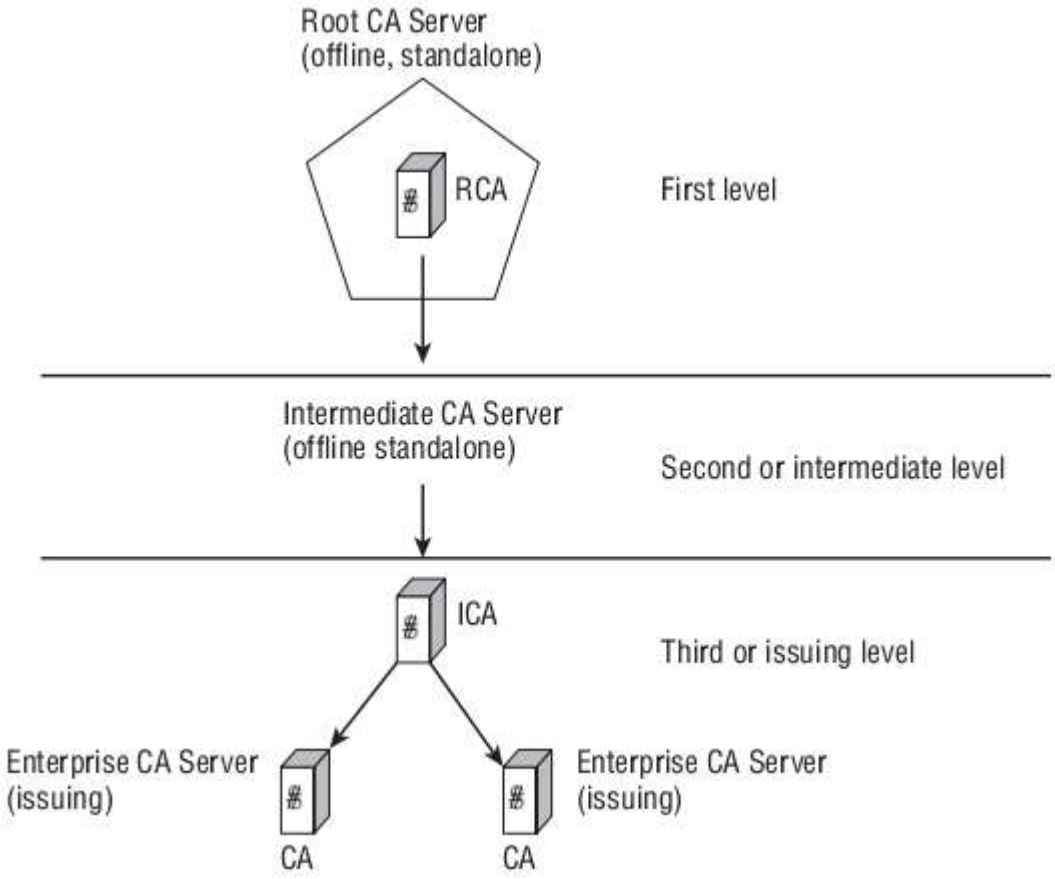
4.1 Windows Server Certificate Services

Vertailuun otettiin mukaan yksi johtava kaupallinen PKI-ratkaisu, jotta saadaan parempi kokonaiskuva ilmaisten ja kaupallisten tuotteiden välillä.

Windows Server 2008 sisältää vahvan PKI -järjestelmän ja tarjoaa integroidun valikoiman työkaluja ja palveluita sen hallintaan. (Shapiro 2008, 583.)

Microsoftin tarjoama PKI -ratkaisu toimii yhteen aktiivihakemiston kanssa (Active Directory), joka julkaisee tietoa myönnetyistä avaimista ja varastoi sertifikaatit, sulkulistat ja tietoa käytänteistä. (Shapiro 2008, 572.)
Windows Serverissä CA on mahdollista asentaa joko Standalone CA:ksi tai Enterprise

CA:ksi. Standalone CA toimii offline-tilassa ja se ei liity aktiivihakemistoon, Standalone CA on yleensä juuri CA tai väli CA. Enterprise CA liitetään mukaan aktiivihakemistoon ja se toimii yleensä alivarmentajana (ks. kuvio 11).



KUVIO 11. Windows Server Certificate Services -hierarkia (Shapiro 2008, 587.)

Microsoftin PKI -ratkaisu on selkeästi tarkoitettu toimivaksi aktiivihakemiston kanssa, vaikkakin CA on mahdollista asentaa myös standalone -moodiin ilman aktiivihakemistoon liittämistä. Ratkaisu vaatii paljon resursseja ja kyseessä on siis kaupallinen tuote, sillä palvelu on osa Windows Server käyttöjärjestelmää. Etuna vapaan lähdekoodin tuotteisiin on helppo integroiminen aktiivihakemistoon, asennuksen ja käytön helppous sekä Microsoftin tarjoama tekninen tuki ongelmatilanteissa. Tuote on myös selkeästi viimeistellympi verrattuna ilmaisiin vaihtoehtoihin.

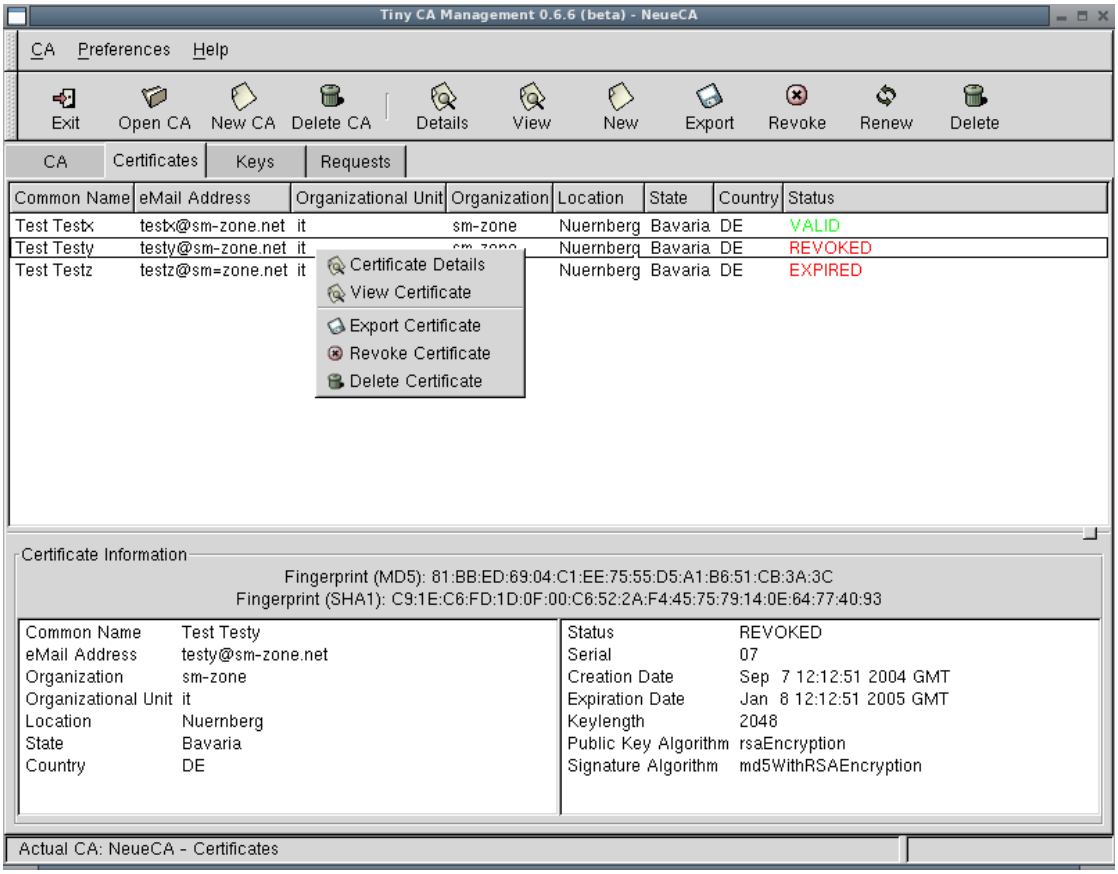
4.2 TinyCA

TinyCA on yksinkertainen, ilmainen ja kevyt pienen sertifiointielimen hallintaan tarkoitettu käyttöliittymä. TinyCA on kirjoitettu Perl-ohjelmointikielellä ja se pohjautuu GTK2 nimiseen graafiseen käyttöliittymäkirjastoon. Se toimii käyttäjärajapintana (frontend) vapaaseen lähdekoodiin perustuvalle OpenSSL-salauskirjastolle (ks. kuvio 12). (TinyCA n.d.)

TinyCA soveltuu hyvin yksinkertaiseen sertifikaattien hallintaan ja se tukee muun muassa seuraavia toimintoja (TinyCA n.d):

- Rajaton määrä sertifiointielimiä
- Tuki luoda ja hallita alivarmentajia
- Tukee yleisimpiä sertifikaattiformaatteja, muun muassa PEM, DER, TXT ja PKCS#12
- Tuki useammalle kielelle.

TinyCA on kuitenkin liian yksinkertainen ja huonosti skaalautuva ratkaisu, sitä ei myöskään enää ole kehitetty eteenpäin moneen vuoteen. Myös muokattavuus ja automatisointi ovat rajoittunutta. (TinyCA n.d.)



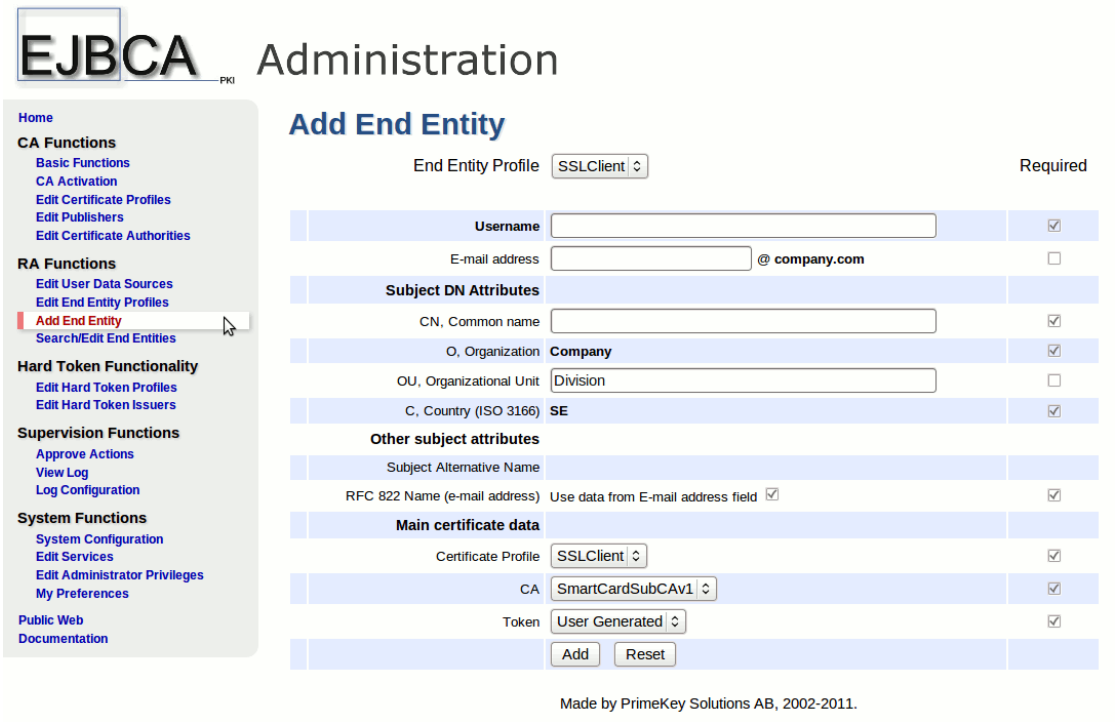
KUVIO 12. TinyCA-käyttöliittymä (TinyCA n.d.)

4.3 EJBCA

Enterprise Java Bean Certificate Authority (EJBCA) on yrityksille suunnattu ilmainen PKI -ratkaisu. EJBCA pohjautuu JEE teknologiaan ja sen kehityksen aloittivat Thomas Gustavsson ja Philip Vendil 90-luvun loppupuolella. Tällä hetkellä projektia ylläpitää ja sponsoroi Ruotsalainen yhtiö nimeltä PrimeKey Solutions AB. Yhtiö omistaa myös tekijänoikeudet suurimpaan osaan ohjelmakoodista. Ensimmäinen julkaisu EJBCA:sta oli vuonna 2001 ja tällä hetkellä uusin versio on 4.0.11. Projektin lähdekoodi on vapaasti saatavilla GNU Lesser General Public Licence -lisenssin alaisuudessa. (Vatra 2011, 534 – 535.)

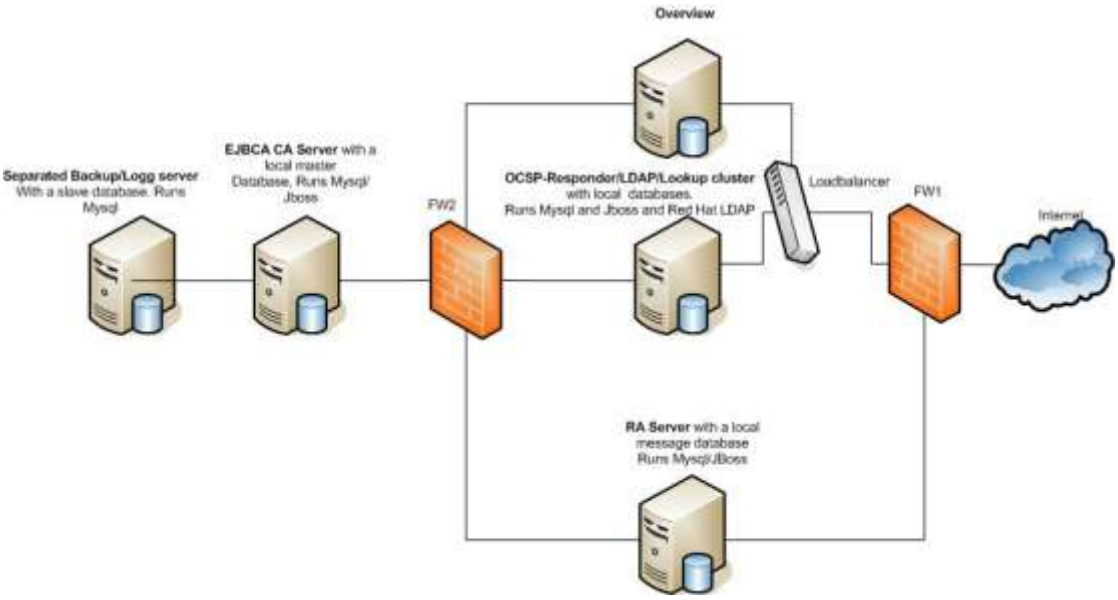
EJBCA on kirjoitettu Javalla ja sen voi asentaa mille tahansa käyttöjärjestelmälle, joka kykenee ajamaan Java sovelluspalvelinta, kuten JBOSS:ia. EJBCA on joustava, vakaa, skaalautuva ja eri komponentteihin perustuva CA, sitä voidaan käyttää joko yksin (standalone) tai integroituna muihin JEE sovelluksiin. Se sisältää runsaasti eri toimintoja ja tukee monia eri standardeja ja teknologioita, muun muassa mobiili-PKI, äly-

kortit, OpenVPN, Cisco ja Juniper. EJBCA:n vahvuutena on myös sen monipuolinen ja selkeä käyttöliittymä (ks. kuvio 13). (EJBCA 2012.)



KUVIO 13. EJBCA-käyttöliittymä (EJBCA 2012)

Kuviossa 14 on esitelty esimerkki EJBCA toteutuksesta, jossa on turvattu CA ulkoisella OCSP- ja RA-palvelimella. RA-palvelin vastaanottaa sertifikaattipyynnöt ja CA kyselee pyyntöjä tietyin väliajoin. CA palauttaa allekirjoitetut sertifikaatit ulkoiselle RA:lle, jolloin liikenne ei missään vaiheessa kulje oikealta vasemmalle FW2-palomuurin läpi. (EJBCA 2012.)



KUVIO 14. EJBCA arkkitehtuuri (EJBCA 2012)

EJBCA on varmasti tunnetuimpia ja käytetyimpiä vapaan lähdekoodin PKI-järjestelmiä OpenCA:n kanssa. EJBCA:n sivuilla onkin listattu useita eri organisaatioita ja julkisen hallinnon elimiä, jotka ovat ottaneet järjestelmän käyttöön. Esimerkiksi Ruotsin poliisi, Ruotsin oikeusministeriö, Ranskan puolustusministeriö ja valtionvarainministeriö, Kreikan poliisi ja LVM AG. Useilla näistä on jopa satoja tuhansia käyttäjiä, sitä käytetään myös laajasti esimerkiksi elektronisten passien myöntämiseen. (EJBCA 2012.)

4.4 OpenCA

OpenCA on vapaan lähdekoodin projekti, joka aloitettiin vuonna 1999. Tarkoituksena oli tarjota vapaaseen lähdekoodiin perustuva halpa, laadukas ja pitkäaikainen PKI -ratkaisu. Ideana oli perustaa OpenCA kolmeen eri pääkomponenttiin: Perl Web-käyttöliittymä, OpenSSL taustalla (backend) vastaamassa kryptografiasta sekä tietokanta.

OpenCA tarjoaa vakaan, muokattavan ja monipuolisen PKI ratkaisun, joka perustuu useaan eri vapaan lähdekoodin ohjelmistoon, kuten OpenLDAP, OpenSSL ja Apache. OpenCA on kirjoitettu Perlillä ja sen turvallisuus perustuu koodin helppolukuisuuteen. Kuka vain pystyy muokkaamaan ja esittämään parannusehdotuksensa vapaan lähdekoodin vuoksi.

Ohjelmisto tarjoaa web-käyttöliittymän Apachen avustuksella, eli selaimella pystytään suorittamaan CA:n vaatimia toimintoja, kuten CA:n konfigurointi, sertifikaattien allekirjoittaminen, sertifikaattipyyntöjen tekeminen, tiedonvaihto sekä sertifikaattien ja sulkulistojen jakaminen. OpenCA tukee myös sertifikaattien tilan tarkistusprotokollaa (OCSP) ja SCEP-protokollaa. (OpenCA Group 2005,)

5 TOTEUTUS

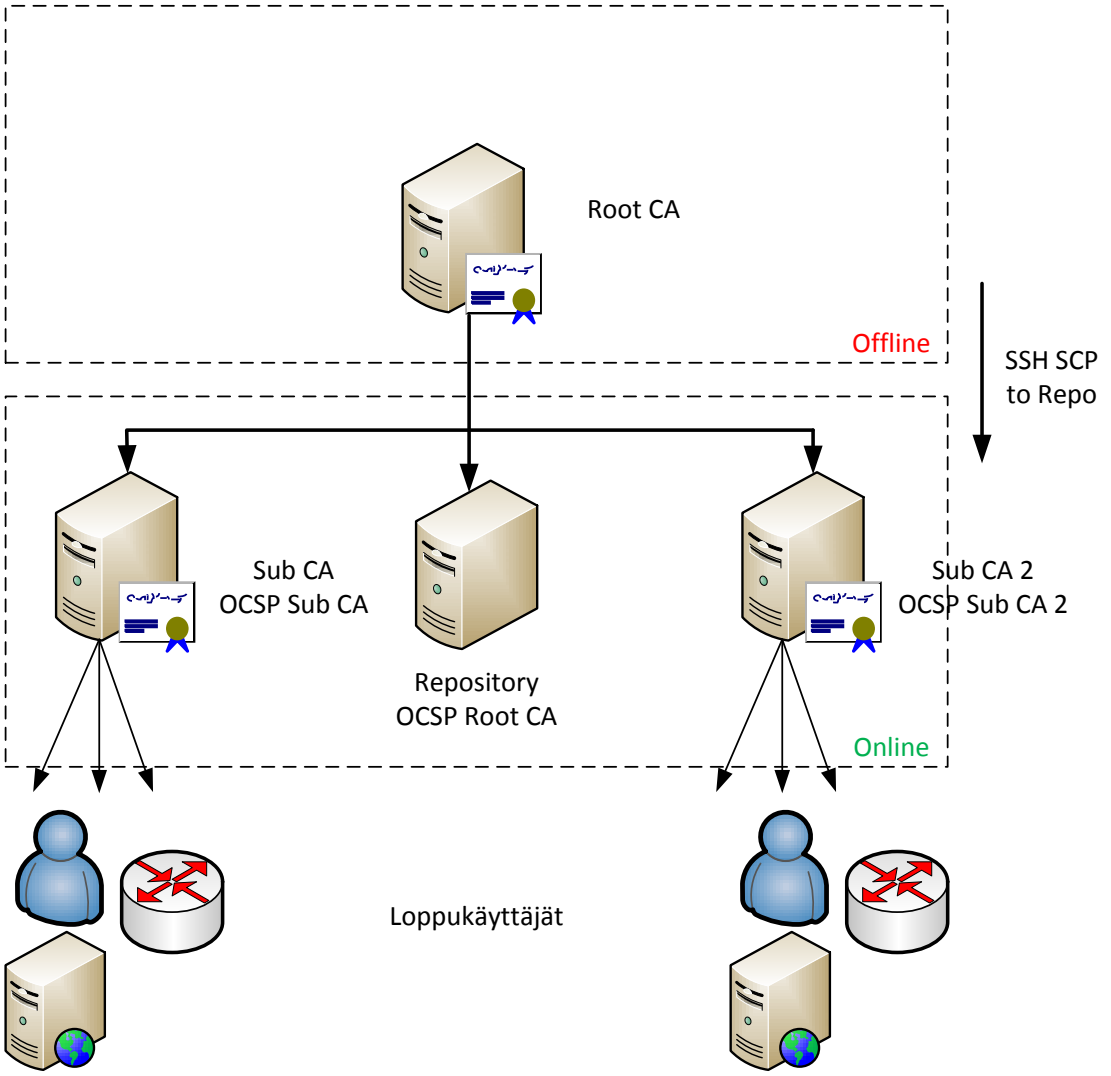
5.1 Topologia, hierarkia ja ohjelmistot/sovellukset

Hierarkiaksi valittiin kaksitasoinen malli (ks. kuvio 15), koska rakenteen hallittavuus on huomattavasti helpompi verrattuna laajempiin kokonaisuuksiin. Yksitasoinen malli on puolestaan liian yksinkertainen ja turvaton, koska juuri joutuu olemaan yhteydellisessä tilassa. Offline juuren ansiosta kaksitasoisen mallin tietoturva paranee, koska juureen ei ole mahdollista kohdistaa verkosta tulevia hyökkäyksiä. Tarvittava tiedonvaihto juuren ja julkisen säilön välillä toteutetaan SSH SCP:n (Secure Copy) avulla. Juuri on hetkellisesti yhteydellisessä tilassa vain kopiointiin vievän ajan. Järjestelmää on helppo laajentaa lisäämällä uusia varmentajia 2-tasolle ja tulevaisuudessa mahdollista myös luoda kolmas taso. Mikäli 2-tasolla oleva varmentaja vaarantuu, voidaan vaarantuneen varmentajan sertifikaatti helposti sulkea juuren toimesta. Toisen alivarmentajan sulkeuduttua, säilyy toinen alivarmentaja edelleen toimintakuntoisena tarjoten sertifikaattipalveluja käyttäjille. Koko rakenteen turvallisuus perustuu juuren turvallisuuden takaamiseen. Topologiassa juuri ei myönnä sertifikaatteja loppukäyttäjille, vaan vain tarvittaessa alivarmentajille.

Jokainen varmentaja sisältää julkisen käyttöliittymän, rekisteröintielimen ja sertifiointielimen. Nämä voitaisiin myös asentaa erikseen, mutta tämä lisää hallinnointiin kuluvaa aikaa. Yleensä RA ja julkinen käyttöliittymä asennetaan samaan ja CA erikseen, mutta tällöin tiedonvaihtoon vaaditaan ylimääräistä työtä ylläpidolta. Vaikka nämä asennetaan ns. samaan nodeen, eri osat on eroteltu, eikä loppukäyttäjällä ole pääsyä RA:n tai CA:n käyttöliittymään.

Topologiassa on myös mukana julkinen Repository. Palvelimelle on koottu keskitetysti varmentajien sertifikaatit, jotka ovat julkisesti saatavilla kaikille.

Jokaiselle varmentajalle on oma OCSP-palvelu, joka vastaa kyseisen varmentajan sertifikaattien tilasta. Käyttäjät voivat tehdä palvelimelle kyselyjä sertifikaattien tilasta, joko manuaalisesti tai automaattisesti, mikäli käytetty sovellus sitä tukee. Palvelu asennetaan molemmille alivarmentajille ja Repository-palvelimelle, joka vastaa juuren sertifikaattien tilasta, koska juuri on yhteydettömässä tilassa. Ratkaisu on vikasietoinen, sillä Repository:n vikaantuessa koko OCSP-palvelu ei lakkaa toimimasta. Palvelun voisi asentaa myös ainoastaan Repository-palvelimelle, mutta OpenCA-projektin tarjoama OCSP Responder konfiguroituna usealle CA:lle aiheuttaa ongelmia ja toisaalta vikasietoituus on huono.



KUVIO 15. JST hierarkia

Taulukkoon 2 on merkattu varmentajille myönnettävien sertifikaattien voimassaoloajat ja avaimen pituudet sekä CA:n tyyppi. 4096 bittinen avain juurella on käytännössä murtamaton nykypäivän tekniikalla ja odotetusti vielä pitkän aikaa. Mielestäni pituus ei ole ylilyönti, vaikka useimmat varmenneorganisaatiot käyttävät vain 2048 bittisiä avaimia juurivarmentajillaan. Tekniikka kehittyy ja laskentateho kasvaa jatkuvasti, joten pitkällä avaimella varmistetaan juuren turvallisuus pitkälle tulevaisuuteen. Avaimen käytön hitaus ei myöskään ole ongelma juuren tapauksessa, koska se myöntää varmenteita ainoastaan alivarmentajille. Juurivarmenteelle voidaan huoletta laittaa 20 vuoden voimassaoloaika. Alivarmentajille puolitetaan avaimen pituus ja voimassaoloaika. 2048 bittinen avain on erittäin vahva ja tuplasti nopeampi kuin 4096 bittinen. Alivarmentajille vaaditaan nopeammat avaimet, koska ne vastaavat sertifikaattien myöntämisestä loppukäyttäjille.

TAULUKKO 2. Varmentajien voimassaolo

CA:n nimi	Voimassaoloaika	Avaimen pituus	Tyyppi
Root CA	20 vuotta	4096 bittiä	Offline
Sub CA	10 vuotta	2048 bittiä	Online
Sub CA 2	10 vuotta	2048 bittiä	Online

Eri toteutusvaihtoehdoista vertailtiin OpenCA, EJBCA, TinyCA ja Windows Server Certificate Services. Jokaista ohjelmistoa testatiin ennakkoon ja arvioitiin sen soveltuvuutta toimeksiantajan tarpeisiin. Saatujen havaintojen ja kerätyn tiedon perusteella valittiin sopivin toteutusvaihtoehto pisteyttämällä ohjelmistot eri kriteerien mukaan (ks. taulukko 3). Eri kriteerien painoarvot määräytyvät toimeksiantajan tarpeiden ja omien päätelmien perusteella:

- Resurssitarve, 0,6
- Modulaarisuus, 0,4
- Skaalautuvuus, 0,4
- Muokattavuus, 0,6
- Toimintojen kattavuus, 0,4

TAULUKKO 3. Ohjelmistojen pisteytys

Sovellus	Pisteet 0-5, suurempi = parempi: Resurssitarve	Modulaarisuus	Skaalautuvuus	Muokattavuus	Toimintojen kattavuus	Pisteet yhteensä
Windows Server Certificate Services	2	2	5	1	5	6,6
TinyCA	5	1	1	1	1	4,8
OpenCA	4	5	3	5	4	10,2
EJBCA	2	4	4	3	5	8,2

Ohjelmistoista karsiutui saman tien Microsoftin tarjoama Windows Serverin mukana tuleva Certificate Services. Windows Server on kaupallinen tuote ja se vaatii paljon resursseja. Toimeksiantajan kahtena kriteerinä oli ilmainen ja kevyt ratkaisu, joten tähän käyttötarkoitukseen tuote ei sovellu.
TinyCA on nimensä mukaisesti liian suppea toimeksiantajan käyttötarkoitukseen ja se on huonosti laajennettavissa. Siitä myös puuttuivat automatisointitoiminnot, muokattavuus ja modulaarinen rakenne. TinyCA sopii lähinnä yksinkertaiseen sertifikaattien hallintaan.
OpenCA ja EJBCA ovat varmasti suosituimmat vapaan lähdekoodin CA -ratkaisut ja niitä kehitetään edelleen, tosin OpenCA:n kehitys on hidastunut viime vuosina. Tämä ilmenee pitkästä julkaisuvälistä versioiden välillä ja versioiden välisistä muutoksista.

EJBCA on täysiverinen yritystason PKI-järjestelmä, joka on todella monipuolinen ja skaalautuva. (Ghori & Parveen 2006, 12) mukaan EJBCA on erittäin monimutkainen ottaa käyttöön ja konfiguroida. EJBCA:n asennusta ja konfigurointia testattiin myös käytännössä ja varmistettiin väitteen paikkaansapitävyys. EJBCA pohjautuu Javaan, joten se toimii millä tahansa Java sovelluspalvelinta pyörittävällä alustalla. EJBCA on kuitenkin ylilyönti toimeksiantajan tarpeisiin ja sen konfigurointi vaatii paljon työtä ja perehtymistä. Lisäksi Java EE on raskas ja monimutkainen sekä altis haavoittuvuuksille. Toimeksiantaja ei myöskään ollut myönteinen Java-pohjaisesta ratkaisusta.
Lopuksi päädyttiin toteuttamaan varmennepalvelut OpenCA:lla. OpenCA tarjoaa parhaan muokattavuuden ja modulaarisuuden. Se on kevyt ja hyvin dokumentoitu ohjelmisto. OpenCA:n kehitys aloitettiin jo vuonna 1999 ja se on ollut aktiivista tähän päivään saakka. Viimeisin versio 1.3.0 on julkaistu vuoden 2012 huhtikuussa. Taustal-

la on aktiivinen yhteisö ja erilaisia ohjeita on saatavilla runsaasti. OpenCA sisältää lähes samat toiminnot kuin EJBCA, mutta jotkin toiminnoista vaativat kustomointia. Perusominaisuuksien käyttöönotto vaatii suhteellisen paljon perehtymistä ja konfigurointia, mutta ei kuitenkaan samassa mittakaavassa kuin EJBCA.

Käyttöjärjestelmänä käytettiin Debian Linux-jakeluun perustuvaa Ubuntu 11.10 Onereic Ocelot Server -versiota. Ubuntu on ennestään tuttu ja se on helppo ottaa käyttöön. Ubuntu on erittäin suosittu ja sillä on aktiivinen suomalainen käyttäjäyhteisö. Ubuntu tukee myös kehittynyttä apt-get ohjelmapakettienhallintaa. Tällä hetkellä uusin versio Ubuntusta on 12.04.1 LTS, mutta tämän version kanssa oli ongelmia OpenCA:n kääntämisessä. Asennuksessa käytettiin Ubuntu Server-versiota minimaalisella kokoonpanolla, johon asennettiin graafinen GNOME-käyttöliittymä ilman lisäosia. Graafista käyttöliittymää tarvitaan paikalliseen hallinnointiin, erityisesti yhteydettömässä tilassa olevalle juurelle.

OpenCA vaatii toimiakseen myös joukon tukisovelluksia:

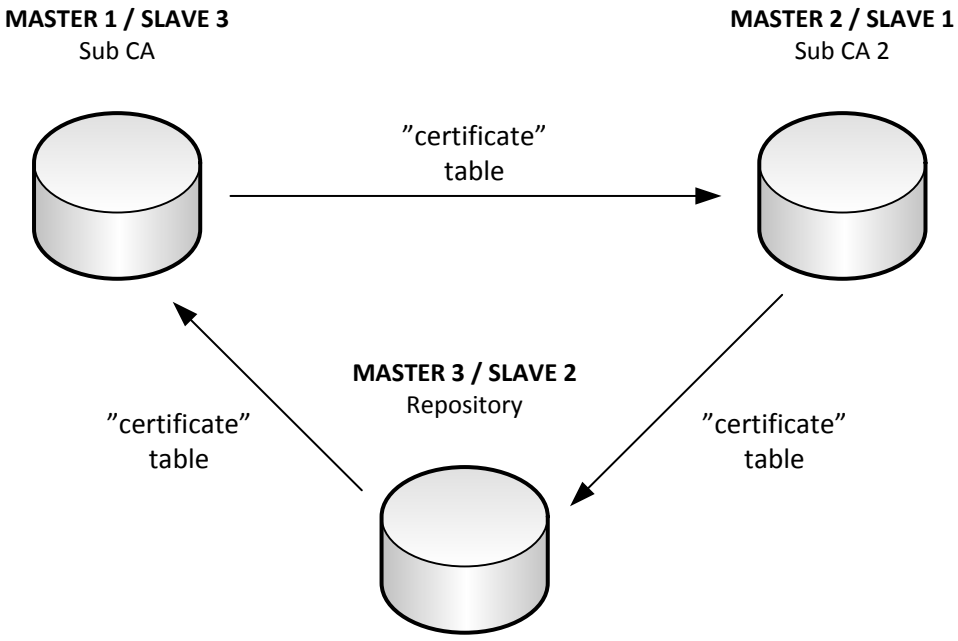
- PERL tuki
- Useita eri PERL moduuleja
- OpenSSL 0.9.7+ salauskirjasto
- Apache web-palvelin
- Tietokanta (MySQL)
- OpenCA-Tools

OpenCA tukee myös muita relaatiotietokantoja, kuten PostgreSQL, DB2 ja Oracle. Valitsin kuitenkin käytettäväksi MySQL:n, koska se on ennestään tuttu, kevyt, nopea ja helppokäyttöinen relaatiotietokanta.

OCSP Responder on OpenCA:n tekijöiltä ja se vaatii toimiakseen libPKI kirjaston, LDAP- ja XML-kehitystyökalut.

Julkinen Repository on toteutettu MySQL-replikoinnilla. Eri palvelimet replikoivat sertifikaattitaulut Master-Slave rinkiperiaatteella, eli jonkun palvelimen sertifikaattitaulussa tapahtuva muutos replikoituu kaikille ringissä oleville. Repository toimii Slave-tilassa Sub CA 2:lle. Repository toimii Master-tilassa Sub CA:lle. Sub CA toimii puo-

lestaan Master-tilassa Sub CA 2:lle, joten rinki sulkeutuu. Kuviosta 17 selviää tarkemmin miten replikointi tapahtuu.



KUVIO 16. MySQL-replikointi

Toistaiseksi MySQL ei tue usean Master-noden replikointia yhdelle Slave-nodelle, joten molempien varmentajien sertifikaattien saamiseksi Slave-nodelle, tarvitsee luoda Master-Slave rinki, joka synkronoi kaikkien nodejen sertifikaattitaulut rinkiperiaatteella. Multi Master & One Slave tekniikkaan on olemassa joitakin epävirallisia ratkaisuja. Slave-nodelle voitaisiin luoda kaksi eri instanssia MySQL:lle, jotka käyttävät samaa tietokantaa, mutta eri Master-nodea replikointiin. Ratkaisu ei ole kuitenkaan tuettu suoraan MySQL:n puolesta ja voi aiheuttaa yhteentörmäyksiä sekä muita ongelmia. Tästä syystä valittiin kuvion 17 mukainen ratkaisu, joka tarjoaa kaikki sertifikaatit saataville myös alivarmentajille. Master 2:n vikaantuessa voidaan Slave 2 määrittää replikoimaan sertifikaatit Master 1:ltä. Vikasietoisuuden ja saatavuuden lisäämiseksi voitaisiin ottaa käyttöön myös toinen Repository, joka toimisi Slave-tilassa Master 1-nodelle.

5.2 Esivalmistelut ja apuohjelmien asennus sekä konfigurointi

Asennetaan Ubuntu 11.10 Server versio minimaalisilla asetuksilla. Luodaan sopiva pääkäyttäjä ja ryhmä. Käyttöjärjestelmään asennetaan graafinen GNOME-käyttöliittymä ilman lisäosia. Virtuaalikoneelle asennetaan VMware-tools, joka mahdollistaa copy & paste -toiminnon.

Lisätään luotu käyttäjä /etc/sudoers tiedostoon:

nano /etc/sudoers

 # User privilege specification
root ALL=(ALL:ALL) ALL
<käyttäjä> ALL=(ALL:ALL) ALL

Asennetaan Ubuntuun uusimmat päivitykset:

sudo apt-get update
sudo apt-get upgrade

Asennetaan seuraavat paketit:

sudo apt-get install gcc apache2 mysql-server perl libssl-dev libexpat-dev libmysql-client-dev libdb4.8++-dev (11.10 versiolla paketin nimi on libdb4.8++)

Valmistellaan MySQL tietokanta OpenCA:ta varten.

mysql -u root -p -h localhost
CREATE DATABASE openca;
use openca;
*GRANT ALL PRIVILEGES ON *.* TO '<käyttäjä>'@'localhost' IDENTIFIED BY '<salasana>';*

Testataan tietokantaan kirjautumista luoduilla tunnuksilla:

mysql -u <käyttäjä> -p -h localhost openca

Komennon jälkeen MySQL kysyy salasanaa ja kirjautumisen pitäisi onnistua aikaisemmassa kohdassa luodulla salasanalla.

Lisäksi asennetaan phpMyAdmin tietokannan hallintaa varten ja pakotetaan se käyttämään HTTPS-protokollaa. Määritetään myös juuren hallintaan pääsy sallituksi ainoastaan localhost osoitteesta:

sudo apt-get install phpmyadmin

sudo cp /etc/phpmyadmin/apache.conf /etc/apache2/conf.d/phpmyadmin.conf
sudo nano /etc/phpmyadmin/config.inc.php

$cfg['ForceSSL'] = TRUE;

sudo nano /etc/apache2/conf.d/phpmyadmin.conf

<Directory /usr/share/phpmyadmin>

    Allow from localhost

    Deny from all

sudo /etc/init.d/mysql restart
sudo /etc/init.d/apache2 restart

Haetaan ja asennetaan uusin versio OpenSSL-salauskirjastosta virallisilta sivuilta, tällä hetkellä uusin versio on 1.0.1c. Paketinhallintajärjestelmä ei sisällä uusinta versiota, joten asennetaan paketti käsin.

wget http://www.openssl.org/source/openssl-1.0.1c.tar.gz
tar xzfv openssl-1.0.1c.tar.gz
cd openssl-1.0.1c.tar.gz
./config
make
sudo make install

Kaikki tiedostot asentuvat /usr/local/ssl tiedostoon. Varmistetaan, että käyttäjät käyttävät uusinta versiota lukiessaan manuaalia. Lisätään tekstirivi ennen ensimmäistä MANPATH_MAP tekstiä:

sudo nano /etc/manpath.config

MANPATH_MAP /usr/local/ssl/bin /usr/local/ssl/man

sudo mandb

Lisätään /usr/local/ssl/bin polku /etc/environment tiedoston PATH-muuttujaan ennen Ubuntun määrittämää /usr/bin polkua OpenSSL:lle:

sudo nano /etc/environment
PATH="/usr/local/sbin:/usr/local/bin:/usr/local/ssl/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games"

Kirjaudutaan ulos ja takaisin sisään, jonka jälkeen tarkastetaan versio:

openssl version
OpenSSL 1.0.1c 10 May 2012

OpenCA vaatii toimiakseen useita eri Perl moduuleja. Nämä kannattaa asentaa CPAN-työkalulla, joka tulee Perlin mukana. CPAN (Comprehensive Perl Archive Network) on kokoelma Perlillä kirjoitettuja ohjelmia, moduuleja ja dokumentaatioita. Tarvittaessa moduulit on myös mahdollista ladata CPAN:in sivuilta ja asentaa manuaalisesti lähdekoodista. Suoritetaan CPAN-työkalu ja asennetaan tarvittavat moduulit. CPAN kannattaa päivittää ennen asennusta. CPAN tulee konfiguroida kun se ajetaan ensimmäisen kerran. Vastataan kysymyksiin default arvoilla.

sudo perl -MCPAN -e shell

install CPAN
reload CPAN
install CGI::Session
install Convert::ASN1
install Digest::MD5
install Digest::SHA1
install Encode::Unicode
install IO::Socket::SSL
install IO::Stringy
install MIME::Base64
install MIME::Lite
install MIME::Tools
install MailTool
install Net::Server
install URI
install XML::Twig
install XML::SAX::Base
install Digest::HMAC
install Authen::SASL
install Net::SSLeay
install G/GU/GUIDO/libintl-perl-1.20.tar.gz
install G/GB/GBARR/perl-ldap-0.4001.tar.gz
install DBI
install CGI
install XML::Parser

install Parse::RecDescent
install X500::DN

Ladataan OpenCA tools projektin virallisilta sivuilta, tällä hetkellä uusin versio on 1.3.0. Puretaan ja asennetaan paketti suoraan lähdekoodista. Asennusvaiheessa on hyvä määrittää asennuspolku ja käyttäjä sekä ryhmä:

wget http://ftp.openca.org/openca/openca-tools/releases/v1.3.0/sources/openca-tools-1.3.0.tar.gz
tar xzfv openca-tools-1.3.0.tar.gz
cd openca-tools-1.3.0/
./configure --prefix=/opt/openca --exec-prefix=/opt/openca --with-openca-prefix=/opt/openca --with-openca-user=<käyttäjä> --with-openca-group=<ryhmä>
make
sudo make install

Muokataan Apachen sites-enabled tiedostoon valmiiksi oikea polku CGI-skripteille:

sudo nano /etc/apache2/sites-enabled/000-default

```
#ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
#<Directory "/usr/lib/cgi-bin">
#       AllowOverride None
#       Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
#       Order allow,deny
#       Allow from all
#</Directory>

<Directory /var/www/cgi-bin/>
        AllowOverride None
        Options ExecCGI -MultiViews +SymLinksIfOwnerMatch
        SetHandler cgi-script
        Order allow,deny
        Allow from all
</Directory>
```

Luodaan alias konfiguraatio valmiiksi Apachelle, jotta eri käyttöliittymien polut on lyhyempi kirjoittaa:

sudo nano /etc/apache2/conf.d/openca.conf

```
ScriptAlias /cgi-bin/pki/batch /var/www/cgi-bin/pki/batch
ScriptAlias /cgi-bin/pki/ca /var/www/cgi-bin/pki/ca
ScriptAlias /cgi-bin/pki/ra /var/www/cgi-bin/pki/ra
ScriptAlias /cgi-bin/pki/node /var/www/cgi-bin/pki/node
```

ScriptAlias /cgi-bin/pki/pub /var/www/cgi-bin/pki/pub
ScriptAlias /cgi-bin/pki/ldap /var/www/cgi-bin/pki/ldap

Alias /pki/batch /var/www/html/pki/batch
Alias /pki/ca /var/www/html/pki/ca
Alias /pki/ra /var/www/html/pki/ra
Alias /pki/node /var/www/html/pki/node
Alias /pki/pub /var/www/html/pki/pub
Alias /pki/ldap /var/www/html/pki/ldap

Käynnistetään vielä Apache uudelleen:

sudo /etc/init.d/apache2 restart

Luodaan OpenCA:lle käynnistysskripti (ks. liite 5):

sudo nano /etc/init.d/openca

Lopuksi annetaan skriptille vielä suoritusoikeudet:

sudo chmod o+x /etc/init.d/openca

Haetaan OpenCA:n uusin versio OpenCA-projektin virallisilta sivuilta. Uusin versio on tällä hetkellä 1.3.0. Puretaan paketti jo valmiiksi.

wget http://ftp.openca.org/openca/openca-base/releases/v1.3.0/sources/openca-base-1.3.0.tar.gz
tar xzfv openca-base-1.3.0.tar.gz

DBI-moduulissa on bugi, joka kaataa tietokannan kun tarkastellaan CA:n sertifikaattia. CA sertifikaatin avaintunniste lisätään tietokantaan heksana, mutta DBI-moduulista puuttuu tarkistus heksa-arvon varalta. Muut sertifikaatit lisätään desimaalina. Korjauksena lisätään kelpoisuustarkistus (sanity check) kyseiseen moduuliin. Ajetaan liitteenä 1 oleva ”.patch”-tiedosto:

cd /home/openca/openca-base.1.3.0/src/modules/openca-dbi/
patch DBI.pm opencadb.patch

Lisäksi muokataan sertifikaattipyyntöihin liittyviä komentoja. Komennoissa on sähköpostin tarkistukseen liittyvä mekanismi, joka lisää pyynnöt tietokantaan TEMP-NEW-tilaan, jolloin pyyntöjä ei näytetä käyttöliittymässä. Tilan kuuluu olla NEW, jotta pyynnöt näkyvät oikein käyttöliittymässä. Sähköpostin tarkistus ei ole olennainen toimeksiantajan suljetussa ympäristössä ja se tuo lisää hallinnollisia toimenpiteitä. Toiminto voidaan tarvittaessa ottaa helposti takaisin käyttöön.

cd /home/openca/openca-base.1.3.0/src/common/lib/cmds

Kansiosta löytyy neljä tiedostoa: authenticated_csr, advanced_csr, pkcs10_req ja basic_csr, joista tulee poistaa TEMP -sana seuraavasta kohdasta:

```
if( $verifyEmailAddress =~ /Y/i ) {
$status = "$TEMPstatus";
}
$new_req->setStatus( "$status" );
```

Lopputuloksena syntyy valmis pohja, jota voidaan käyttää varmentajien asennuksessa. Virtuaalikonetta on helppo kloonata ja asentaa haluttu CA valmiista pohjasta.

5.3 Juurivarmentaja

5.3.1 Yleistä

Suurin ero juurivarmentajalla ja muilla järjestelmän osilla on se, että juurivarmentaja toimii yhteydettömässä tilassa, joten sitä voidaan hallita vain paikallisesti. Verkkorajapintojen konfiguraatiosta on syytä kommentoida rajapintojen tuominen ylös järjestelmän käynnistyessä.

5.3.2 Hostname ja hosts-tiedosto

Määritetään juurelle hostname ja muokataan ”hosts”-tiedosto sopivaksi. Hostnamen vaihdon jälkeen kone täytyy käynnistää uudelleen.

sudo nano /etc/hostname

<hostname>

sudo nano /etc/hosts

127.0.0.1 localhost.localdomain localhost
127.0.1.1 <hostname>

sudo reboot

5.3.3 OpenCA asennus ja konfigurointi

Asennusvaiheessa voidaan monipuolisesti määrittää eri asetuksia OpenCA:lle. Eri asetusvaihtoehdoista saa lisätietoa komennolla:

cd openca-base-1.3.0
./configure --help

Osan asetuksista voi myös muuttaa ja määrittää jälkikäteen OpenCA:n konfiguraatiotiedostoista. Osan asetuksista asennus tunnistaa yleensä automaattisesti, esimerkiksi httpd käyttäjä ja ryhmä. Alla on lisätietoa tärkeimmistä OpenCA:n tarjoamista esiasetuksista:

--prefix=/opt/openca --exec-prefix=/opt/openca --with-openca-prefix=/opt/openca
OpenCA:n asennuskansio.

--with-openca-user=<käyttäjä> --with-openca-group=<ryhmä>
OpenCA:n käyttäjä ja ryhmä.

--with-module-prefix=/opt/openca/modules
Moduulien asennuskansio.

--with-openca-tools-prefix=/opt/openca
OpenCA-Tools sijainti.

--with-db-name=openca --with-db-host=localhost --with-db-user=<käyttäjä> --with-db-passwd=<salasana> --with-db-type=mysql
Tietokantaan liittyvät asetukset. Tyypiksi voidaan myös määrittää Pg (PostgreSQL), DB2 tai Oracle.

--enable-scep
SCEP-protokollan käyttöönotto.

--with-httpd-user=www-data --with-httpd-group=www-data

Web-palvelimen käyttäjä ja ryhmä. Apachella usein www-data.

--with-ocsp-server=<URL>

OCSP Responderin osoite.

--with-openssl-prefix=/usr/

OpenSSL:n sijainti.

--with-service-mail-account=<email> --with-support-mail-address= <email> --with-cert-chars=UTF8 --with-ca-organization=<organisaatio> --with-ca-locality=<kaupunki> --with-ca-state=<maakunta> --with-ca-country=<maa, esim. FI>

CA:n tiedot.

Konfiguroidaan asennus seuraavilla optioilla:

./configure --prefix=/opt/openca --exec-prefix=/opt/openca --with-openca-prefix=/opt/openca --with-openca-user=<käyttäjä> --with-openca-group=<ryhmä> --with-module-prefix=/opt/openca/modules --with-openca-tools-prefix=/opt/openca --with-db-name=openca --with-db-host=localhost --with-db-user=<käyttäjä> --with-db-passwd=<salasana> --with-db-type=mysql --with-ocsp-server=<URL>

Onnistuneen konfiguroinnin jälkeen kannattaa tarkastaa asetukset, jonka jälkeen luodaan makefile ja asennetaan paketti:

make
sudo make install-offline install-online

Muokataan OpenCA:n konfiguraatiotiedostoa config.xml.

nano /opt/openca/etc/openca/config.xml

Käydään läpi konfiguraatio ja tehdään tarvittaessa seuraavat muutokset:

organization = <organisaatio>
ca_organization = <CA organisaatio>
ca_country = <maa, esim. FI>
support_mail_address = <email>
service_mail_account = <email>
policy_link = <URL>

CRLDistributionPoints = URI.1=<URL>
NS_CRLDistributionPoint = <URL>
authInfoAccess = caIssuers;URI: <URL>, OCSP;URI: <URL>
dataexchange configuration = CA Only
dataexchange_device_local = <polku tar-tiedostolle>

Otetaan SSL pois käytöstä RA:n käyttöliittymässä (ks. kohta 5.6.1), koska Apachelle ei ole vielä luotu sertifikaattia.

Konfiguroinnin jälkeen asetukset pitää vielä päivittää muihin tiedostoihin. Tätä varten OpenCA:n kansiosta löytyy skripti nimeltä configure_etc.sh:

sudo ./configure_etc.sh

5.3.4 OpenCA käyttöönotto

Määritetään OpenCA käynnistymään bootin yhteydessä:

update-rc.d openca defaults

Käynnistetään OpenCA ensimmäisen kerran komennolla:

sudo /opt/openca/etc/init.d/openca start

Jostain syystä käynnistysskripti antaa ”-path:: not found”-virheen, vaikka se ajetaan openca käyttäjänä pääkäyttäjän komennolla sudo. Tämä ei kuitenkaan vaikuta ohjelman toimintaan tai käynnistykseen, vaan ainoastaan skriptin kysyessä web-käyttöliitymän salasanaa, näytetään se ruudulla selväkielisenä. Root käyttäjällä skriptin suoritus toimii oikein. Salasanalla ei kuitenkaan ole suurta merkitystä, koska myöhemmin konfiguroidaan X.509-pohjainen kirjautuminen. Jatkossa OpenCA:n voi käynnistää aiemmin luodulla skriptillä:

sudo /etc/init.d/openca start

Kirjaudutaan sisään CA:n web-käyttöliittymään osoitteessa http://<hostname>/pki/ca, oletuskäyttäjänimi on admin ja salasana määritettiin aiemmin käynnistysvaiheessa.

Kohdassa ”PKI Init & Config -> Initialization -> DB, Key and Cert Init” löytyy tarvittavat toiminnot kun otetaan CA käyttöön ensimmäistä kertaa.

”Initialize Database” luo tietokantaan tarvittavat taulut, tämä tarvitsee tehdä vain kerran. Mikäli tietokanta luodaan myöhemmin uudelleen, tyhjentyy sen sisältö.

Luodaan uusi salainen avain CA:lle kohdassa ”Generate new CA secret key”. Symmetriseksi salausalgoritmiksi valitaan aes256 ja asymmetriseksi rsa sekä avaimen pituus 4096 bittiä. Seuraavaksi syötetään vahva salasana, jolla salainen avain salataan.

Luodaan uusi sertifikaattipyyntö kohdassa ”Generate new CA Certificate Request (use generated secret key)”, jossa käytetään edellisessä kohdassa luotua salaista avainta. Täytetään kohdat seuraavasti:

Common Name (e.g., CertAuth 1) = <CA:n nimi>
Organizational Unit Name (e.g. MyUnit) = <organisaatioyksikkö>
Organization (e.g. OpenCA Labs) = <organisaatio>
ISO 3166 Country Code (e.g. IT, DE, US, ...) = <maa, esim. FI>

Luodaan itseallekirjoitettu sertifikaatti kohdassa “Self Signed CA Certificate (from already generated request)”. Täytetään kohdat seuraavasti:

Serial Number (eg., 00, a0d399) = 00
Certificate Validity (Days) = 7300
Subject Alt Name = email, “ “
Extensions = Self Signed CA

Luodaan luottoketju kohdasta “Rebuild CA Chain”. Luottoketjua tarvitaan sertifikaattien allekirjoitusten tarkistamisessa.

CA:n sertifikaatti löytyy kohdasta ”Information -> CA Certificates -> Valid”. Painamalla sertifikaatin sarjanumeroa avautuu lisätietoja. Asetetaan CA luotetuksi painamalla sertifikaatin kuvaa, jonka jälkeen sitä voidaan tarkastella selaimen säilöstä kohdasta ”Authorities”.

Lopuksi luodaan CA operaattorille (CA Operator) sertifikaatti (ks. kohta 5.6.2).

5.4 Alivarmentajat

5.4.1 Yleistä

Molempien alivarmentajien asennus on lähes identtinen, joten molempiin pätee sama asennusohje, lukuun ottamatta osoitteita ja nimeämisiä. Alla on esitetty toisen alivarmentajan asennus ja konfigurointi.

5.4.2 Hostname ja hosts-tiedosto

Määritetään alivarmentajalle sopiva hostname ja muokataan ”hosts”-tiedosto sopivaksi. Hostnamen vaihdon jälkeen kone täytyy käynnistää uudelleen.

sudo nano /etc/hostname

<hostname>

sudo nano /etc/hosts

127.0.0.1 localhost.localdomain localhost
127.0.1.1 <hostname>

sudo reboot

5.4.3 OpenCA asennus ja konfigurointi

OpenCA:n asennus ja konfigurointi on vastaavanlainen kuin juurivarmentajan kohdalla (ks. kohta 5.3.2 ja 5.3.3), lukuun ottamatta pieniä muutoksia muun muassa konfigurointiparametriin:

./configure --prefix=/opt/openca --exec-prefix=/opt/openca --with-openca-prefix=/opt/openca --with-openca-user=<käyttäjä> --with-openca-group=<ryhmä> --with-module-prefix=/opt/openca/modules --with-openca-tools-prefix=/opt/openca --with-db-name=openca --with-db-host=localhost --with-db-user=<käyttäjä> --with-db-passwd=<salasana> --with-db-type=mysql --with-ocsp-server=<URL> --enable-scep
make
sudo make install-offline install-online

Tehdään alla olevat muutokset pääkonfiguraatiotiedostoon:

nano /opt/openca/etc/openca/config.xml

organization = <organisaatio>
ca_organization = <CA organisaatio>
ca_country = <maa, esim. FI>
support_mail_address = <email>
service_mail_account = <email>
authInfoAccess = caIssuers;URI:<URL>,OCSP;URI:<URL>
dataexchange configuration = CA Only
dataexchange_device_local = <polku tar-tiedostolle>

Otetaan SSL pois käytöstä RA:n käyttöliittymässä (ks. kohta 5.6.1), koska Apachelle ei ole vielä luotu sertifikaattia.

Määritetään OpenCA käynnistymään bootin yhteydessä:

update-rc.d openca defaults

Otetaan asetukset käyttöön ja käynnistetään OpenCA ensimmäisen kerran:

sudo bash /opt/openca/etc/openca/configure_etc.sh
sudo /opt/openca/etc/init.d/openca start

5.4.4 OpenCA käyttöönotto

Alivarmentajan sertifikaatin allekirjoittaminen juurella vaatii ylläpitäjältä useita manuaalisia vaiheita.

Kirjaudutaan sisään CA:n web-käyttöliittymään osoitteessa http://<hostname>/pki/ca, oletuskäyttäjänimi on admin ja salasana määritettiin aiemmin käynnistysvaiheessa.

Kohdassa ”PKI Init & Config -> Initialization -> DB, Key and Cert Init” löytyy tarvittavat toiminnot kun otetaan CA käyttöön ensimmäistä kertaa.

Luodaan tietokantaan tarvittavat taulut ”initialize Database”-kohdasta.

Luodaan uusi salainen avain CA:lle kohdassa ”Generate new CA secret key”. Symmetriseksi salausalgoritmiksi valitaan aes256 ja asymmetriseksi rsa sekä avaimen pituus 2048 bittiä. Seuraavaksi syötetään vahva salasana, jolla salainen avain salataan.

Luodaan uusi sertifikaattipyyntö kohdassa ”Generate new CA Certificate Request (use generated secret key)”, jossa käytetään edellisessä kohdassa luotua salaista avainta. Täytetään kohdat seuraavasti:

Common Name (e.g., CertAuth 1) = <CA:n nimi>
Organizational Unit Name (e.g. MyUnit) = <organisaatioyksikkö>
Organization (e.g. OpenCA Labs) = <organisaatio>
ISO 3166 Country Code (e.g. IT, DE, US, ...) = <maa, esim. FI>

Viedään pyyntö pakattuun tiedostoon kohdasta “Export CA Certificate Request”.

Puretaan paikalliseen tiedonvaihtoon tarkoitetun tarrin sisältö ja siirretään careq.pem käyttäjän kotikansioon:

tar xvf <polku>
mv careq.pem /home/<käyttäjä>/

Purettu tiedosto pitäisi olla nimeltään careq.pem, joka on siis alivarmentajan luoma sertifikaattipyyntö. Tehdään uusi palvelimelle tarkoitettu sertifikaattipyyntö juuren julkisen käyttöliittymän kautta osoitteessa http://<hostname>/pki/pub. Palvelinpyyntö löytyy kohdasta “My Certificates -> Request a Certificate -> Server Certificate Request”. Täytetään tiedot seuraavasti:

Request [PEM formatted file] = /home/<käyttäjä>/careq.pem
Registration Authority = Trustcenter itself
Role = Sub-CA
Level Of Assurance = High
DNS (Subject Alt Name) = <Sub CA FQDN>

Kirjaudutaan sisään juuren CA-käyttöliittymään ja etsitään uudet sertifikaattipyynnöt kohdasta ”CA Operations -> Certification Requests -> New”. Painetaan pyynnön sarjanumerosta ja muokataan pyyntöä kohdasta ”Edit Request”. Määritetään varmenteen LOA:ksi (Level of Assurance) High ja voimassaoloajaksi 3650 päivää, eli 10 vuotta.

Myönnetään sertifikaatti kohdasta ”Issue certificate”.

Myönnetty sertifikaatti löytyy juuren kansiosta /opt/openca/var/openca/crypto/certs, josta se täytyy kopioida määritettyyn alivar-

mentajan paikallisen tiedonvaihdon kansioon. Polku määritettiin aikaisemmin CA:n konfiguraatiotiedostoon. Sertifikaatti täytyy myös nimetä ”cacert.pem”-tiedostoksi ja pakata tar:lla oikean nimiseksi. ”Cacert.pem”-tiedoston ja tar-paketin omistaja on syytä tarkistaa ennen sertifikaatin tuontia alivarmentajalle, omistajana tulee olla ”www-data”, eli Apachen käyttäjä.

Kopioidaan myös juuren sertifikaatti kansiosta /opt/openca/var/openca/crypto/cacerts/cacert.crt alivarmentajalle kansioon /opt/openca/var/openca/crypto/chain/rootcert.crt ja varmistetaan, että omistajana säilyy ”www-data”.

Kirjaututaan alivarmentajan CA-käyttöliittymään kohtaan ”PKI Init & Config -> Initialization -> DB, Key and Cert Init” ja painetaan ”Import CA certificate (approved by Root CA)”. Komennon jälkeen OpenCA ilmoittaa onnistuiko tuonti.

Painetaan vielä “Rebuild CA Chain”, joka luo luottoketjun uudelleen kopioidun juurisertifikaatin perusteella.

5.4.5 Muut

Asennetaan OCSP Responder (ks. kohta 5.6), joka vastaa ainoastaan kyseisen alivarmentajan tilasta.

Kytketään automaattinen sertifikaattien myöntäminen ja sulkeminen päälle CA:n käyttöliittymästä. Lisäksi määritetään uuden sulkulistan julkaisu automaattisesti kerran päivässä. Toiminnot löytyvät ”CA Operations”-kohdasta:

- Auto Certificate Issuing
- Auto Certificate Revocation
- Auto CRL Issuing

Otetaan ruksi pois kohdasta “Processed (Signed) Requests Only”. Myönnetään ja suljetaan automaattisesti kaikki sertifikaatit, joiden pyyntö on hyväksytty ilman allekirjoitusta RA:n toimesta, lukuun ottamatta CA-sertifikaatteja, joiden hyväksytyt pyynnöt täytyy käsitellä manuaalisesti CA-käyttöliittymässä. Muuten käytetään vakioasetuksia. OpenCA ei toistaiseksi tue toimintojen automaattista käynnistystä palvelun

uudelleenkäynnistyksen yhteydessä, joten ne pitää ottaa käyttöön joka kerta kun OpenCA käynnistetään uudelleen.

5.5 Repository

5.5.1 Yleistä

Sertifikaattien tilasta kertova palvelu juurelle ja sertifikaattisäilö asennetaan samalle palvelimelle. Sertifikaattisäilönä toimii OpenCA:n julkinen käyttöliittymä, johon synkronoidaan uusimmat sertifikaatit MySQL-replikoinnin avulla.

5.5.2 Hostname ja hosts-tiedosto

Määritetään säilölle sopiva hostname ja muokataan ”hosts”-tiedosto sopivaksi. Hostnamen vaihdon jälkeen kone täytyy käynnistää uudelleen.

sudo nano /etc/hostname

<hostname>

sudo nano /etc/hosts

127.0.0.1 localhost.localdomain localhost
127.0.1.1 <hostname>

sudo reboot

5.5.3 OpenCA asennus ja konfigurointi

OpenCA:n asennus ja konfigurointi eroaa hieman juurivarmentajan ja alivarmentajien asennuksesta, koska Repository tarvitsee vain julkisen käyttöliittymän.

Konfiguroidaan OpenCA:

./configure --prefix=/opt/openca --exec-prefix=/opt/openca --with-openca-prefix=/opt/openca --with-openca-user=<käyttäjä> --with-openca-group=<ryhmä> --with-module-prefix=/opt/openca/modules --with-openca-tools-prefix=/opt/openca --with-db-name=openca --with-db-host=localhost --with-db-user=<käyttäjä> --with-db-passwd=<salasana> --with-db-type=mysql
make
sudo make install-common install-modules install-node install-public

Tehdään seuraavat muutokset pääkonfiguraatiotiedostoon:

organization = <organisaatio>
ca_organization = <CA organisaatio>
ca_country = <maa, esim. FI>
support_mail_address = <email>
service_mail_account = <email>
dataexchange configuration = Public/SCEP Only (muokattu, ks. liite 7)
dataexchange_device_local = <polku tar-paketille>

Repository:n julkisesta käyttöliittymästä on karsittu ”My Certificates”-osiosta kohtia, koska kyseisen käyttöliittymän kautta ei voida hakea tai sulkea sertifikaatteja. Myöskään sertifikaattipyyntöjen tilaa ei voida tarkastella ”Information”-kohdasta. Käyttöliittymästä voidaan kuitenkin salaisen avaimen perusteella asentaa sertifikaatti käyttäjän säilöön, vaikka se on haettu alivarmentajilta. Käyttöliittymää voidaan muokata ”menus”-kansiosta, jossa on myös alkuperäinen tiedosto tallella. Muokattu ”pubmenu.xml.template”- tiedosto löytyy kokonaisuudessaan liitteestä 5.

Määritetään OpenCA käynnistymään bootin yhteydessä:

update-rc.d openca defaults

Käynnistetään OpenCA ensimmäisen kerran komennolla:

sudo /opt/openca/etc/init.d/openca start

5.5.4 Muut

Asennetaan OCSP Responder (ks. kohta 5.6), joka vastaa ainoastaan juuren sertifikaattien tilasta.

5.6 OCSP Responderin asennus ja konfigurointi

Palvelu asennetaan molemmille alivarmentajille, jotka vastaavat oman CA:n sertifikaattien tilasta. Lisäksi Repository-palvelimelle tulee oma OCSP, joka vastaa yhteydettömässä tilassa olevan juuren sertifikaattien tilasta. Palvelun asennus on sa-

manlainen jokaiselle osapuolelle, lukuun ottamatta nimeämisiä ja sertifikaattien luontia. Alla on esitetty vain juuren palvelun asennus julkiselle Repository-palvelimelle.

OCSP Responder vaatii toimiakseen seuraavat paketit:

sudo apt-get install libldap2-dev libxml2-dev

Haetaan libPKI ja OCSP Responder OpenCA-projektin virallisilta sivuilta. Molemmat puretaan ja asennetaan vakioasetuksilla.

wget http://ftp.openca.org/libpki/releases/v0.6.7/sources/libpki-0.6.7.tar.gz
wget http://ftp.openca.org/openca-ocspd/releases/v2.1.1/sources/openca-ocspd-2.1.1.tar.gz
tar xzfv libpki-0.6.7.tar.gz
tar xzfv openca-ocspd-2.1.1.tar.gz
./configure
make
sudo make install

Asennuksen jälkeen luodaan uusi sertifikaattipyyntö kyseisen CA:n OCSP Responderille. Tätä varten OpenCA-tiimi on tehnyt erillisen skriptin, joka löytyy /usr/local/bin kansiosta.

sudo bash /usr/local/bin/ocspd-genreq.sh
CN=<nimi>, OU=<organisaatioyksikkö>, O=<organisaatio>, C=<maa, esim. FI>
Algorithm = RSA-SHA256
Key Size = 1024

Suorituksen jälkeen skripti antaa ohjeet jatkotoimenpiteisiin:

Luotu pyyntö löytyy polusta /usr/local/etc/ocspd/req.pem. Pyyntö täytyy kopioida openca käyttäjän kotihakemistoon ja vaihtaa omistajaksi openca, jotta pyyntö voidaan lähettää selaimella CA:lle.

Luodaan uusi palvelimille tarkoitettu sertifikaattipyyntö CA:n julkisessa käyttöliittymässä ja allekirjoitetaan sertifikaatti CA:n toimesta. Pyyntöä tehdessä on tärkeää valita rooliksi ”OCSP Server”.

Myönnetty sertifikaatti tulee sijoittaa polkuun /usr/local/etc/ocspd/certs/cert.pem ja lisäksi myöntäjän sertifikaatti polkuun /usr/local/etc/ocspd/certs/cacert.pem.

Seuraavaksi konfiguroidaan CA tiedostoon /usr/local/etc/ocspd/ca.d/self-certs.xml, johon määritetään nimi, jota käytetään lokitiedostoissa ja polku CA:n sertifikaatille sekä sulkulistalle:

<pki:name><nimi></pki:name>
<pki:caCertUrl>/opt/openca/var/openca/crypto/cacerts/cacert.pem</pki:caCertUrl>
<pki:crlUrl>/opt/openca/var/openca/crypto/crls/cacrl.pem</pki:crlUrl>

Lopuksi käynnistetään palvelu komennolla:

/usr/local/etc/init.d/ocspd start

Jostain syystä OCSP Responder ei osaa automaattisesti päivittää sulkulistaa laskureiden päätyttyä, mutta manuaalinen päivitys onnistuu normaalisti komennolla:

/usr/local/etc/init.d/ocspd reload-crl

Lisätään komento crontabin suoritettavaksi joka minuutti, jotta saadaan ajantasainen tieto sulkulistan muutoksista. Lista haetaan paikallisesta kansiosta, joten resurssien tarve päivitykselle on minimaalinen.

crontab –e
**/1 * * * * /usr/local/etc/init.d/ocspd reload-crl >/dev/null 2>&1*

OCSP:n toiminta voidaan helposti todeta OpenSSL:n tarjoamalla komennolla:

openssl ocsp -issuer cacert.pem -cert apache.pem -url <URL>-CAfile cafile.pem

jossa

- issuer = myöntäjän sertifikaatti
- cert = sertifikaatti, joka halutaan tarkistaa
- url = OCSP Responderin osoite
- CAfile = CA ketju, esimerkiksi jos sertifikaatin on myöntänyt juuri, osoitetaan CAfile juurisertifikaattiin. Jos sertifikaatin on myöntänyt alivarmentaja, tulee

tiedoston sisältää alivarmentajan ja juuren sertifikaatit PEM muodossa allekkain.

Onnistuneen pyynnön jälkeen Responder ilmoittaa sertifikaatin tilan:

Response verify OK
apache.pem: revoked

This Update: Sep 18 06:28:12 2012 GMT
Next Update: Sep 18 06:33:12 2012 GMT
Revocation Time: Sep 17 18:40:09 2012 GMT

Tilan kysely onnistuu myös ilman CA ketjua, mutta tässä tapauksessa saatuja vastauksia ei pystytä todentamaan.

5.7 MySQL replikointi

5.7.1 Master

Toteutuksessa konfiguroidaan yhteensä kolme Master-nodea. Alla on esimerkki Sub CA Master-noden konfiguroinnista.

Tehdään seuraavat muutokset/lisäykset MySQL:n konfiguraatiotiedostoon ”my.cnf”:

[mysqld]

 # bind-address = 127.0.0.1
server-id = 1
log_bin = /var/log/mysql/mysql-bin.log
binlog_do_db = openca
log-slave-updates
replicate-same-server-id = 0
replicate-ignore-table=openca.ca_certificate
replicate-ignore-table=openca.crl
replicate-ignore-table=openca.crr
replicate-ignore-table=openca.messages
replicate-ignore-table=openca.request
replicate-ignore-table=openca.user
replicate-ignore-table=openca.user_data

Bind-address kommentoidaan, jotta tietokantaan saadaan yhteys ulkopuolelta. Server-id tulee olla eri jokaisella rinkiin kuuluvalla.

Log-slave-updates vaaditaan, että Master-nodelta vastaanotettu tieto välittyy vastaanottajan Slave-nodelle. Replicate-same-server-id välttää silmukoiden syntymistä. Ignore-asetuksilla määritetään osapuolet replikoimaan ainoastaan ”certificate”-taulu.

Luodaan jokaiselle ringissä olevalle osapuolelle käyttäjä replikointioikeuksilla. Tunnusta tarvitaan orjan konfiguroinnissa.

mysql -u root -h localhost –p
*GRANT REPLICATION SLAVE ON *.* TO '<käyttäjä>'@'%' IDENTIFIED BY '<salasana>';*
FLUSH PRIVILEGES;
show master status;

”Show master status;”-komennon tuloste on kuvion 18 mukainen. Tulosteen antamia tietoja tarvitaan orjan konfiguroinnissa ja tiedot tulee poimia jokaiselta Master-nodelta erikseen ja syöttää ne tämän orjan konfiguroinnin yhteydessä. Tietokantaan ei saa tapahtua muutoksia tietojen poiminnan ja orjan konfiguroinnin välissä.

```
mysql> show master status;
+------------------+----------+--------------+------------------+
| File             | Position | Binlog_Do_DB | Binlog_Ignore_DB |
+------------------+----------+--------------+------------------+
| mysql-bin.000015 |    89981 | openca       |                  |
+------------------+----------+--------------+------------------+
1 row in set (0.00 sec)
```

KUVIO 17. "Show master status;"-komennon tuloste

5.7.2 Slave

Toteutuksessa konfiguroidaan yhteensä kolme Slave-nodea. Alla on esimerkki Sub CA Slave-noden konfiguroinnista. Muiden orjien konfiguroinnissa tulee huomioida erityisesti MASTER_HOST, MASTER_LOG_FILE ja MASTER_LOG_POS -parametrit, jotka saadaan kohdan 5.7.1 ”show master status;”-komennon tulosteesta.

CHANGE MASTER TO MASTER_HOST='<hostname>', MASTER_USER='<käyttäjä>', MASTER_PASSWORD='<salasana>', MASTER_LOG_FILE='mysql-bin.000015', MASTER_LOG_POS=89981;

START SLAVE;

Lopuksi tarkastetaan orjan tila komennolla:

SHOW SLAVE STATUS \G;

Tarkastetaan oikea Master_Host ja käyttäjä, sekä kohdat Slave_IO_Running ja Slave_SQL_Running tulee lukea “Yes”. Tulosteesta voidaan vielä tarkistaa, että aikaisemmin tehdyt ignore-asetukset näkyvät oikein.

5.8 Varmentajien tietoturva

5.8.1 SSL:n poistaminen käytöstä konfigurointivaiheessa

RA-käyttöliittymässä on oletuksena SSL käytössä, tämä kannattaa ottaa pois päältä konfigurointivaiheessa, koska Apachelle ei ole vielä luotu sertifikaattia. Pääsynhallinnan asetukset löytyvät ”access_control”-kansiosta, jossa on jokaiselle käyttöliittymälle oma tiedosto. Muokataan aina ”.template”-tiedostoja, koska itse ”.xml”-tiedostot ylikirjoitetaan ”.template”-tiedostojen perusteella.

nano /opt/openca/etc/openca/access_control/ra.xml.template

<protocol>.*</protocol>
<symmetric_keylength>0</symmetric_keylength>

sudo /etc/init.d/openca restart

5.8.2 CA operaattorin (CA Operator) sertifikaatin luonti

CA operaattorin sertifikaattia käytetään muun muassa RA:lle tulevien sertifikaattipyyntöjen allekirjoittamiseen ja X.509-pohjaiseen kirjautumiseen. X.509-pohjaisen kirjautumisen ollessa käytössä, vain CA Operator sertifikaatin omistaja voi kirjautua ylläpidolle tarkoitettuihin käyttöliittymiin, kuten CA ja RA. Pääsynhallinta konfiguroidaan pääsylistojen avulla (ks. kohta 5.8.3).

RA:lle voidaan myöntää myös erikseen RA operaattorin sertifikaatti, mutta tässä toteutuksessa CA operaattori hallinnoi sekä CA:ta, että RA:ta. Tässä toteutuksessa CA operaattorin sertifikaattia käytetään vain juuren käyttöliittymässä.

Voimassaoloaika juuren operaattorilla on 10 vuotta ja avaimen pituus 2048 bittiä.

Kohdasta “PKI Init & Config -> Initialization -> CA Administrator” löytyy tarvittavat toimenpiteet CA operaattorin sertifikaatin luomiseen. Luodaan ensin uusi pyyntö kohdasta ”Create a new request”. Täytetään pakolliset kohdat koskien sertifikaatin pyytäjää:

First Name = <etunimi>
Last Name = <sukunimi>
E-Mail Address = <email>

Sertifikaatin tiedot:

Subject Name = <nimi>
Certificate Request Group = Users
Certificate Template = CA Operator
Selected Registration Authority = Trustcenter itself
Level of Assurance = High
Key Generation Mode = Browser (Your Computer)

Määritetään PIN-koodi, jota käytetään sertifikaattipyynnön todentamiseen ja salaisen avaimen salaamiseen. Tässä tapauksessa koodilla ei ole suurta merkitystä, koska pyynnön tekijä on ylläpitäjä itse ja avaimen generointi tapahtuu paikallisesti selaimen avainsäilöön. Koodin vähimmäispituus on 5 merkkiä.

Valitaan avaimen vahvuudeksi (Key Strength) High Grade ja luodaan pyyntö.

Myönnetään sertifikaatti kohdasta “PKI Init & Config -> Initialization -> CA Administrator -> Issue the certificate”. Ennen sertifikaatin myöntämistä, muokataan pyyntöä kohdasta “Edit Request”, johon määritetään sertifikaatin ikä:

Valid for ## days = 3650

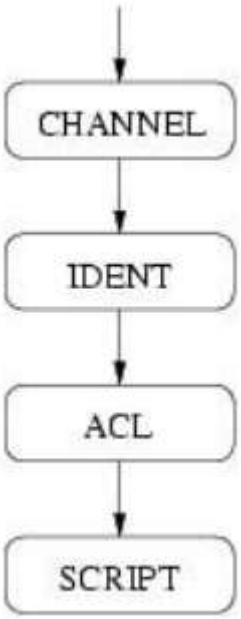
Lopuksi myönnetään sertifikaatti kohdasta ”Issue certificate”.

Sertifikaatti täytyy vielä asentaa selaimen säilöön, jotta sillä voidaan allekirjoittaa. Kohdasta “PKI Init & Config -> Initialization -> CA Administrator -> Handle the certificate” valitaan “More Info…”. Painetaan “Install Certificate”, jonka jälkeen sertifikaatti asennetaan selaimen säilöön ja sitä voidaan käyttää allekirjoittamiseen.

5.8.3 Todennus ja pääsynhallinta

OpenCA:n pääsynhallinta koostuu neljästä eri osasta (ks. kuvio 18), joista jokainen on eristetty toisistaan, lukuun ottamatta kirjautumista ja istunnonhallintaa (OpenCA Group 2005, 43):

1. Kanavan varmistus
2. Kirjautuminen
3. Istunnonhallinta
4. Pääsylistat



KUVIO 18. OpenCA pääsynhallinta (OpenCA Group 2005, 44)

Konfiguroidaan X.509-pohjainen todennus juurelle, joka tässä tapauksessa tarkoittaa, että juuren käyttöliittymään voi kirjautua vain juuren myöntämällä CA operaattorin sertifikaatilla. Alivarmentajat käyttävät normaalia käyttäjätunnus ja salasana -yhdistelmää. OpenCA tukee monipuolisia pääsynhallinta-asetuksia, esimerkiksi eri käyttäjille voidaan sallia vain tietyt operaatiot eri käyttöliittymissä. OpenCA tukee hyvin roolipohjaista pääsynhallintaa.

Pääsynhallinnan asetukset löytyvät ”access_control”-kansiosta, jossa on jokaiselle käyttöliittymälle tarkoitettu konfigurointitiedosto erikseen. Otetaan kirjautuminen käyttöön jokaisessa juuren käyttöliittymässä. Alla on esitetty käyttöönotto CA-

käyttöliittymässä. X.509-pohjaisen kirjautumisen käyttöönotto on yksinkertaista. Tiedostoon kommentoidaan kaikki muut kirjautumistavat login-kohtaan, paitsi X.509. Parametrille tarvitsee vain ilmoittaa CA-ketjun sijainti (OpenCA Group 2005, 47):

nano /opt/openca/etc/openca/access_control/ca.xml.template

<type>x509</type>
<chain>/opt/openca/var/openca/crypto/chain</chain>

Alivarmentajille jätetään kirjautumisasetukset vakioiksi. Tiedostoista löytyy myös muita pääsynhallinnan asetuksia. Otetaan käyttöön SSL kaikkien varmentajien kaikissa käyttöliittymissä. Source-asetuksella voidaan helposti suodattaa sallitut IP-osoitteet, esimerkiksi sallitaan ainoastaan hallinta-VLAN:sta tulevat IP-osoitteet. Mikäli suodatusta ei haluta tehdä, asetetaan source-kohtaan ”.*”-arvo. Alla on esimerkki juuren CA-käyttöliittymän konfiguraatiosta, jossa otetaan SSL käyttöön ja sallitaan pääsy ainoastaan paikallisesta localhost-osoitteesta (OpenCA Group 2005, 44).

nano /opt/openca/etc/openca/access_control/ca.xml.template

```
<channel>
  <type>mod_ssl</type>
  <protocol>ssl</protocol>
  <source>localhost</source>
  <asymmetric_cipher>.*</asymmetric_cipher>
  <asymmetric_keylength>0</asymmetric_keylength>
  <symmetric_cipher>.*</symmetric_cipher>
  <symmetric_keylength>0</symmetric_keylength>
</channel>
```

Tiedoston lopusta löytyy myös pääsylistoihin liittyvät konfiguraatiot, johon voidaan asettaa esimerkiksi käytettävän pääsylistan sijainti. Vakiona tarkemmat konfiguraatiot pääsylistoille löytyvät kansiosta ”rbac”. X.509-pohjaisen kirjautumisen käyttöönotto ei vielä itsessään tuo turvaa, koska pääsynhallintaa kontrolloidaan pääsylistoilla, joihin määritetään eri roolit ja rooleille sallitut operaatiot. Kun sertifikaatti myönnetään, saa se tietyn roolin. Esimerkiksi CA:n hallintaan myönnettiin CA operaattorin

sertifikaatti, jonka rooli on vakiona CA Operator. Käytetyt roolit löytyvät tiedostosta /opt/openca/etc/openca/rbac/roles.xml. Uusia rooleja voidaan luoda vapaasti kyseiseen tiedostoon, uusi rooli pitää lisätä myös OpenSSL ”extfiles”-kansioon. Konfiguroidaan kaikki juuren toimenpiteet sallituksi ainoastaan CA operaattorille (OpenCA Group 2005, 48 - 49):

nano /opt/openca/etc/openca/rbac/acl.xml.template

```
<openca>
 <access_control>
    <acl>
  <permission>
    <module>.*</module>
    <role>CA Operator</role>
    <operation>.*</operation>
    <owner>.*</owner>
  </permission>
  </acl>
   </access_control>
</openca>
```

Myös käyttäjätunnus ja salasana -pohjaisessa kirjautumisessa voidaan eri käyttäjät liittää tiettyihin rooleihin. Tämä tapahtuu muokkaamalla ”access_control”-kansion tiedostoja, johon voidaan lisätä eri käyttäjiä ja liittää käyttäjät ”roles.xml”-tiedoston sisältämiin rooleihin. Esimerkiksi vakiokäyttäjä, joka on konfiguroitu ”config.xml”-tiedostoon, on liitetty vakiona CA operaattoriksi (OpenCA Group 2005, 48):

```
        <user>
          <name>@default_web_username@</name>
          <algorithm>sha1</algorithm>
          <digest>@default_web_password@</digest>
          <role>CA Operator</role>
        </user>
```

Jotta sertifikaatit ja käyttäjät liitetään rooleihin, tulee “access_control”-tiedostoon määrittää map_role-parametri tilaan ”yes”. Parametri on vakiona ”yes”-tilassa kaikissa käyttöliittymissä, lukuun ottamatta julkista käyttöliittymää.

<map_role>yes</map_role>

Lopuksi ajetaan OpenCA:n konfigurointiskripti ja käynnistetään OpenCA uudelleen, jotta tehdyt muutokset astuvat voimaan:

sudo bash /opt/openca/etc/openca/configure_etc.sh
sudo /etc/init.d/openca restart

5.8.4 SSL:n käyttöönotto Apachelle

SSL otetaan käyttöön jokaiselle varmentajalle ja Repository:lle. Varmentajat allekirjoittavat itse SSL varmenteen omalle web-palvelimelle, lukuun ottamatta Repository:ä, jonka varmenteen allekirjoittaa toinen alivarmentajista. Lisäksi pakotetaan SSL päälle OpenCA:n kansioihin Rewrite-moduulin avulla.

Laitetaan SSL- ja Rewrite-moduuli päälle Apachelle komennolla:

sudo a2enmod ssl
sudo a2enmod rewrite

Luodaan uusi sertifikaattipyyntö käyttäen hyväksi OpenSSL-kirjastoa. Luodaan ensin salainen avain. Valitaan epäsymmetriseksi salausalgoritmiksi 1024 bittinen RSA:

openssl genrsa -out apache.key 1024

Luodaan pyyntö käyttäen hyväksi luotua salaista avainta:

openssl req -new -key apache.key -out apachereq.csr

Tehdään palvelimille tarkoitettu sertifikaattipyyntö CA:n julkisessa käyttöliittymässä ja allekirjoitetaan pyyntö CA:n toimesta.

Ladataan pyyntö PEM formaatissa web-palvelimen kansioon /etc/apache2. Luodaan avaimille tarkoitettu kansio, kopioidaan luotu salainen avain sinne ja rajoitetaan kansion oikeudet.

sudo mkdir /etc/apache2/keys
sudo mv apache.key /etc/apache2/keys
sudo chmod -R 700 /etc/apache2/keys

Lisätään HTTPS-asetukset Apachen ”virtual hosts”-tiedostoon (ks. liite 6):

sudo nano /etc/apache2/sites-enabled/000-default

Lopuksi käynnistetään Apache uudelleen, jotta muutokset astuvat voimaan.

sudo /etc/init.d/apache2 restart

5.8.5 Tiedonvaihto

Tiedonvaihto juuren ja sertifikaattisäilön välillä toteutetaan SSH:lla ja autentikaatio avaimilla. Repository-palvelimelle asennetaan OpenSSH-palvelin, jota juuri pystyy käyttämään tiedonvaihtoon. Juuri on yhteydellisessä tilassa vain tiedonsiirtoon kuluvan ajan.

Käyttäjänä root, luodaan ”.ssh”-kansio juuren polkuun /var/www ja siirrytään sinne:

mkdir /var/www/.ssh
cd /var/www/.ssh

Luodaan uusi avainpari ilman salasanaa, jolle annetaan nimeksi esimerkiksi id_rsa_1024_opencawww:

ssh-keygen -t rsa -b 1024

Vaihdetaan ”.ssh”-kansion ja sen sisällön omistajaksi www-data, eli Apache. Asetetaan ainoastaan omistajalle täydet oikeudet kansiolle ja sen sisällölle:

chown -R www-data:www-data /var/www/.ssh
chmod -R 700 /var/www/.ssh

Kopioidaan ”.pub” -päätteinen julkinen avain host-koneelle (Repository), esimerkiksi käyttäen SCP:tä ja roottia käyttäjänä.

scp /var/www/.ssh/id_rsa_1024_opencawww.pub root@<hostname>:/var/www/

Seuraavaksi siirrytään Repository-palvelimelle ja luodaan vastaavasti kansiot kuin juurella, mutta ei luoda avainparia. Kansion luonnin jälkeen kopioidaan siirretty julkinen avain ”/var/www”-kansiosta ”.ssh”-kansioon:

mkdir /var/www/.ssh
mv /var/www/id_rsa_1024_opencawww.pub /var/www/.ssh

Jotta avain sallitaan, tulee se kopioida ”.ssh”-kansiossa olevaan ”authorized_keys”-tiedostoon. Jos tiedostoa ei ole, luodaan se:

cp /var/www/.ssh/id_rsa_1024_opencawww.pub /var/www/.ssh/authorized_keys

Jos tiedosto on jo ennestään, voidaan avain liittää sen perään (append):

cat /var/www/.ssh/id_rsa_1024_opencawww.pub >>
/var/www/.ssh/authorized_keys

Varmistetaan vielä oikeudet kansiolle ja sen sisällölle:

chown -R www-data:www-data /var/www/.ssh
chmod -R 700 /var/www/.ssh

Asennetaan OpenSSH Repository-palvelimelle komennolla:

sudo apt-get install ssh

Lopuksi kytketään RSA- ja Pubkey-autentikaatio päälle ja salasana-autentikaatio pois päältä:

nano /etc/ssh/sshd_config

RSAAuthentication yes

PubkeyAuthentication yes

PasswordAuthentication no

/etc/inid.d/ssh restart

Seuraavaksi muokataan node-käyttöliittymän konfiguraatiota juurella. Muutetaan EXPORT_IMPORT-asetuksia siten, että juuri käyttää SCP:tä tiedonvaihtoon ja käynnistetään lopuksi OpenCA uudelleen:

nano /opt/openca/etc/openca/servers/node.conf.template

dataexchange with a lower level of the hierarchy

EXPORT_IMPORT_DOWN_DEVICE "root_to_repo.tar"
EXPORT_IMPORT_DOWN_START "sudo /sbin/ifup eth0"
EXPORT_IMPORT_DOWN_STOP "sudo /sbin/ifdown eth0"
EXPORT_IMPORT_DOWN_EXPORT "/bin/tar -cvpf /opt/openca/var/openca/tmp/@__DEVICE__@ -C @__SRC__@ ." "/usr/bin/scp -i /var/www/.ssh/id_rsa_1024_opencawww /opt/openca/var/openca/tmp/@__DEVICE__@ www-data@<hostname>:/opt/openca/var/openca/tmp/" "rm /opt/openca/var/openca/tmp/@__DEVICE__@"
EXPORT_IMPORT_DOWN_TEST ""

sudo bash /opt/openca/etc/openca/configure_etc.sh
sudo /etc/init.d/openca restart

Jotta Apachen käyttäjä pystyy kytkemään verkkorajapinnan päälle pois, täytyy sille lisätä sudo oikeudet kyseiseen komentoon:

nano /etc/sudoers
www-data ALL=NOPASSWD:/sbin/ifup eth0, /sbin/ifup eth0

Määritetään tiedonvaihtoasetukset myös Repository:lle ja käynnistetään OpenCA uudelleen:

nano /opt/openca/etc/openca/servers/node.conf.template

dataexchange with a higher level of the hierarchy

EXPORT_IMPORT_UP_DEVICE "<polku tar-tiedostolle>"
EXPORT_IMPORT_UP_START ""
EXPORT_IMPORT_UP_STOP ""
EXPORT_IMPORT_UP_EXPORT "/bin/tar -cvpf @__DEVICE__@ -C @__SRC__@ ."
EXPORT_IMPORT_UP_IMPORT "/bin/tar -xvf @__DEVICE__@ -C @__DEST__@"
EXPORT_IMPORT_UP_TEST "/bin/tar -tvf @__DEVICE__@"

sudo bash /opt/openca/etc/openca/configure_etc.sh
sudo /etc/init.d/openca restart

Lopuksi voidaan testata tiedonvaihtoa viemällä tar-paketti alemmalle hierarkia-tasolle eli Repository:lle:

Enroll data to a lower level of the hierarchy -> ALL.

Käyttöliittymästä on myös mahdollista viedä vain osa tiedoista, esimerkiksi pelkästään uudet sertifikaatit. Tässä toteutuksessa ainoastaan uusien sertifikaattien ja sulkulistojen toimittaminen Repository:lle on tarpeellista. Repository:n tiedonvaihtoasetuksiin on määritelty (ks. liite 7), että se voi vastaanottaa ainoastaan uuden CA-sertifikaatin ja uusia sertifikaatteja sekä sulkulistoja. OpenCA tunnistaa, mikäli sama sertifikaatti löytyy jo ennestään tietokannasta.

Seuraavaksi tuodaan uudet tiedot Repository:n node-käyttöliittymästä kohdasta: Download data from a higher level of the hierarchy -> ALL.

Lopuksi OpenCA antaa raportin tuonnin onnistumisesta ja kertoo mitkä kohteet lisättiin tietokantaan.

6 TULOKSET

6.1 Yleistä

Työn lopputuloksena saatiin toimiva ja laaja PKI-järjestelmä, joka palvelee tarkoitustaan JYVSECTEC:n simulaatioympäristön sertifikaattien tarjoajana hyvin. Järjestelmän Sertifikaatteja voidaan myöntää useisiin eri käyttötarkoituksiin, muun muassa web-palvelimille, VPN-palvelimille, sähköpostipalvelimille, tavallisille käyttäjille ja Domain Controllereille. Järjestelmä on myös laajasti muokattavissa ja laajennettavissa. Uusia varmentajia voidaan lisätä helposti valmiista pohjasta ja eri osat voidaan muokata eri tarpeisiin sopivaksi vapaan lähdekoodin vuoksi. Järjestelmä ei missään nimessä ole täydellinen ja aukoton.

OpenCA:n käytössä ilmeni useita ongelmia ja pieniä bugeja, joita ei kaikkia onnistuttu korjaamaan. Suurin ongelma oli se, että alivarmentajat eivät osaa hyödyntää OpenCA:n toiminnoissa CA-ketjua, vaikka CA-ketju on luotu oikein ”/opt/openca/var/openca/crypto/chain”-kansioon dokumentaation perusteella. Ketjua tarvitaan RA:lle tulevien pyyntöjen allekirjoittamiseen. Tästä syystä RA:n pyynnöt hyväksytään ilman operaattorin allekirjoitusta, mikä ei toisaalta haittaa, koska loppukäyttäjillä ei ole pääsyä RA-käyttöliittymään. Ongelma ei kuitenkaan vaikuta itse sertifikaatin allekirjoittamiseen ali CA:n toimesta, koska ketju toimii oikein esimerkiksi selaimessa. Juurella pyyntöjen allekirjoitus operaattorin sertifikaatilla toimii ongelmitta. Ongelma ei ratkennut kattavasta vianselvityksestä huolimatta, apua haettiin myös yhteisöltä tuloksetta.

6.2 Sertifikaattien myöntäminen

6.2.1 Yleistä

Sertifikaatin hakuprosessi löytyy kokonaisuudessaan liitteestä 3. Prosessi kuvaa sertifikaatin hakemista käyttöönotetussa järjestelmässä ja se pätee molempiin alivarmentajiin, mutta ei juurivarmentajaan. Juurivarmentaja ei myönnä sertifikaatteja loppukäyttäjille, lukuun ottamatta sen OCSP Responderia. Myöntämisprosessiin liittyy useita eri järjestelmän osia:

- Loppukäyttäjä
- CA operaattori
- Automaatio
- Tietokanta
- Julkinen käyttöliittymä
- RA käyttöliittymä
- CA käyttöliittymä

6.2.2 Käyttäjäsertifikaatin hakeminen

Todennetaan prosessi hakemalla testikäyttäjälle sertifikaatti ja lopuksi asentamalla sertifikaatti selaimen säilöön. Ensin luodaan uusi pyyntö alivarmentajan JST Sub CA julkisesta käyttöliittymästä (ks. kuvio 19):

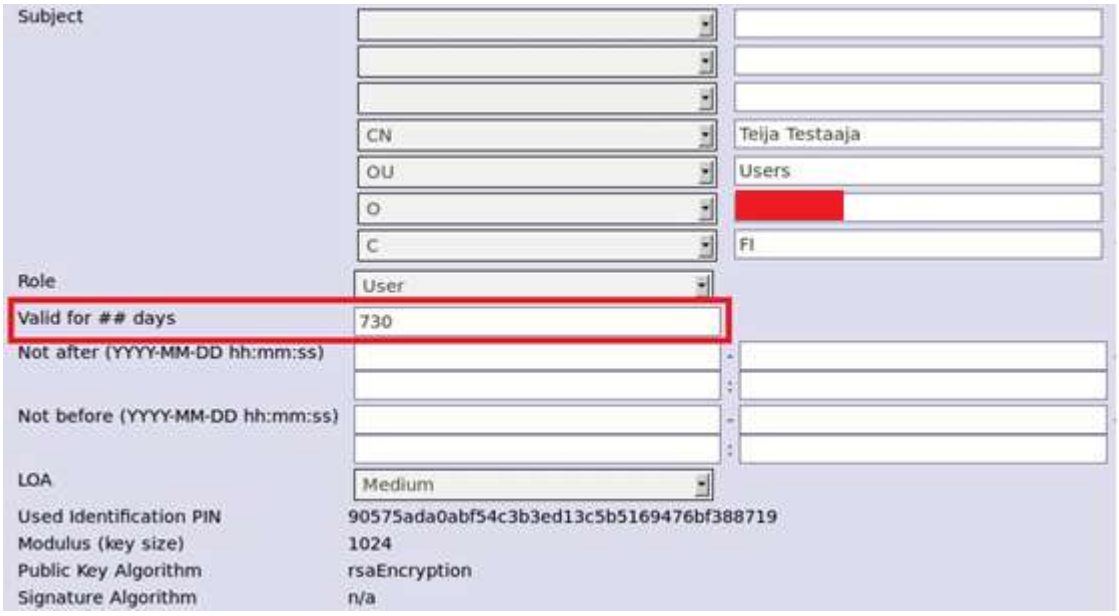
KUVIO 19. Sertifikaattipyynnön tiedot

Valitaan käyttäjän asymmetrisen avaimen vahvuudeksi ”Medium Grade” (ks. kuvio 20), joka tarkoittaa käytännössä 1024 bittistä avainta.

KUVIO 20. Asymmetrisen avaimen vahvuus

Ennen pyynnön hyväksymistä, tulee CA operaattorin tarkastaa ja todentaa pyynnön tekijä sekä pyyntöön syötetyt tiedot. Pyyntöä tulee myös muokata siten, että sille

syötetään vähintään sertifikaatin elinikä. Kuvion 21 mukaan käyttäjälle myönnetään kahden vuoden pituinen sertifikaatti, pituus tulee syöttää päivissä.



KUVIO 21. CA operaattorin muokkausnäkymä

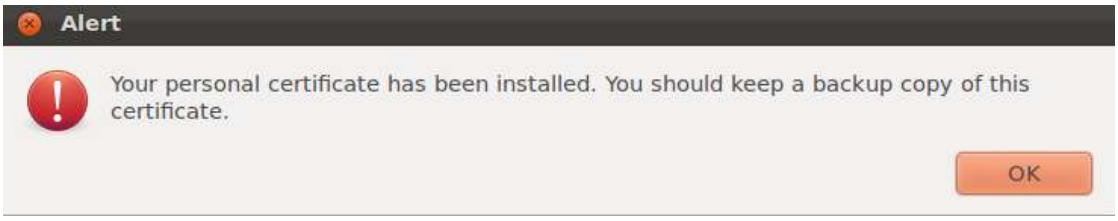
Seuraavaksi CA operaattori hyväksyy pyynnön. Hyväksyttyjen pyyntöjen perusteella CA:n automaatio myöntää sertifikaatin (ks. kohta 5.4.5), joten lopullinen sertifikaatin myöntäminen ei enää vaadi manuaalisia toimenpiteitä ylläpidolta. Automaatio hoitaa myös pyynnön ja sertifikaatin arkistoimisen tietokantaan sekä sertifikaatin replikoinnin toiselle alivarmentajalle ja Repository:lle. Replikoinnin jälkeen myönnetty sertifikaatti on saatavilla esimerkiksi Repository:n julkisen käyttöliittymän kautta (ks. kuvio 22).

KUVIO 22. Sertifikaatin automaattinen replikointi Repository:lle

Koska pyyntöä tehdessä salaisen avaimen generointiin käytettiin paikallista selainta, ainoastaan salaisen avaimen hallussapitäjä, joka tässä tapauksessa on paikallisen selaimen omistaja, voi asentaa sertifikaatin sen sarjanumeron perusteella (ks. kuvio 23).

KUVIO 23. Sertifikaatin noutaminen julkisen käyttöliittymän kautta

Kuvion 24 ilmoituksen perusteella voidaan todeta, että sertifikaatin pyyntöprosessi onnistui ja asennettua sertifikaattia voidaan käyttää allekirjoittamiseen.



KUVIO 24. Sertifikaatin onnistunut vieminen selaimen säilöön

6.2.3 Palvelinsertifikaatin hakeminen

Palvelimelle tarkoitettua pyyntöä tehdessä itse prosessi ei eroa, mutta tässä tapauksessa salainen avain ja pyyntö generoidaan yleensä valmiiksi esimerkiksi OpenSSL:n komennon avulla paikallisesti palvelimelle. Generoitu pyyntö tarvitsee ainoastaan lähettää julkisen käyttöliittymän kautta ja määrittää haluttu rooli. Sertifikaattia ei myöskään tarvitse asentaa, vaan ainoastaan ladataan julkinen PEM-koodattu sertifikaatti palvelimelle.

Todetaan toiminta hakemalla Apachelle domainiin ”testi.com” kahden vuoden sertifikaatti. Käytetään hyväksi OpenSSL-kirjastoa luomalla uusi salainen avain paikallisesti, jonka pituus on 1024 bittiä (ks. kuvio 25).

```
:~$ openssl genrsa  -out apache.key 1024
Generating RSA private key, 1024 bit long modulus
.............++++++
.....................................++++++
e is 65537 (0x10001)
```

KUVIO 25. Salaisen avaimen generointi OpenSSL:llä

Luodun avaimen perusteella generoidaan uusi sertifikaattipyyntö seuraavilla tiedoilla (ks. kuvio 26.):

- Country Name: FI
- Organization Name: Testi Organisaatio
- Organizational Unit Name: Web
- Common Name: testi.com
- Email Address: admin@testi.com

Tärkein tieto on Common Name, jonka perusteella sertifikaatti yhdistetään domainiin.

```
:~$ openssl req -new -key apache.key -out apachereq.csr
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:FI
State or Province Name (full name) [Some-State]:
Locality Name (eg, city) []:
Organization Name (eg, company) [Internet Widgits Pty Ltd]:Testi Organisaatio
Organizational Unit Name (eg, section) []:Web
Common Name (e.g. server FQDN or YOUR name) []:testi.com
Email Address []:admin@testi.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

KUVIO 26. Uusi sertifikaattipyyntö OpenSSL:llä

Kuviosta 27 voidaan todeta, että yllä suoritetut komennot luovat ”.key” ja ”.csr” -päätteiset tiedostot kansioon, johon ”-out”-optio osoittaa. Key-tiedosto on salainen avain, jota ei tässä tapauksessa suojattu salasanalla. Mikäli salasana määritetään Apachen salaiselle avaimelle, joudutaan se syöttämään joka kerta Apachen käynnis-

tyessä. Tämä lisää ylläpitoa ja huonontaa saatavuutta esimerkiksi uudelleenkäynnistyksen yhteydessä. Parempi tapa on määrittää avaimen lukuoikeudet ainoastaan root-käyttäjälle. Csr-tiedosto on sertifikaattipyyntö, joka pitää allekirjoittaa CA:n toimesta.

KUVIO 27. OpenSSL:n generoimat tiedostot

Seuraavaksi ”.csr”-tiedosto lähetetään CA:lle allekirjoitettavaksi julkisen käyttöliittymän kautta. Pyynnön tulee olla PKCS#10 tyyppinen, joka on formaatti sertifikaattipyynnölle ja käytännössä tarkoittaa sitä, että pyyntö on luotu paikallisesti hakijan toimesta. Vaikka pyyntö on jo täytetty ennakkoon paikallisesti hakijan toimesta, tulee pyynnön toimittamisen yhteydessä vähintään määrittää haluttu rooli, varmuustaso ja PIN-koodi (ks. kuvio 28).

KUVIO 28. PKCS#10 palvelinpyyntö

Seuraavat vaiheet ovat samantyyppisiä kuin käyttäjäsertifikaatin myöntämisessä (ks. kohta 6.2.2), lukuun ottamatta sertifikaatin asentamista. Määritetään RA-käyttöliittymän kautta CA operaattorin toimesta sertifikaatin iäksi kaksi vuotta ja hyväksytään pyyntö. Hyväksymisen jälkeen automaatio myöntää sertifikaatin (ks. kohta 5.4.5). Kuviosta 29 voidaan todeta, että palvelimelle saatiin haettua onnistuneesti

sertifikaatti. Kuten aiemmin mainittiin, sertifikaatin asentaminen eroaa sillä, että se haetaan palvelimelle PEM-muodossa ja käytetään yhdessä paikallisesti luodun salaisen avaimen kanssa (ks. kohta 5.8.4).

KUVIO 29. Voimassaoleva palvelinsertifikaatti

6.3 Sertifikaattien sulkeminen

Sertifikaatin sulkuprosessi löytyy kokonaisuudessaan liitteestä 4. Prosessi kuvaa sertifikaatin sulkemisen sekä itse käyttäjän että CA operaattorin aloitteesta. Prosessi pätee kaikkiin varmentajiin, lukuun ottamatta juurta.

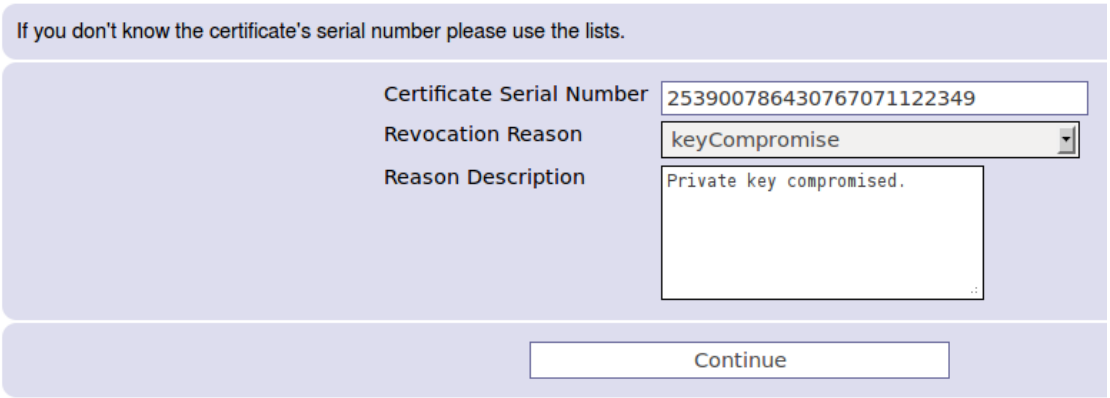
Juurella prosessi eroaa sillä, että käyttäjä ei voi tehdä aloitetta sulkemiselle, vaan ainoastaan CA operaattori. Juurella myös sulkulistan julkaisu eroaa muista, koska uutta sulkulistaa ei julkaista automaattisesti tietyn ajan kuluessa, vaan uuden julkaisun tekee aina CA operaattori. Sulkulista tulee myös toimittaa manuaalisesti SCP:n avulla node-käyttöliittymästä Repository:lle (ks. kohta 5.8.5). Repository vastaa juuren OCSP-palvelusta ja sulkulistan säilytyksestä. Juuren sulkulistan koko on pieni ja julkaisuväli pitkä, koska se myöntää sertifikaatit ainoastaan omalle OCSP Responderille, alivarmentajille, omalle web-palvelimelle ja Root CA operaattorille. Koska juuri sulkee sertifikaatteja harvoin, ei juuren CA-käyttöliittymään ole konfiguroitu automaatiota, joka automaattisesti sulkee sertifikaatin kun pyyntö hyväksytään RA-käyttöliittymässä. Tarkoituksena on myös estää virheelliset sulkemiset, koska alivarmentajan sertifikaatin sulkeminen sulkee myös kaikki sen alla toimivat loppukäyttäjät.

Todennetaan prosessi sulkemalla aikaisemmassa kohdassa luotu testikäyttäjän sertifikaatti CA operaattorin aloitteesta. Lisäksi todennetaan OCSP-palvelun toiminta manuaalisten OpenSSL-kyselyiden avulla sulkuprosessin aikana.

Kuviosta 30 voidaan todeta, että ennen sulkemista tehty kysely OCSP-palvelulle palauttaa sertifikaatin tilaksi ”good”, eli sertifikaatti on voimassa.

KUVIO 30. OCSP-kysely ennen sulkemista

Luodaan uusi sertifikaatin sulkupyyntö RA-käyttöliittymän kautta. Valitaan sertifikaatin sulkemisen syyksi salaisen avaimen paljastuminen, tiedot kenttään voidaan lisätä mahdollisia lisätietoja (ks. kuvio 31).



KUVIO 31. Sertifikaatin sulkupyyntö CA operaattorin toimesta

Tarkastetaan pyynnön tiedot ja hyväksytään pyyntö ilman allekirjoitusta (ks. kuvio 32). Jostain syystä tietokanta-mooduli epäonnistuu kun pyyntöä yritetään muokata tässä vaiheessa, joten mikäli pyynnössä ilmenee puutteita tai jokin virhe, tulee se poistaa ”Delete Request”-kohdasta kokonaan ja luoda uusi. Myös ”Delete Request”-toiminto epäonnistuu tietokantakyselyyn liittyvän virheen vuoksi. Ongelma voidaan kiertää poistamalla sulkupyyntö käsin tietokannasta ja poistamalla sulkuun liittyvät tiedot sertifikaattitaulusta. Kattavasta vianselvityksestä huolimatta ongelmaan ei löytynyt ratkaisua. Sama ongelma löytyy myös yhteisön postituslistalta, mutta toistaiseksi ratkaisua tälle ei löytynyt. Ongelmaan haettiin tukea myös yhteisöltä, mutta

ratkaisua ei löytynyt. Perl-kielen osaavalle henkilölle ongelman korjaaminen pysyvästi ei varmasti tuota ongelmia.

KUVIO 32. Sulkupyynnön tarkastaminen ja hyväksyminen

Sulkupyynnön hyväksymisen jälkeen CA-käyttöliittymän automaatio käsittelee hyväksytyn sulkupyynnön ja sulkee sertifikaatin (ks. kohta 5.4.5). Kuviosta 33 voidaan todeta, että sertifikaatti on lisätty suljettujen sertifikaattien joukkoon.

KUVIO 33. Sertifikaatti on suljettu avaimen paljastumisen vuoksi

Jotta suljettu sertifikaatti tulee koko PKI-järjestelmän tietoon, tulee se julkaista sulkulistalla. Uusi sulkulista julkaistaan automaation toimesta joka päivä (ks. kohta 5.4.5),

mutta se on syytä tehdä manuaalisesti mahdollisimman pian sulkemisen jälkeen (ks. kuvio 34). Toistaiseksi OpenCA ei tue uuden sulkulistan julkaisua automaattisesti heti uuden sulkemisen jälkeen.

KUVIO 34. Sulkulistan julkaisu manuaalisesti

OCSP-palvelu on ajastettu päivittämään sen sulkulista joka minuutti, joten maksimissaan minuutin kuluttua OCSP-kysely palauttaa sertifikaatin uudeksi tilaksi ”revoked”, eli suljettu. Kysely palauttaa myös sulkemisen syyn ja sulkemisajan. Onnistunut sulkeminen ja OCSP-palvelun toiminta voidaan todeta kuviosta 35.

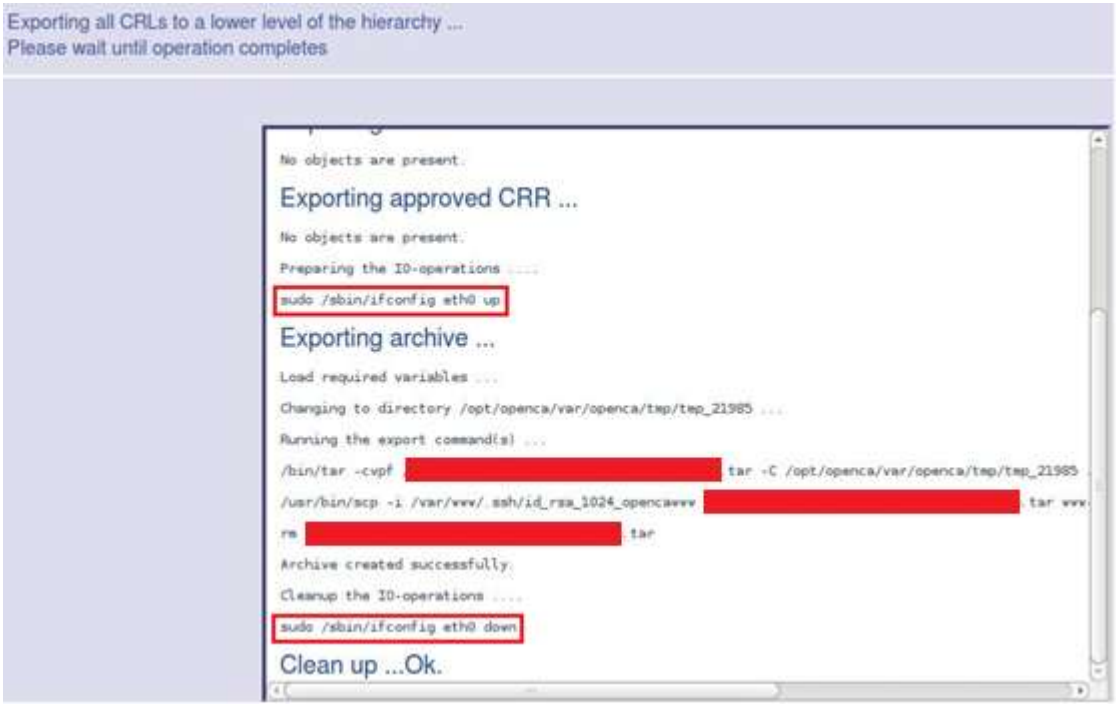
KUVIO 35. OCSP-kysely sulkemisen jälkeen

Mikäli loppukäyttäjä tekee aloitteen sulkemiselle, eroaa prosessi sillä, että loppukäyttäjä voi sulkea ainoastaan oman sertifikaattinsa CRIN-koodilla, jonka käyttäjä on saanut sähköpostilla. Sertifikaatin myöntämisen yhteydessä lähetetään CRIN-koodi käyttäjälle salattuna, salaus voidaan purkaa ainoastaan käyttäjän salaisella avaimella. Sulkemispyynnön yhteydessä CA tarkistaa CRIN-koodin oikeellisuuden, mikäli koodi on väärä, hylätään pyyntö.

6.4 Tietoturva

6.4.1 Juuren ja Repository:n tiedonvaihto

Järjestelmän tärkein ja vahvimmin suojattu osa on juurivarmentaja. Juuri toimii yhteydettömässä tilassa ja se tarvitsee verkkoyhteyden ainoastaan toimittaessaan uusia sulkulistoja tai sertifikaatteja Repository:lle. Kopioinnissa hyödynnetään SSH:n Secure Copy-toimintoa ja avaimiin perustuvaa tunnistautumista. OpenCA on konfiguroitu siten, että verkkorajapinta nostetaan ylös kopioinnin alkaessa ja lasketaan välittömästi alas kun kopiointi on suoritettu loppuun. Ratkaisu tarjoaa tietoturvallisen ja käytännöllisen tavan tiedonvaihtoon yhteydettömän juuren ja julkisen säilön välillä. Kuviosta 36 voidaan todeta turvallinen tiedonvaihto. Tiedonvaihdon alkaessa rajapinta eth0 nousee ylös, jonka jälkeen kopioidaan uudet tiedot säilöön. Kopioinnin päättyessä eth0 rajapinta lasketaan takaisin alas. Lisätietoa kopioinnista löytyy kohdasta 5.8.5.



KUVIO 36. Tiedonvaihto juuren ja julkisen säilön välillä

6.4.2 Juuren pääsynhallinta

Juuren jokainen käyttöliittymä on suojattu käyttäen hyväksi X.509-pohjaista kirjautumista, joka käytännössä tarkoittaa 2-vaiheista autentikointia. Käyttäjän tulee tietää selaimen sertifikaattisäilön salasana sekä omistaa CA operaattorin salainen avain. Joten vaikka jompikumpi näistä joutuu vääriin käsiin, ei juuren tietoturva vielä tässä vaihessa ole täysin vaarantunut. Juuren pääsynhallintaan on myös määritetty sallituksi lähdeosoitteeksi ainoastaan paikallinen localhost-osoite.

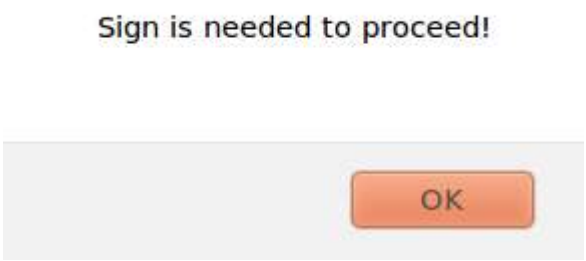
Toiminta voidaan todeta kirjautumalla sisään johonkin juuren käyttöliittymistä (ks. kuvio 37). Kirjauduttaessa juuri lähettää käyttäjälle haasteen, joka täytyy allekirjoittaa juuren myöntämällä CA operaattorin sertifikaatilla.

KUVIO 37. Juuren lähettämä haaste

Pelkkä sertifikaatin salaisen avaimen omistaminen ei riitä onnistuneeseen kirjautumiseen. Käyttäjän tulee tietää myös sertifikaattisäilön salasana ja valita CA operaattorin sertifikaatti haasteen allekirjoitukseen (ks. kuvio 38).

KUVIO 38. Haasteen allekirjoittaminen

Mikäli kirjautuminen onnistuu, siirtyy käyttäjä onnistuneesti CA:n käyttöliittymään. Mikäli kirjautuminen epäonnistuu, esimerkiksi yritetään väärällä sertifikaatilla sisään, ilmoittaa OpenCA allekirjoituksen puutteellisuudesta (ks. kuvio 39).

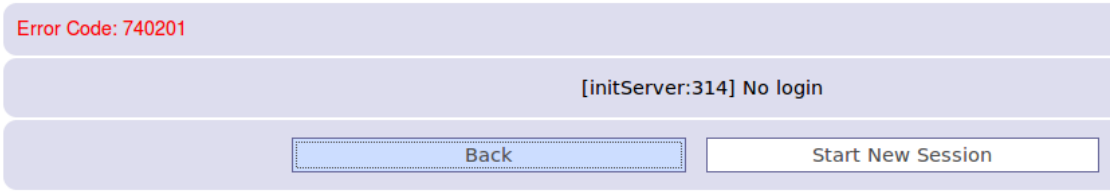


KUVIO 39. Haasteen virheellinen allekirjoitus

6.4.3 Alivarmentajien pääsynhallinta

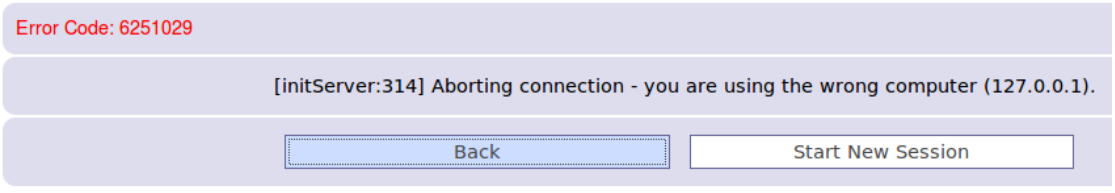
Alivarmentajat on suojattu ainoastaan yksivaiheisella tunnistuksella, joten käyttäjän tarvitsee tietää ainoastaan käyttäjätunnus ja salasana yhdistelmä. Tietoturvaa on helppo parantaa estämällä ei-halutuilta IP-osoitteilta kirjautuminen kokonaan (ks. kohta 5.8.3). Vaikka vihamielinen käyttäjä pääsisi kirjautumaan CA:n käyttöliittymään, tarvitsee käyttäjän saada tietoon myös salaista avainta suojaava salasana, jotta voidaan esimerkiksi myöntää tai sulkea vääriä sertifikaatteja.

Kuviosta 40 voidaan todeta, että OpenCA estää kirjautumisen väärillä tunnuksilla.



KUVIO 40. Epäonnistunut kirjautuminen, väärä käyttäjätunnus ja salasana yhdistelmä

Testataan IP-osoitteiden suodatusta konfiguroimalla CA-käyttöliittymän pääsynhallintaan sallituksi ainoastaan tietty IP-osoite. Kuviosta 41 voidaan todeta, että lähdeosoitteen ollessa jokin muu kuin yllä, estetään kirjautuminen.



KUVIO 41. Epäonnistunut kirjautuminen, väärä lähdeosoite

6.4.4 HTTPS

Repository:n, alivarmentajien ja juuren jokaiseen käyttöliittymään on pakotettu SSL-suojaus päälle. Myös phpMyAdmin toimii ainoastaan SSL-suojattuna. Jokainen varmentaja myöntää varmenteen omalle web-palvelimelleen, lukuun ottamatta Repository:ä, jolle varmenteen myöntää toinen alivarmentajista. HTTPS:n toiminta voidaan todeta kuviosta 42.

KUVIO 42. OpenCA ja HTTPS

6.4.5 Varmentajien salainen avain

Salainen avain on talletettu kansioon, johon ainoastaan Apachen käyttäjällä on pääsy. Lisäksi avain on salattu salasanalla. Linux on itsessään hyvin turvallinen käyttöjärjestelmä. Käytännössä salaisen avaimen paljastuminen vaatii Linuxin juuren salasanan paljastumisen ja salasanan, jolla salainen avain on salattu. Salaukseen käytetään vahvaa AES-256 symmetristä algoritmiä.

7 YHTEENVETO

7.1 Toteutus

Toteutus alkoi tutustumalla aiheeseen liittyvään teoriaan, johon kuuluvat kryptografia ja julkisen avaimen infrastruktuuri. Samalla alettiin myös kartoittaa tarjolla olevia sovelluksia. Seuraavassa vaiheessa testattiin ja vertailtiin lukuisia eri PKI-sovelluksia, ideana oli selvittää paras mahdollinen vaihtoehto toimeksiantajan tarpeisiin. Lopuksi suunniteltiin ja toteutettiin lopullinen järjestelmä, joka dokumentoitiin mahdollisimman kattavasti.

Vaikeimmaksi ja eniten aikaa vieväksi vaiheeksi osoittautui eri sovellusten asennus ja konfigurointi. Erityisesti OpenCA:n asennus osoittautui haasteelliseksi, koska ohjelma piti kääntää suoraan lähdekoodista ja se perustui muun muassa lukuisten eri Perl-moduulien sekä eri tukisovellusten yhteistoimintaan. Esimerkiksi eri moduulien versiot tuli sopia tietyn OpenCA version kanssa yhteen. Aikaisempi kokemus PKI:stä ja tarjolla olevista sovelluksista oli vähäistä. Myöskään tietyt Linuxin komennot ja toiminnot eivät olleet ennestään tuttuja, joten opettelemista riitti. OpenCA:ssa ilmeni myös useita bugeja, joiden selvittelyyn kului aikaa.

Jo työn alkuvaiheessa tiesin, että aihe on erittäin laaja ja järjestelmään voidaan ottaa käyttöön lukuisia eri toimintoja. Tästä syystä työn rajaus oli ratkaisevassa asemassa, jotta aikataulu pitää ja työ pysyy järkevissä mittasuhteissa. Mielestäni rajaus onnistui hyvin ja tärkeimmät toiminnot sekä komponentit onnistuttiin säilyttämään.

7.2 Tulokset

Uskon, että työn tuloksena saatu toimiva ja laaja PKI-infrastruktuuri tukee JYVSEC-TEC-hankkeen toimintaa ja tuo lisäarvoa sen simulaatioympäristöön. Järjestelmää voidaan helposti käyttää hallitsemaan ympäristön sisällä tarvittavia sertifikaatteja erityyppisille loppukäyttäjille. Sertifikaatteja voidaan monipuolisesti myöntää muun muassa web-palvelimille, VPN-palvelimille, henkilöille, laitteille ja toimialueen ohjauskoneille (DC). Järjestelmä tuo hyötyä myös opetuksellisesta näkökulmasta. Järjes-

telmää voidaan käyttää muun muassa demonstroimaan ja esittelemään esimerkkitoteutusta PKI-infrastruktuurista ja sen toiminnoista.

Tekemäni kartoituksen perusteella ilmaisia ja laadukkaita yritystason PKI-sovelluksia on erittäin vähän tarjolla. Ainoastaan EJBCA ja OpenCA ovat pidemmälle kehitettyjä ohjelmistoja. EJBCA:n Java-pohjaisuus ja monimutkaisuus ovat asioita, jotka toivat pisteitä OpenCA:lle. Toisaalta myöskään OpenCA:n käyttöönotto ei ollut yksinkertainen. Asennus ja konfigurointi vaati laajaa perehtymistä tarjolla olevaan viralliseen ja kolmannen osapuolen dokumentaatioon. Vapaan lähdekoodin vuoksi OpenCA on monipuolisesti muokattavissa erilaisiin tarpeisiin ja uusia toimintoja voidaan ottaa käyttöön tarpeen mukaan. Vaikka OpenCA:n kehitys on pitkällä ja sitä on työstetty useita vuosia, havaitsin useita bugeja, puutteita dokumentaatiossa ja yhteensopivuusongelmia käyttöönotossa. Onneksi suurimpaan osaan havaituista ongelmista löytyi ratkaisu muun muassa yhteisön keskustelupalstoilta. Kaiken kaikkiaan OpenCA:sta jäi minulle keskeneräinen kuva, sen käyttöönotto vaatii paljon kustomointia ja taitoa. Mielestäni se on liian keskeneräinen soveltuakseen ns. ”oikeaan” avoimeen yrityksen ympäristöön, vaan tällöin on syytä harkita EJBCA:ta, kaupallisen tuotteen hankintaa tai koko PKI:n ulkoistamista kolmannelle osapuolelle.

7.3 Pohdinta

Älykkäiden mobiililaitteiden lisääntyessä yritysten tietoturvan merkitys korostuu. Monet työntekijät tuovat omia laitteita työpaikalle BYOD (Bring your own device) -periaatteella. Yksi tapa laitteiden kontrollointiin on sertifikaatit. Laitteiden pääsyä esimerkiksi verkkoresursseihin voidaan helposti kontrolloida sertifikaattien avulla. Jokaiselle laitteelle myönnetään oma sertifikaatti, jolle myönnetään tietyt oikeudet tiettyihin yrityksen resursseihin. Sertifikaatti voidaan helposti sulkea havaittaessa väärinkäytöksiä. Suosituimmista valmistajista ainakin Applen mobiilituotteet ovat tukeneet jo iOS 4-käyttöjärjestelmästä asti natiivisti SCEP-protokollaa, jolla voidaan helposti ja vaivattomasti myöntää eri laitesertifikaatteja. Ainakin Applen iOS tukee myös natiivisti sertifikaattipohjaista autentikointia 802.1X:n, VPN:n ja ActiveSync:n kanssa. Mielestäni Android-laitteet ovat selkeästi Applea jäljessä tässä asiassa SCEP-natiivituen puuttuessa. Kolmansien osapuolien puolesta esimerkiksi Cisco tarjoaa

ratkaisua tähän Identity Services Engine (ISE)-tietoturvapalvelimella, joka toimii ns. SCEP-proxynä. Uskon SCEP-protokollatuen laajentuvan tulevaisuudessa mobiililaitevalmistajien keskuudessa myös esimerkiksi Android laitteille.

Uskon, että vapaan lähdekoodin PKI-sovellukset eivät tule merkittävästi lisääntymään tulevina vuosina. OpenCA:n kehitys on selkeästi hidastunut viime vuosina ja se soveltuukin paremmin juuri esimerkiksi testi- ja simulaatioympäristöihin. Opetuksellisesta näkökulmasta ja toimeksiantajan tarpeiden perusteella OpenCA on hyvä vaihtoehto sen muokattavuuden, keveyden ja selkeyden puolesta. EJBCA on selkeästi paras ja käytetyin ilmainen yritystason PKI-ratkaisu, jota myös kehitetään aktiivisesti. Laadukkaita kaupallisia sovelluksia on puolestaan runsaasti tarjolla. Esimerkiksi Safelayer, OpenTrust, Microsoft, CoSign tarjoaa monipuolisen ja turvallisen ”in-house” PKI -ratkaisun. Näistä ainakin Microsoftin, Safelayerin ja OpenTrustin tuotteille on myönnetty korkean luokan Common Criteria -sertifiointi. Microsoftin selkeä etu on sen täydellinen integroituminen aktiivihakemiston kanssa.

SCEP ja AD-integraatio olisi varmasti hyviä kohteita opinnäytetyöni jatkokehitykselle. OpenCA:sta löytyy vakiona SCEP-protokollatuki ja se toisi varmasti helpotusta laitesertifikaattien myöntämiseen. Myös järjestelmän laajentaminen ja tietoturvan parantaminen ovat hyviä kehityskohteita.

SCEP:n käyttöönotto palvelimen päässä ei dokumentaation perusteella ole vaikeaa, mutta toiminta on syytä testata useilla erilaisilla laiteilla, kuten verkon aktiivilaitteilla ja mobiililaitteilla. SCEP:n tietoturva tulee myös suunnitella tarkasti, koska vakiona voidaan käyttää esimerkiksi yhtä ja samaa jaettua avainta kaikille. Yhden avaimen käyttö lisää menetelmän tietoturvariskiä. SCEP rajattiin pois tästä opinnäytetyöstä ajanpuutteen vuoksi.

OpenCA tukee todennettuja sertifikaattipyyntöjä, jossa sertifikaattipyynnön tietolähteenä toimii ulkoinen LDAP-palvelin. Pyyntöä tehdessä kirjaudutaan esimerkiksi Organisaatio X:n LDAP-palvelimelle käyttäjätunnus- ja salasana -parilla. Tällöin sertifikaattipyynnön tietolähteenä toimii LDAP-palvelimelta haetut tiedot. Myönnetty sertifikaatti voidaan julkaista OpenCA:n toimesta määritetylle LDAP-palvelimelle, josta se voidaan edelleen replikoida Organisaatio X:n aktiivihakemistoon. AD-integraatio vaa-

tii paljon kustomointia, mutta uskon, että se voidaan toteuttaa kohtalaisen järkevästi OpenCA:han.

Järjestelmää voidaan jatkossa myös laajentaa lisäämällä uusia alivarmentajia ja lisäämällä kolmas kerros hierarkiaan. Esimerkiksi yksittäisille organisaatioille voidaan myöntää oma CA.

Varmentajien tietoturvaa on mahdollista parantaa jatkossa esimerkiksi ottamalla käyttöön HSM-moduuli, joka on erittäin turvallinen rautapohjainen ratkaisu avaimen säilytykseen. Uusimmat OpenCA versiot tukevat HSM-moduulien käyttöä. Varmentajien eri käyttöliittymiin voitaisiin ottaa käyttöön LDAP-kirjautuminen paikallisen käyttäjätunnus ja salasana -parin sijasta. Myös varmennekäytäntölausuman (CPS) ja varmennepolitiikan (CP) kirjoittaminen alusta loppuun on yksi mielenkiintoinen kehityskohde.

LÄHTEET

Adams, C., Lloyd, S. 2003. Understanding PKI, Concepts, Standards, and Deployment Considerations, Second Edition. Pearson Education, Inc.

Chokhani, S., Ford, W., Sabett, R., Merrill, C. & Wu, S. 2003. RFC 3647 Internet X.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework. Internet RFC Archives.

Choudhury, S. 2002. Public Key Infrastructure Implementation and Design. John Wiley & Sons.

Conklin, W. A., White, G., Williams, D., Davis, R. & Cothren, C. 2011. CompTIA Security+ Exam Guide, Third Edition. McGraw-Hill.

Diodati, M. 2012. Mobile Device Certificate Enrollment: Are You Vulnerable?. Viitattu 28.9.2012. Http://blogs.gartner.com/mark-diodati/2012/07/02/mobile-device-certificate-enrollment-are-you-vulnerable/.

EJBCA. 2012. Ohjelmiston verkkosivut. Viitattu 2.8.2012. EJBCA team. Http://www.ejbca.org/.

Ghori, A., I. & Parveen, A. 2006. PKI ADMINISTRATION USING EJBCA AND OPENCA. George Mason University. Viitattu 3.10.2012. Http://teal.gmu.edu/courses/ECE646/project/reports_2006/IL-3-report.pdf.

Harris, S. 2010. CISSP All-in-One Exam Guide, Fifth Edition. McGraw-Hill/Osborne.

Jyvsectec. 2012. JYVSECTEC-hankkeen verkkosivut. Viitattu 12.7.2012. Http://www.jyvsectec.fi.

Komar, B. 2008. Windows Server 2008 PKI and Certificate Security. Microsoft Press.

Liu, X., Madson, C., Nourse, A. & Vilhuber, J. 2009. RFC draft-nourse-scep-18 Cisco Systems' Simple Certificate Enrollment Protocol. Internet RFC Archives.

OpenCA Group. 2005. OpenCA Guide for Versions 0.9.2+. OpenCA PKI Research Labs. Viitattu 6.8.2012. Http://www.openca.org/projects/openca/docs/openca-guide.pdf.

Shapiro, J. R. 2008. Windows Server 2008 Bible. John Wiley & Sons.

Tietotekniikan koulutusohjelma. 2012b. Jyväskylän ammattikorkeakoulun verkkosivut. Viitattu 12.7.2012. Http://www.jamk.fi, Koulutus, AMK-tutkinnot, nuorten koulutus, Tekniikan ja liikenteen ala, Tietotekniikan koulutusohjelma.

TinyCA. n.d. Ohjelmiston verkkosivut. Viitattu 1.8.2012. Http://tinyca.sm-zone.net/.

Tutustu JAMKiin. 2012a. Jyväskylän ammattikorkeakoulun verkkosivut. Viitattu 12.7.2012. Http://www.jamk.fi, Tutustu JAMKiin.

Vacca, J. R. 2009. Computer and Information Security Handbook. Morgan Kaufmann Publishers.

Vatra, N. 2011. A PKI Architecture Using Open Source Software for E-Government Services in Romania. Indian Journal of Computer Science and Engineering. Viitattu 2.8.2012. Http://www.ijcse.com/docs/INDJCSE11-02-04-177.pdf.

Xenitellis, S. 2000. The Open–source PKI Book. Viitattu 24.7.2012. Http://ospkibook.sourceforge.net/.

LIITTEET

Liite 1; DBI.pm päivitys

```
3315a3316
>     if ( $q_value =~ /^\d+$/ ) {
3317,3318d3317
<       } elsif ( $q_type =~ /BIGINT/ ) {
<         $self->{STH}->bind_param( $q_count, $q_value, SQL_BIGINT );
3320c3319,3326
<         $self->{STH}->bind_param( $q_count, $q_value, SQL_UNKNOWN_TYPE );
---
>       $self->debug ("doQuery: Query Type: $q_type but HEXADECIMAL detected.");
>       $self->{STH}->bind_param( $q_count, $q_value );
>       }
>   } elsif ( $q_type =~ /BIGINT/ ) {
>     $self->{STH}->bind_param( $q_count, $q_value, SQL_BIGINT );
>   } else {
>     $self->{STH}->bind_param( $q_count, $q_value, SQL_UNKNOWN_TYPE );
>   #    $self->{STH}->bind_param( $q_count, $q_value );
```

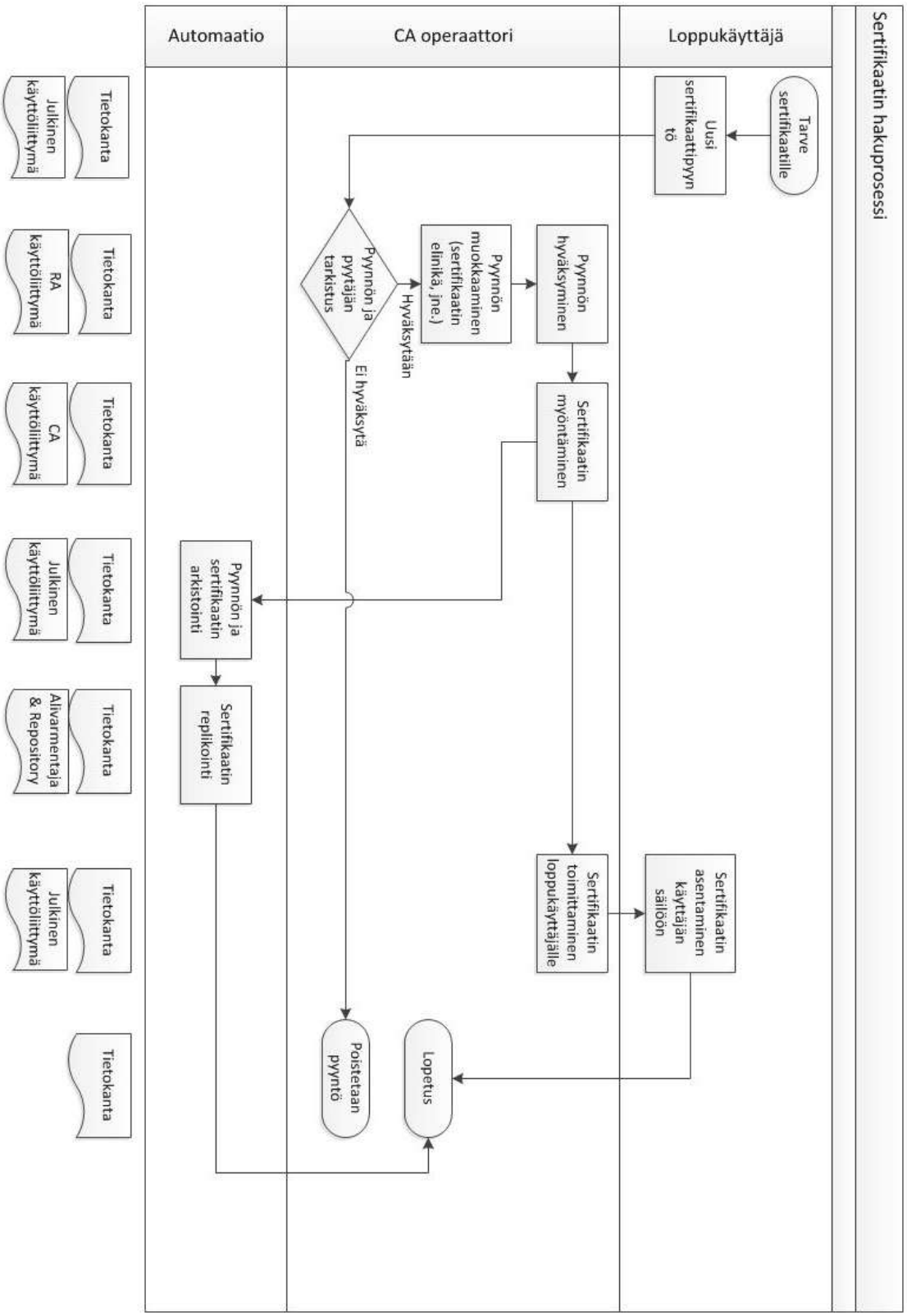
Liite 2; Repository, pub-menu.xml.template

```
<?xml version="1.0" encoding="UTF-8"?>

<openca base="/pki/pub">

 <menu name="PKI Info" img="images/info-key.png">

  <item name="Home" img="" lnk="?cmd=getStaticPage;name=homePage" />

  <item />

  <item name="Get CA Certificate" img=""
lnk="?cmd=getStaticPage;name=download_cacert" />

  <item name="Get Current CRL" img=""
lnk="?cmd=getStaticPage;name=download_crl" />

  <item name="Get CA Policy" img="" lnk="/pki/pub/?" />

  <item />

  <submenu name="Go To..." img="">

   <item name="OpenCA Labs" img="" lnk="http://www.openca.org/?" />

   <item name="OpenCA PKI Project" img=""
lnk="http://www.openca.org/projects/openca?" />

  </submenu>

  <item />

  <item name="About OpenCA..." img="" lnk="?cmd=serverInfo" />

 </menu>

 <menu name="My Certificates" img="">

  <item name="Install My Certificate" img=""
lnk="?cmd=getParams;GET_PARAMS_CMD=getcert" />

  <item />

  </menu>

  <menu name="Information" img="">

<!--    <submenu name="Issued Certificates" img=""> -->

     <item name="Valid Certificates" img="" lnk="?cmd=lists;action=certslist" />

     <item name="Expired Certificates" img=""
lnk="?cmd=lists;action=certsExpiredList" />
```

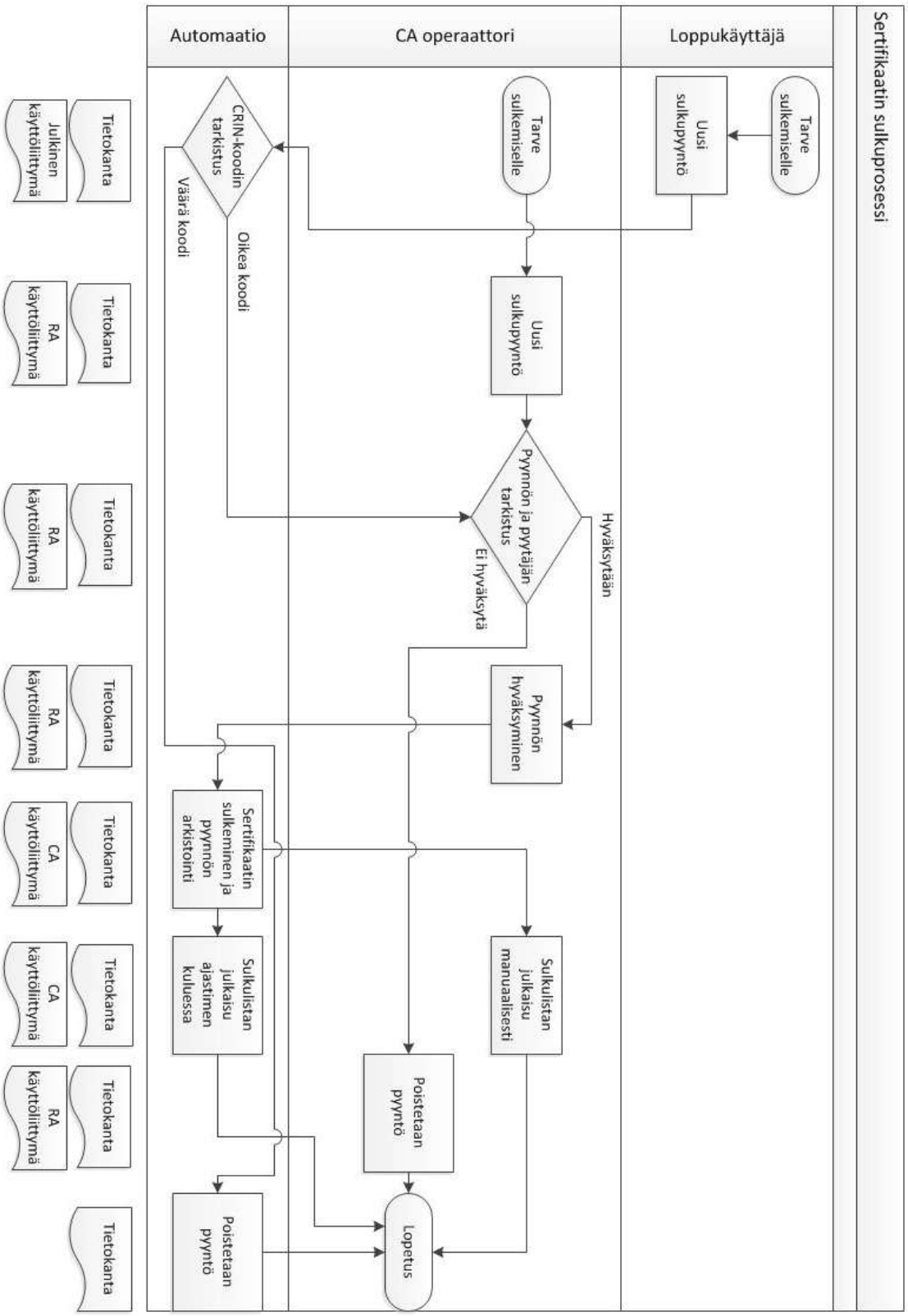
```
      <item />
      <item name="Suspended Certificates" img=""
lnk="?cmd=lists;action=suspendedlist"/>
      <item name="Revoked Certificates" img="" lnk="?cmd=lists;action=revokedlist"
/>
      <item />
     <item name="Current CRL" img="" lnk="?cmd=crlList;dataType=VALID_CRL" />
     <item name="Expired CRLs" img="" lnk="?cmd=crlList;dataType=EXPIRED_CRL" />
   <item />
     <item name="Search Certificates" img=""
lnk="?cmd=getStaticPage;name=search_cert" />
  </menu>
  <menu name="Help" img="">
   <item name="OpenCA PKI Guide" img=""
lnk="http://www.openca.org/projects/openca/docs.shtml?" />
   <item name="OpenCA PKI Home" img=""
lnk="http://www.openca.org/projects/openca?" />
   <item />
   <item name="About OpenCA..." img="" lnk="?cmd=serverInfo" />
  </menu>
  <menu name="Languages" img="" >
   <item name="English" img=""
                              lnk="?cmd=setLanguage;lang=en_GB;charset=UTF-8" />
   <item name="Spanish" img=""
                              lnk="?cmd=setLanguage;lang=es_ES;charset=UTF-8" />
   <item name="Italian" img=""
                              lnk="?cmd=setLanguage;lang=it_IT;charset=UTF-8" />
   <item name="French" img=""
                              lnk="?cmd=setLanguage;lang=fr_FR;charset=UTF-8" />
   <item name="Japanese" img=""
                              lnk="?cmd=setLanguage;lang=ja_JP;charset=UTF-8" />
```

```
  <item name="Greek" img=""
                              lnk="?cmd=setLanguage;lang=el_GR;charset=UTF-8" />
  <item name="Polish" img=""
                              lnk="?cmd=setLanguage;lang=pl_PL;charset=UTF-8" />
  <item name="Slovene" img=""
                              lnk="?cmd=setLanguage;lang=sl_SI;charset=UTF-8" />
  <item name="Russian" img=""
                              lnk="?cmd=setLanguage;lang=ru_RU;charset=UTF-8" />
  <item name="German" img=""
                              lnk="?cmd=setLanguage;lang=de_DE;charset=UTF-8" />
 </menu>
 <item name="Home" img="images/home.png"
                              lnk="?cmd=getStaticPage;name=homePage" />
</openca>
```

Liite 3; Sertifikaatin hakuprosessi



Liite 4; Sertifikaatin sulkuprosessi



Liite 5; OpenCA käynnistysskripti

```
#!/bin/sh
#
# chkconfig: 345 75 55
# description: OpenCA Server

cd /opt/openca/etc/openca  || exit 1
case "$1" in
   start)
      echo -n "Starting OpenCA ... "
      ./openca_start
      echo OK
   ;;
   stop)
      echo "Shutting down OpenCA ... "
      ./openca_stop
   ;;
   restart)
      $0 stop
      $0 start
   ;;
   *)
   echo "Usage: $0 {start|stop|restart}"
   exit 1
esac
```

Liite 6; SSL asetukset Apachen sites-enabled tiedostoon

```
<VirtualHost *:80>
        RewriteEngine on
        RewriteCond %{HTTPS} !=on
        RewriteRule ^/?pki/(.*) https://%{SERVER_NAME}/pki/$1 [R,L]
        RewriteRule ^/?cgi-bin/(.*) https://%{SERVER_NAME}/cgi-bin/$1
[R,L]
<VirtualHost *:443>
DocumentRoot /var/www/
SSLEngine on
SSLCertificateFile /etc/apache2/apache.pem
SSLCertificateKeyFile /etc/apache2/keys/apache.key
#SSLCertificateChainFile /etc/apache2/ca.crt
        <Directory /var/www/cgi-bin/>
                AllowOverride None
                Options ExecCGI -MultiViews +SymLinksIfOwnerMatch
                SetHandler cgi-script
                Order allow,deny
                Allow from all
        </Directory>
</VirtualHost>
```

Liite 7; Repository:n tiedonvaihtoasetukset ”config.xml”-tiedostoon

```
<!--  3. the node acts as public/scep only  -->
  <option>
   <name>enroll_ca_certificate_states</name>
   <value></value>
  </option>
  <option>
   <name>enroll_certificate_states</name>
   <value></value>
  </option>
  <option>
   <name>enroll_crl_states</name>
   <value></value>
  </option>
  <option>
   <name>enroll_crr_states</name>
   <value></value>
  </option>
  <option>
   <name>enroll_csr_states</name>
   <value></value>
  </option>
  <option>
   <name>enroll_mail_states</name>
   <value></value>
  </option>
  <option>
   <name>receive_crr_states</name>
```

```
 <value></value>
</option>
<option>
 <name>receive_csr_states</name>
 <value></value>
</option>
<option>
 <name>download_ca_certificate_states</name>
 <value>VALID</value>
</option>
<option>
 <name>download_certificate_states</name>
 <value>VALID</value>
</option>
<option>
 <name>download_crl_states</name>
 <value>VALID</value>
</option>
<option>
 <name>download_crr_states</name>
 <value></value>
</option>
<option>
 <name>download_csr_states</name>
 <value></value>
</option>
<option>
 <name>download_mail_states</name>
 <value></value>
```

```
</option>
<option>
 <name>upload_crr_states</name>
 <value></value>
</option>
<option>
 <name>upload_csr_states</name>
 <value></value>
</option>
```