TAMPEREEN TEKNILLINEN KORKEAKOULU
Tietotekniikan osasto

ANTTI VÄHÄ-SIPILÄ

SALAUSAVAINTEN JA LUOTTAMUKSEN HALLINNASTA AVOIMISSA TIETOJÄRJESTELMISSÄ

Diplomityö

Aihe hyväksytty osastoneuvoston kokouksessa
9.9.1998
Tarkastaja: Prof. Antti Valmari (TTKK)

Alkulause. Olen tehnyt diplomityöni Nokia Matkapuhelinten Wireless Data -yksikössä Tampereella 1998-1999. Esitän parhaimmat kiitokseni työn tarkastajalle, professori Antti Valmarille, sekä työn ohjaajana Nokia Matkapuhelimilla toimineelle Jari Mäenpäälle. Merkittäviä kommentteja tai korjausehdotuksia tarjosivat myös Jyke Jokinen (TTKK) ja Valtteri Niemi (Nokia Research Center) sekä vanhempani Anneli ja Markku Vähä-Sipilä — kiitokset myös kaikille heille.

Tampereella, 18. helmikuuta 1999

Antti Vähä-Sipilä
Vaajakatu 5 F 125
33720 Tampere

<avs@iki.fi>
<URL:http://www.iki.fi/avs/>

TAMPEREEN TEKNILLINEN KORKEAKOULU
Tietotekniikan osasto
Ohjelmistotekniikka

VÄHÄ-SIPILÄ, ANTTI: Salausavainten ja luottamuksen hallinnasta avoimissa tietojärjestelmissä
Diplomityö, 79 s.
Helmikuu 1999

Tarkastaja: Prof. Antti Valmari
Rahoittaja: Nokia Mobile Phones, Wireless Data
Avainsanat: kryptografia, avaintenhallinta, luottamuksenhallinta,
cryptography, key management, trust management

Tiivistelmä. Tietoverkkojen ja avointen tietojärjestelmien yleistyessä tiedon salaus tulee yhä tärkeämmäksi. *Kryptografisia* menetelmiä käytetään salauksen lisäksi alkuperätodennukseen sekä tiedon eheyden varmentamiseen. Todennuksessa keskeinen ongelma on salaus- ja allekirjoitusavainten alkuperän varmennus. Tätä kutsutaan *avaintenhallinta*ongelmaksi. *Luottamuksenhallinnalla* taas tutkitaan, voiko järjestelmä antaa jonkin tahon tehdä tiettyjä asioita.

Diplomityössäni tutkin olemassaolevia ja ehdotettuja menetelmiä avainten- ja luottamuksenhallintaan. Ensimmäisessä osassa vertailen erilaisia varmennetyyppejä sekä julkisten avainten infrastruktuureja keskenään. Käsittelen myös avainten käytöstä poistamista. Toisessa työn osassa keskityn luottamuksenhallintajärjestelmiin ja selvitän automatisoidun luottamuksenhallinnan perusperiaatteet. Työn lopussa luon silmäyksen tulevaisuuteen.

Erityisesti luottamuksenhallintaongelma on suhteellisen nuori tutkimuksenhaara: useimmat alan julkaisut on kirjoitettu vuonna 1996 tai sen jälkeen ja niissä esitetyt ajatukset ovat vasta tulemassa käyttöön todellisissa järjestelmissä. Minulla oli kaksi pääsyytä aiheen valintaan: ensinnäkin halusin tutkia tällä hetkellä käytettyjen avainten- ja luottamuksenhallintajärjestelmien mahdollisia puutteita ja toiseksi osoittaa, että kryptografiassa on kyse muustakin kuin mukavasta salausalgoritmista. Kryptografisten menetelmien sovellusympäristö ja lainsäädännölliset näkökohdat ovat jopa tärkeämpiä kuin itse algoritmit.

TAMPERE UNIVERSITY OF TECHNOLOGY
Department of Information Technology
Software Systems Laboratory

VÄHÄ-SIPILÄ, ANTTI: On encryption key and trust management in open information systems
M.Sc. thesis, 79 pp.
February 1999

Examiner: Prof. Antti Valmari
Sponsor: Nokia Mobile Phones, Wireless Data
Keywords: kryptografia, avaintenhallinta, luottamuksenhallinta,
cryptography, key management, trust management

Abstract. As computer networks and open information systems are becoming more popular, the encryption (ciphering) of data is becoming increasingly important. *Cryptographic* methods are utilized, in addition to encryption, for authentication of origin and data integrity. *Key management*, used to certify the origin of encryption and signature keys, plays a central role in authentication. Whether a system can trust an entity to let it carry out certain actions is a *trust management* problem.

In my thesis, I consider existing and proposed methods for key and trust management. In the first part, I compare different certificate types and public key infrastructures with each other, including the special issue of key revocation. The second part of the thesis concentrates on trust management systems, explaining the basic principles of automated trust management. The thesis concludes with a short prediction of what the future might bring.

Especially the trust management problem is a relatively new area of research. Most of the publications in this field have been published during and after 1996, and the ideas presented are making their way into real designs little by little. My reasons for choosing this subject for my thesis are twofold: first, I wanted to study the possible shortcomings of currently deployed key and trust management systems and second, I wanted to show that cryptography is a lot more than a nice cipher. The framework and legal setting in which cryptography is implemented is even more important than the algorithms themselves.

Sisältö

Kuvat

1 Johdanto

Työn tarkoituksena on perehdyttää lukija salausavainten hallintaan, jolla avainten oikea alkuperä voidaan varmentaa, sekä luottamuksenhallintaan, jolla tietyn avaimen haltijan oikeus tiettyyn palveluun voidaan todeta. Aihe on varsin mielenkiintoinen, koska alalta ei juuri ole ollut julkaisuja ennen vuotta 1996, eikä niistä ole tehty suomeksi yhtään yhteenvetoa — englanniksikaan niitä ei tunnu helposti löytyvän. Ala kulkee nopeasti eteenpäin ja työn ajatukset perustuvat vuoden 1998 lopussa vallinneeseen tilanteeseen, vaikka teoria sinänsä tuskin vanhenee.

Tavoitteenani oli myös näyttää, että kun salattua tiedonsiirtoa halutaan tuoda kiinteäksi osaksi avoimia tietoverkkoja ja turvalliset yhteydet pitäisi saada kaikkien käyttäjien ulottuville, on ajattelua laajennettava pelkkien algoritmien ja protokollien ulkopuolelle. Avaintenhallinta on todennäköisesti käyttäjälle näkyvin ja hankalimmin miellettävä osa nykyisiä turvallisen tiedonsiirron sovelluksia ja alan ymmärtäminen on tärkeää luotettavien ja käyttäjäystävällisten sovellusten aikaansaamiseksi.

Työn toisessa luvussa käydään lyhyesti läpi kryptografisia menetelmiä. Siinä esitellään symmetrisen salauksen piirteet sekä muita alaan liittyvä käsitteitä kuten MACit ja tiivistefunktiot ja tutustutaan symmetristen salausmenetelmien avaintenhallinnan ongelmiin. Samassa yhteydessä esitetään myös asymmetristen (julkisen avaimen) salausmenetelmien periaate, digitaalisen allekirjoituksen idea sekä käydään esimerkinomaisesti läpi Diffie-Hellman -avaintenvaihtomenetelmä.

Julkisen avaimen salausmenetelmien myötä kehittyivät avaintenhallintajärjestelmät, jotka eivät vaadi osapuolten välillä turvallista kanavaa avaintenvaihdon aikana. Erilaisiin avaintenhallintainfrastruktuureihin perehdytään luvussa 3. Siinä esitellään varmenteen käsite ja käydään läpi hierarkkinen avaintenhallintainfrastruktuuri. Nykyään yleisimmin käytössä olevat X.509-varmenteet perustuvat hierarkkiselle ajattelulle, mutta muunlaiset avaintenhallintamallit tekevät tuloaan joustavuutensa vuoksi. ICE-TELin hybridimallin kautta siirrytään PGP:n tapaiseen luottamusverkkoon.

Luvun 3 loppupuolella tarkastellaan toista näkemyksellistä eroa, joka vallitsee henkilö- ja attribuuttivarmenteiden välillä ja tutustutaan lähemmin SPKI/SDSI-aloitteeseen. Lisäksi luodaan lyhyt silmäys varmenteiden käytöstä poistamiseen liittyvään problematiikkaan sekä luotettujen kolmansien osapuolien käyttöön.

Työn toinen suuri kokonaisuus käsittelee luottamuksenhallintaa. Luvussa 4 esitellään luottamuksenhallinnan perusajatus ja kuvaillaan, mitä yleistetty luottamuksenhallinta tarkoittaa. Luvussa käydään läpi kaksi yleiskäyttöistä automaattista luottamuksenhallintajärjestelmää, PolicyMaker sekä KeyNote. Erilaisiin metatiedon kuvauksiin liittyvät järjestelmät kuten REFEREE ja PICSRules saavat myös hieman huomiota osakseen. Luvun lopussa puhutaan hieman yhä tärkeämmäksi nousevasta XML:stä ja sen suhteesta luottamuksenhallintaan.

Työn lopuksi maalaillaan tulevaisuudenkuvia autonomisista Jini-agenteista ja verkotetuista kahvinkeittimistä luvussa 5.

Olen pyrkinyt löytämään englanninkielisille termeille suomennoksen aina, kun se on mahdollista. Tämän vuoksi työssä esiintyy runsaasti aiemmin käyttämättömiä suomenkielisiä termejä. Jokaisen suomennoksen englanninkielinen vastine on kuitenkin aina näkyvissä.

Internet-viitteistä. Aiheen uutuuden ja Internet-keskeisyyden vuoksi viiteluettelo sisältää useita dokumentteja, jotka on julkaistu ainoastaan Internetissä. Jotkut dokumenteista ovat myös luonteeltaan väliaikaisia. Kyseessä ovat tällöin yleensä työn alla olevat määrittely- ja tekniset dokumentit.

Jos dokumentti löytyy ainoastaan Internetistä eikä se kuulu tunnettuun sarjaan, dokumentin osoitteen vieressä on mainittu kaksi päivämäärää muodossa vuosi-kuukausi-päivä. Ensimmäinen päivämäärä ilmoittaa dokumentin edellisen muutosajan palvelimella (joka ei välttämättä tarkoita sisällön muuttumista) ja jälkimmäinen ilmoittaa päivämäärän, jolloin lähdemateriaalina käytetty dokumentti on palvelimelta haettu.

Mikäli dokumentti on merkitty *Internet-Draft (Work in progress)*, kyseinen viite on työn alla oleva dokumentti ja se vanhenee viimeistään kuuden kuukauden kuluttua sen julkaisemisesta. Uusia versioita kyseisistä viitteistä voi etsiä joko uudemmilla versionumeroilla Internet-Draft -tietokannasta tai RFC (Request for Comments) -dokumenttitietokannasta, jonne useat Internet-Draftit päätyvät sen jälkeen, kun niihin ei enää tehdä muutoksia. Vastaavasti World Wide Web Consortiumin työdokumenteista (*W3C Working Draft*) tulee niiden kypsyttyä suosituksia (*W3C Recommendation*).

Internet-Drafteista pitää kirjaa IETF (Internet Engineering Task Force), `http://www.ietf.org/`. RFC-dokumenteista huolehtii IESG (Internet Engineering Steering Group), `http://www.iesg.org/`. World Wide Web Consortium sijaitsee osoitteessa `http://www.w3.org/`. Osa lähdemateriaalista

on arkistoitu sekä elektronisesti että paperille ja se on saatavissa diplomityön kirjoittajalta <avs@iki.fi>, mikäli dokumentin alkuperäinen kirjoittaja ei rajoita sen levitystä. Myös itse diplomityö on saatavissa elektronisessa muodossa suoraan kirjoittajalta.

2 Salaus ja todennus

Kun kaksi tai useampi osapuolta viestii tiedonsiirtokanavan yli, siirretyn informaation luonteesta riippuen siirtokanavalle asetetaan erilaisia vaatimuksia. Turvallisuusvaatimuksiin kuuluu [Sch96, kappale 1.1] siirtotien *eheys* (integrity), joka kattaa sen, ettei informaatio saa hukkua eikä muuttua siirron aikana. *Luottamuksellisuus*vaatimuksella (confidentiality, secrecy) pyritään estämään informaation joutuminen ulkopuolisen käsiin. Tiedon alkuperä taataan vaatimalla *todennus*mahdollisuus (authentication). Joskus myös *kiistämättömyys* (nonrepudiation) on tarpeen: lähettäjä ei voi myöhemmin kiistää lähettäneensä viestiä.

Turvallisuusvaatimuksista huolehtivien turvallisuuspalveluiden toteutuksen rakennuspalikoina toimivat salaus- ja todennus*algoritmit* ja algoritmeja hyödyntävät *protokollat*. Protokolla koostuu siirrettävän tiedon rakennekuvauksesta ja kommunikoivien osapuolien tilakoneista.

Turvallisuuspalvelut voivat olla olemassa toisistaan riippumatta. On täysin mahdollista rakentaa todennusprotokolla ilman salausta tai salausprotokolla ilman todennusta.

Tietojärjestelmä voi tarjota jollekin taholle (käyttäjälle tai esimerkiksi toiselle tietokoneelle) *palveluita*. Luvattoman käytön estämiseksi käytetään *pääsynvalvontaa* (access control). Yksinkertaisimmillaan kyseessä on *pääsylista* (ACL, Access Control List), joka sisältää palvelua käyttävän tahon tunnisteen. Taho todentaa itsensä (todistaa identiteettinsä) pääsynvalvontaa suorittavalle taholle esimerkiksi haaste-vaste -protokollalla (challenge-response protocol). ACL:ään on tällöin kiinnitetty jokaiselle käyttöoikeuden haltijalle oma salaisuus. Haasteella pyritään selvittämään, tunteeko pääsyä haluava taho ko. salaisuuden.

Hyökkäykset. Taho, joka yrittää saada hänelle kuulumatonta tietoa tiedonsiirtokanavalta tai päästä käyttämään hänelle kuulumatonta palvelua, on *hyökkääjä* (attacker, adversary).[1] Vaikka hyökkääjä usein ajatellaan "pahana", tässä yhteydessä määritelmä on puhtaasti protokollateoreettinen — hyökkääjän pahuus lakien tai yleisen moraalikäsityksen kannalta on merkityksetöntä.

[1] Perinteisesti hyökkääjää kuvataan nimellä Eve tai E-kirjaimella. Tiedonsiirron lailliset osapuolet ovat Alice, Bob, Carol ja David, $A \ldots D$. Tämä merkintätapa on käytössä myös tässä työssä, ellei muuta ole mainittu.

Hyökkäys, jossa hyökkääjä keskittyy vain tarkkailemaan tiedonsiirtokanavaa ja siinä kulkevan liikenteen sisältöä ja liikenneprofiilia mielivaltaisina ajan hetkinä, on *passiivinen* (passive attack). Protokolla, joka vastustaa vain passiivisia hyökkäyksiä, on riittävä jos tiedonsiirtokanavan eheys- ja todennusvaatimukset muuten täyttyvät. Yleensä näin ei ole, jolloin protokollan on huolehdittava itse myös näistä turvallisuuspalveluista.

Mikäli hyökkääjä muuttaa siirrettävää tietoa tai lisää tai poistaa tietovirrasta haluamiaan osia, on kyseessä *aktiivinen hyökkäys* (active attack). Avoimissa järjestelmissä turvallisuuspalveluiden oletetaan yleensä tarjoavan turvallisen kommunikaatiotien sellaista kanavaa pitkin, joka ei täytä mitään turvallisuusvaatimuksia. Tällöin turvaprotokollan ja -algoritmien vaatimuksena on vastustuskykyisyys myös aktiivisia hyökkäyksiä vastaan.

Turvallisuusanalyyseissä tulee myös aina olettaa, että käytetty protokolla ja algoritmit ovat hyökkääjän tiedossa.

Tiedon eheys. Lähetettävään informaatioon lisätään usein ylimääräistä (redundanttia) informaatiota, jonka avulla vastaanottaja voi tunnistaa matkalla virheelliseksi muuttuneet osat ja mahdollisesti korjata siirtovirheet. Tähän käytettäviä tapoja ovat *tarkistussummat* (checksums), *sykliset tarkisteet* (CRC, cyclic redundancy check) sekä *virheenkorjaavat koodit* (error-correcting codes).[2] Virheenkorjauksen matematiikka on suurelta osin tämän työn ulkopuolella. Kiinnostuneille suositeltavia tietolähteitä ovat esimerkiksi [Ruo93] ja [Pre96]. [Sta94, ss. 143–149] sisältää CRC-koodien esittelyn.

Protokollien ajatellaan toimivan yleensä *protokollapinossa* (protocol stack). Pinon alemmat *kerrokset* (layer) tarjoavat tiedonsiirtokanavan ylemmille kerroksille. Ylimpänä kerroksena on usein jokin käyttäjälle näkyvä sovellus. Jokin kerros saattaa toteuttaa turvallisuuspalvelun, jolloin sen yläpuolella olevat kerrokset hyötyvät siitä.

Yleensä virheenkorjauksen oletetaan toimivan siirtotiellä turvallisuuspalveluista riippumatta. Jos virheellinen viesti pääsee nousemaan protokollapinossa vastaanottajan turvallisuuskerrokselle asti, näyttää se eheydestä huolehtivan osan kannalta viestiltä, johon hyökkääjä on yrittänyt tehdä muutoksia. Tällöin viesti yleensä hylätään (jolloin vaikutus on turvallisuuskerroksen

[2] Työssä käytetään sanaa "koodi" tarkoittamaan viestille tehtyä muunnosta, jonka käänteismuunnos on nopeasti toteutettavissa ilman viestin ulkopuolista tietoa (ts. ilman avainta) kun dekoodausalgoritmi on tunnettu. "Koodaus" ei siis tarkoita tiedon salausta, vaikka historiallisten salausmenetelmien yhteydessä [Kah96] puhuttiinkin koodeista. Tällöin salainen avain oli (de)koodausalgoritmi itse.

yläpuolella sama kuin jos viesti olisi hukkunut matkalla). Toki turmeltuneen viestin vastaanotto voisi johtaa myös esimerkiksi yhteyden katkaisuun ja hälytyskellojen soittoon, mutta se altistaisi järjestelmän *palvelunriisto-hyökkäykselle* (denial of service attack), jossa hyökkääjän päätarkoituksena onkin saada yhteydet jumiutettua ja järjestelmän ylläpito ylityöllistettyä.

2.1 Symmetrinen kryptografia

Salaus eli *kryptaus* (encryption, ciphering) on tapahtuma, jossa *selväkielinen* (plaintext) viesti (joskus myös *selkotekstiviesti*) M annetaan yhdessä *avaimen* (key) kanssa syötteenä salausfunktiolle E. Tuloksena on *kryptoteksti* (cryptotext, ciphertext) C:

$$C = E(M, K)$$

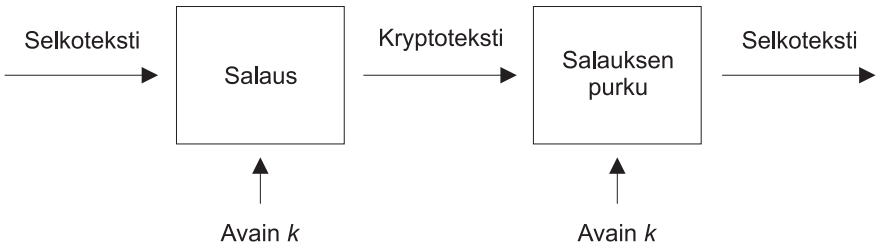
E:n käänteisfunktio D *dekryptaa* (decryption, deciphering) eli *purkaa* (joskus myös *tulkitsee*) salauksen. *Symmetrisessä salauksessa* (symmetric encryption) siihen käytetään samaa avainta K:

$$M = D(C, K)$$

Useissa käytössä olevissa salausmenetelmissä pätee $D = E$, eli kryptaus- ja dekryptausfunktio ovat täysin samat. Luonnollisesti pätee

$$M = D(E(M, K), K).$$

Tapahtuma on esitetty kuvassa 1.



Kuva 1: Symmetrinen (salaisen avaimen) salaus

Hyvien salaus- ja purkufunktioiden eli *kryptosysteemien* (cryptosystem) ominaisuuksiin kuuluu nk. *Kerckhoffsin oletus* [Sch96, kappale 1.1], [Kah96, ss.

233–236], jonka mukaan kaiken turvallisuuden on oltava avaimessa ja salausfunktion ominaisuuksissa[3]. E ja D ovat siis kaikkien yleisessä tiedossa.

Symmetrinen salaus ei riitä takaamaan siirrettävän tiedon eheyttä, sillä tietovirtaan voi sen estämättä lisätä ja siitä voi poistaa osia. Jos kryptoteksti riippuu salausalgoritmin rakenteen vuoksi aiemmasta kryptotekstistä, poisto- tai lisäyshyökkäys johtaa yleensä kanavassa kulkevan tiedon muuttumiseen ainakin joksikin aikaa valesatunnaiseksi kohinaksi. Mikäli kyseessä on *vuosalain* (stream cipher) tai ilman lohkojen ketjutusta käytettävä *lohkosalain* (block cipher), jossa kukin salattava selväkielinen osa (tyypillisesti vuosalaimilla yksi oktetti eli kahdeksan bittiä, lohkosalaimilla muutamasta oktetista kymmeniin oktetteihin) salataan ympäröivästä kryptotekstistä välittämättä, oikein ajoitetulla lisäys-, poisto- tai muuntohyökkäyksellä voidaan aiheuttaa laskelmoituja ja vaikeasti havaittavia muutoksia selväkieliseen tietoon.

MAC. Viestiin saadaan eheyden tarkistusmahdollisuus ajamalla selväkieli vaikkapa pieni osa kerrallaan yksisuuntaisen *tiivistefunktion*[4] (one-way hash) läpi. Tiivistefunktio täyttää *yksisuuntaisuusehdon*, mikäli annettaessa jokin tiivistefunktion arvojoukon alkio, on laskennallisesti ylivoimaisen työlästä löytää ainuttakaan siihen kuvautuvaa määrittelyjoukon alkiota. *Törmäysehto* täyttyy, jos on laskennallisesti ylivoimaista löytää kahta määrittelyjoukon alkiota, jotka kuvautuvat samaan arvojoukon alkioon. Yksisuuntaisuus- ja törmäysehdot täyttävä tiivistefunktio on *kryptografisesti vahva* tiivistefunktio.

Kun tiivistefunktion antama arvo riippuu myös salaisesta avaimesta K, voi vastapää viestin avaamisen jälkeen verrata saapunutta ja itse laskemaansa tiivistettä keskenään. Mikäli ne ovat samat, viesti on eheä. Tätä kutsutaan MACiksi (Message Authentication Code).

2.2 Symmetristen järjestelmien avaintenhallinta

Koska kaikki turvallisuus liittyy viime kädessä salausavaimeen, on *avaintenhallinta* (key management) oleellisen tärkeä järjestelmän turvallisuuden kannalta. Luonnollisesti ei ole mitään mieltä lähettää salausavainta turvattoman

[3] Itse asiassa tämä on 1800-luvulla vaikuttaneen Kerckhoffsin toinen oletus. Nykytermejä käyttäen muut olivat käytännön murtamattomuus, helppo uudelleenavainnus, soveltuvuus telekommunikaatioon, hyvä siirrettävyys ja helppokäyttöisyys [Kah96, ss. 233–236].

[4] Joskus myös *hajautusfunktio*. Termi "tiivistefunktio" johtuu siitä, että funktion lopputulos on tyypillisesti paljon sen syötettä pienempi.

kanavan yli ja sen jälkeen luottaa siihen, että hyökkääjä ei huomannut sitä, koska jo passiivisen hyökkäyksen määritelmän mukaan hyökkääjä voi tarkkailla kanavaa mielivaltaisena ajan hetkenä. Salausavaimen siirto tehdäänkin tästä syystä symmetrisissä menetelmissä käyttäen vaihtoehtoista tietä (out-of-band) — vaikkapa kuriiripostina paperille painettuna tai levykkeelle tallennettuna.

Kun suuri joukko henkilöitä kommunikoi keskenään, he voivat joko jakaa saman salausavaimen, tai jokaisella kommunikaatioparilla voi olla oma avain. Edellisessä tapauksessa ei viestin lähettäjää ryhmän sisällä voida todentaa, ja yhden avaimen vuotaessa ryhmän ulkopuolelle kaikkien avaimet on uusittava. Jälkimmäisessä tapauksessa päästään eroon näistä ongelmista, mutta hallittavien avainten määrä kasvaa suuruusluokkaan $\Theta(n^2)$. Kummassakin tapauksessa osapuolien sijaitessa vaikeapääsyisellä alueella (armeijan viestiyksiköt hajaantuneina pitkin aavikkoa tai avaruussukkulan henkilökunta matkalla avaruusasemalle) on avainten siirto esimerkiksi paperilla varsin hankala toteuttaa.

Ratkaisuja n^2-avaintenhallintaongelmaan. Ongelman ratkaisemiseksi on ehdotettu ja käytetty *luotettua kolmatta osapuolta* (TTP, Trusted Third Party).[5] Luotettu kolmas osapuoli on luotettu palvelin, johon muodostettu yhteys toteuttaa salaus-, eheys- ja todennettavuusvaatimukset.

Avaintenjakopalvelin [MVOV96, ss. 543–555] jakaa nimensä mukaisesti (mahdollisesti kertakäyttöisiä tai lyhytikäisiä) avaimia tarpeen mukaan.[6] Jos A ja B haluavat kommunikoida, A ottaa ennakkoon tuntemallaan avaimella K_{AT} yhteyden T:hen. T muodostaa avaimen K_{AB}, jonka se välittää sekä A:lle (käyttäen avainta K_{AT}) että B:lle (käyttäen avainta K_{BT}).

Avaimenmuuntopalvelin toimii vastaavasti, mutta avaimen K_{AB} luo A, joka lähettää sen T:lle K_{AT}:llä salattuna. T purkaa salauksen ja salaa avaimen uudelleen K_{BT}:llä ja välittää sen sitten B:lle (tai palauttaa A:lle välitettäväksi eteenpäin B:lle).

Ongelmia symmetrisiin menetelmiin perustuvassa avaintenhallinnassa. Käytännössä luotetun palvelimen T on oltava koko ajan saatavilla ja käyttökunnossa jokaiselle kommunikoivalle osapuolelle. Jos ajatellaan esimerkiksi Internetin sähköpostia, tulisi jonkin julkisen palvelimen (tai näiden

[5] Luotettu kolmas osapuoli on perinteisesti nimeltään Trent (T).

[6] Eräs tunnettu sovellus, jossa luotettu palvelin voi antaa asiakkaalle istuntoavaimen on MIT:n Kerberos [CDK95, ss. 495–502].

muodostaman luotetun verkoston) olla saatavissa jokaiselle miljoonista sähköpostin käyttäjistä. Tämä voi aiheuttaa ongelmia jo pienissä yritysverkoissa, jos esimerkiksi osa käyttäjistä on yhteydessä sisäverkkoon matkapuhelinyhteyttä käyttäen tai muutoin satunnaisesti.

Myös palvelimen luotettavuus ja turvallisuus on hyvin vaikea taata. T:llä on pääsy kaikkien osapuolten väliseen kommunikaatioon, ja kuten kaikissa keskitetyissä malleissa, luotettu palvelin on hyökkääjälle hyvin houkutteleva kohde. Koko järjestelmä on korkeintaan yhtä turvallinen kuin T.

2.3 Julkisen avaimen kryptografia

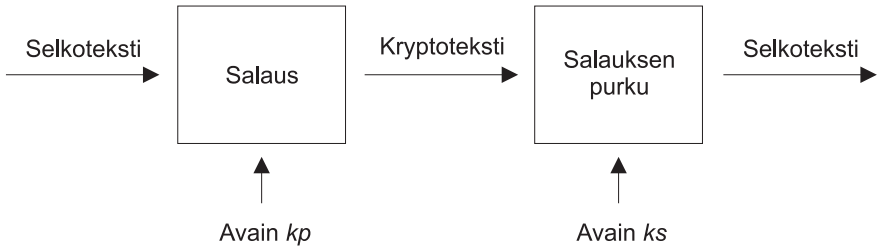
Whitfield Diffie ja Martin Hellman esittivät 1970-luvulla ratkaisun salausavainten vaihtoon turvattoman kanavan läpi käyttämättä ennakkoon sovittua salaista avainta[7]. Tämä *Diffie-Hellman -avaintenvaihtomenetelmäksi* nimetty protokolla, jossa kaikki osapuolten siirtämät viestit voivat olla julkisia, [DH76] avasi uuden aikakauden kryptografian alalla, jossa symmetrisiä menetelmiä (joskus myös "perinteisiksi" kutsuttuja) kaikkine avaintenhallintaongelmineen oli käytetty satoja, ellei tuhansia vuosia.

Julkisen avaimen kryptosysteemit (public key cryptosystems) perustuvat ongelmaan, joka on laskennallisesti erittäin raskas. Tällainen voi olla esimerkiksi luvun jako tekijöihin tai diskreetin logaritmin laskeminen. Perusajatuksena on se, että viestin laillinen vastaanottaja tietää ongelmasta jotakin ylimääräistä, joka tekee ratkaisun laskennallisesti mahdolliseksi. Tätä tietoa kutsutaan *salaoveksi* (trapdoor). Hyökkääjällä ei tätä informaatiota ole, joten murtaakseen järjestelmän hänen olisi ratkaistava ongelma, joka on hyvin hankala — käytännössä jopa mahdoton maailmankaikkeudessa käytettävissä olevien resurssien puitteissa.

On kuitenkin huomattava, että *todistetusti vahvoja* julkisen avaimen järjestelmiä ei ole. Mikäli edistysaskeleet vaikeiden ongelmien ratkaisemisessa vähentävät esimerkiksi tekijöihin jaon kompleksisuutta ratkaisevasti, kärsii usea kryptosysteemi.

[7] Englannin signaalitiedustelupalvelun alaisuudessa toimiva CESG oli itse asiassa tutkinut avaintenvaihtomenetelmiä jo 1960-luvulla, mutta tämä tieto pysyi luottamuksellisena aina vuoteen 1998 asti [Ell87]. Tästä voidaan päätellä jotakin epämiellyttävää muiden valtioiden turvallisuuspalveluiden tämänhetkisistä kryptologisista ja kryptoanalyyttisistä kyvyistä.

Avainparit. Julkisen avaimen järjestelmien salausavaimet jaetaan kahteen osaan, joista *julkista avainta* (public key) käytetään viestin salaamiseen ja *salaista avainta* (secret key) sen avaamiseen. Salainen avain on siis edellä mainitun salaoven avaamiseen tarvittava lisäinformaatio. Koska salaus- ja purkuvaiheessa käytetään eri avaimia, kutsutaan julkisen avaimen järjestelmiä myös *asymmetrisiksi kryptosysteemeiksi*. Kuva 2 selkiyttää tilannetta.



Kuva 2: Asymmetrinen (julkisen avaimen) salaus

Julkisen avaimen järjestelmien suunnittelutavoitteena on saada julkinen avain nimensä mukaisesti julkistamiskelpoiseksi eli sen perusteella ei pitäisi voida päätellä mitään salaisesta avaimesta. Julkisen avaimen levitys on tämän vuoksi hyvin helppoa: avaimen siirtoon käytettyä kanavaa ei tarvitse salata, koska hyökkääjälle ei julkisen avaimen haltuunsa saannista ole mitään hyötyä.

Kun tilannetta verrataan usean osapuolen väliseen kommunikaatioon symmetrisillä järjestelmillä, havaitaan, että julkisen avaimen menetelmillä voidaan saavuttaa kaikki symmetrisen avaintenhallinnan tavoitteet ilman hankalia yhteyksiä luotettuun kolmanteen osapuoleen.

Esimerkki: Diffie-Hellman-avaintenvaihtomenetelmä. Diffie-Hellman-menetelmä ei varsinaisesti ole salausmenetelmä vaan avaintenvaihtomenetelmä, eli sen avulla voidaan sopia yhteisestä salaisuudesta, jolla siirrettävä tieto voidaan myöhemmin salata. Lisäksi se on helppo ymmärtää, lyhyt kuvata ja muutenkin hyvä, joten se on suosittu esimerkki.

Potenssiin korotus modulo n on huomattavasti helpompaa kuin sen käänteisoperaatio, diskreetin logaritmin laskeminen. Diffie-Hellman-menetelmä [DH76], [Sti95, ss. 270–273] hyödyntää tätä seuraavasti:

A ja B valitsevat julkisesti luonnolliset luvut g ja n (nk. Diffie-Hellman-parametrit) siten, että n on iso alkuluku ja tyypillisesti pieni g on generaattori

mod n.[8] Menetelmän turvallisuus nojaa n:n suuruuteen.

A valitsee suuren satunnaisen luvun $x : x \in \mathbb{N}, 0 \leq x < n-1$ ja laskee $X = g^x \mod n$, ja lähettää X:n B:lle. B valitsee niin ikään suuren satunnaisen luvun $y : y \in \mathbb{N}, 0 \leq y < n-1$ ja lähettää A:lle Y:n, $Y = g^y \mod n$. X (tai Y) Diffie-Hellman-parametrien kera muodostaa A:n (tai vastaavasti B:n) "julkisen avaimen". x (tai y) muodostavat "salaisen avaimen".

A laskee $k = Y^x \mod n$ ja B laskee $k' = X^y \mod n$. Nyt $k = k' = g^{xy} \mod n$. k on A:n ja B:n yhteinen salaisuus, jota voidaan käyttää esimerkiksi istuntoavaimen (ks. jäljempänä) salaamiseen.

Jotta hyökkääjä saisi selvitettyä x:n tai y:n, tulisi hänen laskea diskreetti g-kantainen logaritmi X:stä tai Y:stä modulo n.

Yllä kuvattu nk. anonyymi Diffie-Hellman-menetelmä on kuitenkin haavoittuvainen *välityshyökkäykselle* (man-in-the-middle attack), jossa E asettuu protokollassa A:n ja B:n väliin ja teeskentelee A:lle olevansa B sekä päinvastoin. Ongelma ratkaistaan lisäämällä protokollaan tapa varmistua vastapään avaimen aitoudesta, josta myöhemmin.

Käytännön näkökohtia. Julkisen avaimen salausmenetelmien ongelmana on useissa sovellutuksissa se, että niiden laskutoimitukset ovat usein raskaita ja täten hankalia hyvin pienissä järjestelmissä ja toisaalta esimerkiksi voimakkaasti kuormitetuilla palvelimilla. Viime aikoina ovat suosiota saavuttaneet *elliptisten käyrien kryptosysteemit* (elliptic curve cryptosystems, ECC), jotka ovat esimerkiksi Diffie-Hellman-muunnoksia, joissa lasketaan nk. elliptisillä käyrillä kokonaislukukuntien sijaan. Järjestelmän sydämenä toimivan vaikean ongelman oletetaan olevan joissakin ryhmissä vaikeampi kuin toisissa, jolloin näissä voidaan käyttää pienempiä avaimia ja näin nopeuttaa laskentaa pienemmissä laskentaympäristöissä.

Käytännön sovellutuksissa julkisen avaimen menetelmää käytetään vain *istuntoavaimen* (session key) välitykseen. Siirrettävä tieto salataan julkisen avaimen menetelmään verrattuna nopealla symmetrisellä algoritmilla ja istuntoavain lähetetään tiedon mukana salattuna vastaanottajan julkisella avaimella. Tavallisesti salausprotokolla käyttää siis kolmea eri algoritmia: avaimenvaihto-, salaus- ja MAC-algoritmia. Tämä yleensä itse protokollasta riippumaton kolmikko on *algoritmiyhdistelmä* (cipher suite). Tyypillinen salausprotokolla sopii kulloisellakin yhteydelllä käytettävän algoritmiyhdistelmän kummankin osapuolen kykyjen mukaan.

[8] g on generaattori $\mod n$, jos kaikille $b : b \in 1, \ldots, n-1 : \exists a : g^a \equiv b \mod n$.

Digitaalinen allekirjoitus. Tarkastellaan salausta asymmetrisessä kryptosysteemissä. Merkitsemme julkista avainta K_P:llä ja salaista avainta K_S:llä. Nyt kryptaus ja dekryptaus ilmaistaan seuraavasti:

$$\begin{aligned} C &= E(M, K_P) \\ M &= D(C, K_S) \end{aligned}$$

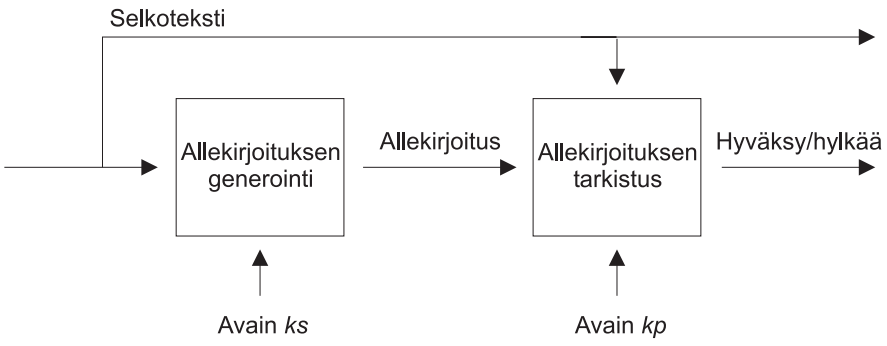
Julkisen avaimen järjestelmää voidaan käyttää myös *digitaalisen allekirjoituksen* (digital signature) luomiseen. Tällöin viesti (tai oikeammin sen tiiviste) ”salataan” salaisella avaimella ja kuka tahansa voi ”avata” sen julkisella avaimella. Koska salainen avain on vain lähettäjän tiedossa, julkisen avaimen haltija voi näin varmistua viestin alkuperästä ja oikeellisuudesta. Jos $H_M = H(M)$ on viestin M tiiviste, ja S_M on allekirjoitus, niin

$$S_M = E(H_M, K_S)$$

Vastaanottajalle lähetetään S_M ja M. Vastaanotettuja viestejä merkitään S'_M ja M'. Vastaanottaja tarkistaa allekirjoituksen K_S:ää vastaavalla julkisella avaimella K_P.

$$H'_M = D(S'_M, K_P)$$

Jos $H'_M = H(M')$, voidaan päätellä, että viesti oli allekirjoitettu vastaanottajan hallussa olevaa julkista avainta vastaavalla salaisella avaimella (kuva 3). Viestin alkuperän varmistuminen riippuu siitä, voiko vastaanottaja olla varma salaisen avaimen haltijasta.



Kuva 3: Digitaalinen allekirjoitus

Salaus ja digitaalinen allekirjoitus voidaan yhdistää. Viesti allekirjoitetaan ensin ja allekirjoitus viesteineen välitetään sen jälkeen salattuna. Tämä muodostaa salatun, eheän kanavan kahden osapuolen välille, joilla kummallakin on oma avainparinsa. Rakenne löytyy lähes kaikista turvaprotokollista.

Nimestään huolimatta digitaalinen allekirjoitus ei ole täysin analoginen tavallisen allekirjoituksen kanssa [EFL+98a], sillä digitaalinen allekirjoitus ei suoranaisesti sisällä tietoa allekirjoittaneesta tahosta. Digitaalisen allekirjoituksen tekijästä voidaan varmistua vasta kun allekirjoituksen tarkistamiseen sopiva julkinen avain on saatavissa ja kyseisen avaimen omistaja on selvillä. Käytännön toteutuksissa digitaalisten allekirjoitusten yhteydessä siirretäänkin myös allekirjoittajan julkisen avaimen omistajan tunnistetieto. Tavallisen allekirjoituksen tapauksessa allekirjoittajan nimi taas on ainakin teoriassa aina mukana allekirjoituksessa, joskin joskus nimen selvennys on tarpeen.

Digitaalinen allekirjoitus on hyvin voimakkaasti sidottu dokumentin informaatioon, kun taas tavallinen allekirjoitus on yleensä samanlainen dokumentista riippumatta ja se on helposti kopioitavissa dokumentista toiseen.

Joskus digitaalisen allekirjoituksen sijaan käytetään termiä *elektroninen allekirjoitus*. Näin tehdään varsinkin lainsäädännössä, kun halutaan kattaa allekirjoitus myös analogisen signaalinkäsittelyn menetelmin.

3 Avaintenhallinta

Yleisesti avaintenhallinnan ongelmakentässä käsitellään seuraavia asioita:

- Avainten *sopiminen* (key establishment):
  - Avainten *luonti* (key generation)
  - Avainten *välitys tarvitsijoille* (key transportation)
  - Avainten *levitys* kaikille tarvitsijoille (key distribution)
- Avainten alkuperän *varmennus* (key certification)
- Avainten *peruminen* (key revocation)
- Avainten *tuhoaminen*
- Avainten *varastointi*
- Avainten *luovutus kolmannelle osapuolelle* (key escrow, key recovery)

Koska avaimet ovat julkisia, niiden välitys tarvitsijoille voi tapahtua turvatonta kanavaa pitkin. Luonti, varastointi ja tuhoaminen, vaikkakin sisältävät useita huomionarvoisia seikkoja, jätetään tässä työssä vähälle huomiolle. [MVOV96] ja [Sch96] käsittelevät myös näitä osa-alueita.

Suljetussa järjestelmässä julkisten avaimien levitys on suhteellisen yksinkertaista: ylläpito asentaa tarvittavat avaimet jokaiselle viestin lähettäjälle ja vastaanottajalle. Avaimien alkuperään voi tällöin luottaa täysin.

Ongelma muuttuu hankalammaksi avoimessa tietojärjestelmässä, joille on tyypillistä, että käyttäjien määrä saattaa lisääntyä tai vähentyä ilman keskitettyä valvontaa. Lisäksi avoimista järjestelmistä voi lähteä ja sinne voi saapua viestejä järjestelmän ulkopuolelta.

Varmenne. Perusidea julkisten avainten alkuperän varmistamiseksi esitettiin ensi kerran 1978 [Koh78], jonkin verran Diffie-Hellman-järjestelmän keksimisen jälkeen. Konsepti on nimeltään *varmenne*, joskus myös *sertifikaatti* (certificate). Yksinkertaisimmillaan varmenteessa on julkinen avain, julkisen avaimen omistajan identifioivia tietoja ja digitaalinen allekirjoitus avaimesta ja tiedoista. Digitaalinen allekirjoitus on tehty jollakin toisella avaimella, jonka alkuperä varmasti tunnetaan.

Välittömästi nähdään, että ongelma ei kokonaan ratkea, koska allekirjoittavan avaimen alkuperän selvittäminen on edelleen auki. Allekirjoittavan avaimen julkinen osa voidaan kuitenkin myös toimittaa vastaanottajalle varmenteessa ja näin varmennusongelma saadaan jälleen sysättyä *varmenneketjussa* (certificate chain) pykälällä eteenpäin.

Se, miten varmennus järjestelmässä hoidetaan, riippuu käytetystä *julkisten avainten infrastruktuurista* (public key infrastructure). Eri infrastruktuurien ja niiden hyvien ja huonojen puolien ymmärtäminen on oleellista luottamushallinnan käsitteiden ymmärtämiseksi.

Taho, joka tutkii allekirjoitusten oikeellisuuden ja arvioi luottamuksen määrän, on nimeltään *todentaja* (verifier).

3.1 Hierarkkinen avaininfrastruktuuri

Keskusjohtoinen varmennus. Kuka tahansa armeijan järjestelmän tunteva henkilö tuntee olonsa kodikkaaksi hierarkkisessa avaininfrastruktuurissa.

Kuten armeijan komentotie, myös kyseisen infrastruktuurimallin varmenneketju on tarkasti määritelty, eikä siitä voi lipsua. Varmenneketjun jonkin lenkin pettäminen aiheuttaa pahimmassa tapauksessa koko järjestelmän sortumisen.

Varmenneviranomainen. Hierarkkisessa julkisten avainten hallintajärjestelmässä ylimpänä sijaitsee *ylin varmenneviranomainen* (root certification authority, root CA). Ylimmän varmenneviranomaisen julkinen avain on aina jokaisen järjestelmässä olevan tahon ulottuvilla ja sen alkuperä on täysin luotettu.

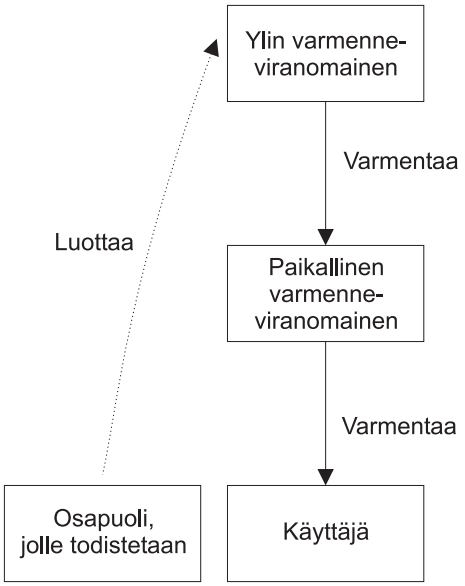
Tyypillisesti ylimmän varmenneviranomaisen avain on esimerkiksi pysyvästi koodattu ohjelmaan siten, että käyttäjä ei pysty sitä muuttamaan. Avain tai sen kryptografisesti vahva tiiviste voidaan myös julkaista esimerkiksi laajalevikkisessä painotuotteessa, jolloin avaimen autenttisuudesta voidaan varmistua hankkimalla riittävä määrä ko. julkaisua toisistaan riippumattomista lähteistä.

Nyt mikä tahansa ylimmästä varmenneviranomaisesta lähtöisin oleva varmenneketju pystytään tarkistamaan ja teoriassa minkä tahansa ketjussa olevan varmenteen alkuperä voidaan tarkistaa. Ylin varmenneviranomainen samaistetaan usein julkiseen avaimeensa, *juuriavaimeen*. Esimerkki kolmen varmenteen muodostamasta varmenneketjusta on kuvassa 4.

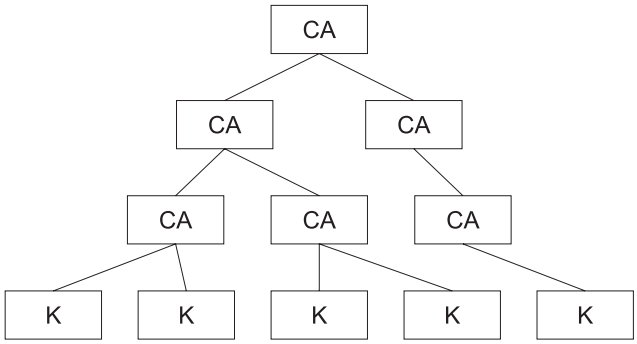
Ylimmän varmenneviranomaisen alapuolella on tavallisesti muita varmenneviranomaisia, joita kutsutaan tapauksesta riippuen eri nimillä ja jotka myös usein samaistetaan avaimiinsa. Tyypillisesti ylin varmenneviranomainen ei varmenna lainkaan loppukäyttäjien varmenteita vaan ainoastaan alempia varmennusviranomaisia, ja nämä hoitavat jokapäiväisen varmennustyön. Tähän on useita syitä; työmäärän tasaisemman jakautumisen lisäksi juuriavaimen salaista puoliskoa ei tarvitse näin käyttää kovin usein ja sen salassapito helpottuu. Juuriavaimen salainen puolisko saattaa esimerkiksi sijaita toimikortilla, joka on suljettu pankin holviin.

Juuriavain voi myös olla huomattavasti pidempi ja näin vaikeampi murtaa kuin paikallisten varmenneviranomaisten salaiset avaimet. Jos alemman varmenneviranomaisen turvallisuus pettää, ei se ole hierarkialle niin kohtalokasta kuin juuriavaimen *paljastuminen* tai *murtuminen* (compromisation).

Varmenneviranomaisten ja varmenteiden saajien muodostamaa pyramidia kutsutaan *varmennepuuksi* (certification tree) (kuva 5). Kuvassa K tarkoittaa julkista avainta ja siihen sidottua varmennetta. CA on varmenneviranomaisen julkinen avain ja siihen sidottu varmenne.



Kuva 4: Kolmitasoinen varmenneketju



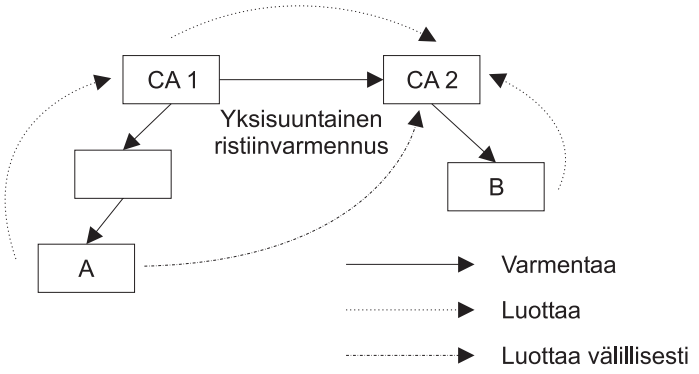
Kuva 5: Monihaarainen varmennepuu

Varmennusohjesääntö. Ylin varmenneviranomainen julkaisee tyypillisesti *varmennusohjesäännön* (certification practice statement, CPS), jota kaikkien samassa varmennuspuussa olevien on noudatettava. Varmennusohjesäännössä annetaan yleensä tarkat ohjeet avaimien luonnin teknisistä yksityiskohdista, salaisen avaimen säilytyksestä ja siitä, ketkä ovat kelvollisia varmennusviranomaisia toimimaan kyseisessä varmennuspuussa. Muita osa-alueita, joita varmennusohjesääntö käsittelee, ovat varmenteiden hankkijan oikeudet (jotka yleensä rajoittavat varmenneviranomaisen juridista vastuuta) ja varmenneanomuksien käsittelyohjeet.

Varmennusohjesääntö määrittelee myös sen, mitä varmenteessa oleva allekirjoitus todella tarkoittaa. Esimerkiksi näistä määritelmistä käy tämän hetken tunnetuimman kaupallisen varmenneviranomaisen VeriSignin varmennusohjesääntö [Ver97]. VeriSignin luokan 1 varmenne ei takaa muuta kuin että varmenteessa esiintyvä varmenteen omistajan sähköpostiosoite on yksikäsitteinen VeriSignin myöntämien varmenteiden keskuudessa. Luokan 3 varmenteen myöntämisehtona taas on, että varmenteen saaja käy näyttäytymässä varmenteen myöntäjän luona ja tallentaa salaisen avaimensa turvallisesti. On huomattava, ettei kyseinen varmennusohjesääntö kerro mitään varmenteen omistavan tahon luotettavuudesta sinänsä, ainoastaan nimen ja avainparin välisestä sidonnasta.

Yritysten sisäisissä varmennepuissa varmennusohjesäännön tulisi olla osa yrityksen kokonaisturvallisuusohjetta tai laatujärjestelmää.

Ristiinvarmennus. Kaksi varmennusviranomaista voivat varmentaa toistensa julkiset avaimet. Tätä kutsutaan *ristiinvarmennukseksi* (cross certification) ja se voi olla joko yksi- tai kaksisuuntaista. Esimerkiksi suuren yhtiön yksi tuoteprojekti voi varmentaa yhtiön ylimmän varmennusviranomaisen avaimen, jolloin tuoteprojektin varmentamat avaimet voivat muodostaa varmennusketjun muihinkin yhtiön avaimiin. Yhtiön ylin varmenneviranomainen taas ei välttämättä luota tuoteprojektin avaintenhallinnan pätevyyteen siinä määrin, että sen alla sijaitsevat varmenteet voisivat käyttää tuoteprojektin varmennusjuurta omassa varmennusketjussaan.



Kuva 6: Yksisuuntainen ristiinvarmennus

Esimerkki yksisuuntaisesta ristiinvarmennuksesta on kuvassa 6. Käyttäjä A luottaa varmenneviranomaiseen CA 1, joka on myöntänyt A:n varmenteen. Hän luottaa tällöin myös muihin samassa varmennepuussa oleviin varmenteisiin. Kun tämän varmennepuun ylin varmenneviranomainen ristiinvarmentaa

toisen varmenneviranomaisen CA 2, luottaa A välillisesti myös tähän toiseen varmenneviranomaiseen ja sitä kautta B:n varmenteen oikeellisuuteen. Myös B luottaa oman varmennepuunsa ylimpään varmenneviranomaiseen CA 2:een, mutta koska ristiinvarmennus on yksisuuntainen, se ei anna B:lle perusteita luottaa A:han tai CA 1:een.

Ristiinvarmennuksen ongelmana on erityisesti varmennusohjesääntöjen yhteensovittamisen ongelma. Ohjeet on yleensä kirjoitettu ihmisten (vieläpä lakimiesten) luettavaksi ja toimesta, eikä lakiteksti nykyisillä tekoälyjärjestelmillä taivu koneen ymmärrettäväksi. Mikäli varmennusketjua seuratessa hypätään varmennepuusta toiseen, tulisi koneellisen allekirjoituksen tarkistamisen lisäksi myös arvioida varmenteiden pätevyys oikeudellisesta näkökulmasta. Toisaalta elektronisen kaupan alalla ristiinvarmennus tai samaan juuriavaimeen luottaminen voidaan myös nähdä ongelmana, koska se saattaa pakottaa kilpailevat tahot täysin epäluonnolliseen luottamussuhteeseen keskenään [BFL96].

Ristiinvarmennuksen seurauksena varmennusketjua voidaan seurata kahden eri varmennepuun välillä, jolloin varmennusohjesäännöt yhdistetään siten, että:

- Kummankin ohjesäännön kaikki rajoitteet varmenteen käyttöä kohtaan on yhdistettävä ja jos käyttötarkoituksia ei jää jäljelle, ristiinvarmennusta ei voida hyväksyä. Jos esimerkiksi toinen varmenneviranomainen myöntää varmenteita vain sähköpostikäyttöön ja toinen veroilmoituksen allekirjoittamiseen, ei varmennuspolkuja voida seurata näiden varmennuspuiden välillä.
- Myös ohjesääntöjen rajoitukset siitä, kenelle varmenne voidaan myöntää ja miten salaista avainta on säilytettävä, on yhdistettävä. Mikäli kohdevarmenteen omistaja ei täytä kummankin varmennusohjesäännön minimivaatimuksia, ei ristiinvarmennus ole mahdollista.

Kuten yllä olevasta voidaan päätellä, ristiinvarmennuksien ongelmien monitahoisuus tekee automaattisen ristiinvarmennuksen huomioon ottamisen lähestulkoon mahdottomaksi. Mikäli varmennusohjesäännöt eivät ole täysin samat, käytännön toteutuksissa ainoa realistinen mahdollisuus on antaa molempien varmennuspuiden varmennusohjesääntö käyttäjän luettavaksi (tai yhtiön turvallisuuspolitiikan tai laatujärjestelmän kehittäjien erityisesti hyväksyttäväksi) ja sen jälkeen olettaa, että ristiinvarmennukset voidaan hyväksyä. Tämä ei ole käyttäjäystävällistä eikä välttämättä kovinkaan turvallista.

3.2 Luottamus hierarkkisessa järjestelmässä

Kunvarmenneviranomainen allekirjoittaa salaisella avaimellaan varmenteen ja jokin taho pystyy tarkistamaan kyseisen allekirjoituksen onnistuneesti, tämä taho voi luottaa siihen, että jos salainen avain on edelleen yksin allekirjoittajan tiedossa, allekirjoitus on jonkinasteinen vakuutus varmenteen sisältävän tiedon aitoudesta.

Jos taho on vakuuttunut myös allekirjoittajan julkisen avaimen aitoudesta, sanotaan ko. tahon ja varmenteessa olevan avaimen välillä vallitsevan *luottamusta* (trust). Mikäli juuriavain on turvallisesti (ja usein pysyvästi) ko. tahon saatavilla, se muodostaa *luottamusjuuren* (trust root), johon kaikki muut luottamussuhteet voi perustaa. Kyseisellä taholla voi olla useita luottamusjuuria, tyypillisesti yksi jokaista varmennepuuta kohden.

Vaikka terminä käytetään sanaa "luottamus", on huomattava, ettei se missään tapauksessa implikoi mitään muuta luottamusta kuin uskon siihen, että varmenteessa tullut julkinen avain ja varmenteen muut tiedot liittyvät toisiinsa varmenneohjesäännön ilmaisemalla tavalla. Ei voida ajatella, että julkisen avaimen omistava taho olisi jollakin tavalla luotettava, ellei varmenneohjesääntö kerro jotakin kyseisen tahon luotettavuudesta (jokin ohjesääntö voisi esimerkiksi edellyttää luottokelpoisuustarkistuksen).

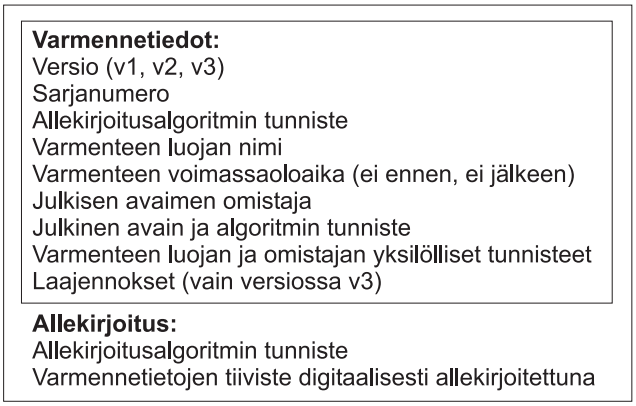
Luottamus ei myöskään ole koskaan sataprosenttista. Jos voidaan olla vakuuttuneita siitä, että luottamus on paikallisen turvallisuuspolitiikan valossa riittävä, sanotaan avaimen olevan "luotettava". Luotettavankin allekirjoituksen tulkitsemisessa on aina pieni virhemarginaali [JY98]. Avainten ja luottamuksen hallinta on aina myös riskien hallintaa.

3.3 Varmenteet hierarkkisessa järjestelmässä

Ehdottomasti tämän hetken tunnetuin ja käytetyin varmennetyyppi on ITU-T:n[9] suositus X.509 [ITU93]. Sitä käyttävät useimmat Internetissä käytössä ja suunnitteilla olevat turvallisuusprotokollat kuten S/MIME [DHRW98] (sähköposti), SSL ja TLS [DA99] (nk. TCP/IP-protokollapinossa TCP-tason yläpuolinen turvallisuuskerros), IPSec [KA98] (IP-tason turvallisuuskerros) ja SET [SET97] (luottokorttimaksut), muutamia mainitakseni.

X.509 on kuvattu ASN.1-kuvauskielellä. Varmenne koostuu kuvassa 7 esitetyistä osista.

[9] International Telecommunications Union, Telecommunication Standardisation Sector



Kuva 7: X.509-varmenteen rakenne

Ensimmäinen X.509-varmenteiden silmiin osuva ja hyvin paljon käytännön hankaluuksia Internet-käytössä aiheuttava osa on Distinguished Name (DN), joka on ISOn[10] tapa antaa nimi kaikille olemassa oleville asioille X.500-hakemistopuussa. X.509-varmenteessa varmenteen omistajan ja luojan nimet ilmoitetaan tässä muodossa. Kuten tunnettua, Internetin osoitteistus on rakentunut pääosin käyttäjätunnuksen tai postin vastaanottajan nimen ja vastaanottajan postin käsittelevän koneen nimen yhdistelmistä (tunnus@kone). Tämän vuoksi X.500-nimi, jossa tyypillisinä osina ovat etunimi, sukunimi, kaupunki, maa, organisaatio ja organisaatioyksikkö, sopii varsin huonosti Internet-ympäristöön.

X.509-varmenteiden kolmanteen versioon on tullut mahdollisuus lisätä laajennoksia, joilla voidaan esimerkiksi määrätä varmenteen käyttötarkoitus. Näillä laajennoksilla voidaan varmenteeseen sisällyttää myös Internet-tyylisiä sähköpostiosoitteita [Gut98].

X.500-nimien luontevin käyttöympäristö lieneekin yritysmaailmassa, missä jokaiselle henkilölle voidaan osoittaa tietty organisatorinen yksikkö, ja X.500-nimien luonti voidaan keskittää jollekin taholle, joka pitää ne säännönmukaisina.

X.509 on ongelmistaan huolimatta saanut varsin tukevan jalansijan käytännön sovellutuksissa ja on erittäin todennäköistä, että se jatkaa tärkeänä varmennetyyppinä pitkälle tulevaisuuteen. Koska sovellusten ei tarvitse välttämättä käsitellä X.500-nimikenttää, voidaan sitä käyttää (yhteensopivuus joidenkin hakemistopalvelujen kanssa uhraamalla) muun kaltaisen osoitteen

[10] International Standardisation Organisation

varastointiin tai se voidaan jättää tyhjäksi. Vastaavasti laajennosmekanismilla voidaan X.509-varmenteeseen kiinnittää lähes mitä tahansa dataa, jolloin varsinainen X.509 jää pelkäksi torsoksi, mutta mahdollistaa jonkinasteisen uudelleenkäytön X.509-varmenteita käsittelevien ohjelmien kanssa.

Esimerkki: PEM. Ensimmäinen huomiota saanut ehdotus yleiseksi varmenneinfrastruktuuriksi oli vuonna 1993 julkaistu PEM (Privacy Enhanced Mail) [Lin93], [Ken93], [Bal93], [Kal93]. PEMin määrittelemän avaininfrastruktuurin juurena toimii IPRA (Internet Public Key Registration Authority), jonka alla toimii useita PCA:ta (Policy Certification Authority). Kullakin PCA:lla on oma varmennusohjesääntönsä[11], jonka ne esittävät IPRAlle hankkiessaan avaimeensa IPRAn allekirjoituksen. PCA:den alla toimii taas useita CA:ta (Certification Authority), ja vasta näiden alla ovat loppukäyttäjät.

PEMin suunnittelu lähti ylväistä lähtökohdista — olisi olemassa jonkinlainen Internetin globaali luottamusjuuri, johon kaikki voisivat varauksetta luottaa. Koskaan ei tulisi tilannetta, jossa varmenteen luottamusta ei voitaisi määrittää, vaan kaikilla olisi kaikkialla aina yksi piste, johon he voisivat tukeutua. Internetissä niin tärkeän anonymiteetin turvaksi PEMiin oli jopa rakennettu käsite "persoonavarmenteet", jotka olivat erityisten PERSONA-varmenneviranomaisten myöntämiä pseudonyymivarmenteita.

PEMistä ei kuitenkaan koskaan tullut suosittua. Sen suurin vika oli jäykkä infrastruktuuri, joka olisi pakottanut kaikki organisaatiot samaan luottamuspuuhun. Kun ymmärrämme esimerkiksi yritysten halun pitää omat organisaationsa omana tietonaan, ei ole vaikea ymmärtää, miksi (Yhdysvalloissa sijaitseva) IPRA ei ollut niiden mielestä hyvä idea. PEMin laajempi käyttöönotto olisi myös edellyttänyt kattavaa PCA- ja CA-verkostoa yksityiskäyttäjiä varten. Kun valmista infrastruktuuria ei välittömästi ollut saatavissa ja yksityiskäyttäjien tiedonsalaustarpeesta oli samaan aikaan kilpailemassa PGP (ks. myöhemmin), PEMin loppusaldoksi jäi muutama yksittäinen implementaatio.

Tekniseltä toteutukseltaan PEM ei myöskään venyisi enää nykyvaatimuksiin. Kuten nimestäkin voi päätellä, PEM-varmennepuu oli tarkoitettu sähköpostin varmennukseen. Nykyään sähköposti nähdään vain yhtenä turvallisuutta ja varmenteita tarvitsevana sovelluksena lukuisten muiden joukossa. PEM ei myöskään ottanut huomioon sähköpostin muuttumista erilaisia tietotyyppe-

[11] PEM käyttää termiä "policy statement".

jä (MIME-objekteja)[12] välittäväksi kanavaksi. PEM käsitteli viestiä yhtenä monoliittisena kappaleena eikä rakenteellisena objektipuuna, jollaisena tyypillinen sähköpostiviesti voidaan kuvata.

Nykykäytäntö. PEMin kaltaisia massiivisia ja kaikenkattavia varmennejärjestelmiä ei sittemmin ole syntynyt Internet-käyttöä varten. Nykytilanteessa jokaisella organisaatiolla on yleensä oma varmenneinfrastruktuurinsa (usein, mutta ei aina, X.509-pohjainen). Mikäli taho haluaa olla valmis tarkistamaan aiemmin tuntemattomien avaimien luottamusparametreja, sillä on oltava kattava valikoima juurivarmenteita. Esimerkiksi suosittu seittiselain Netscape Communicator sisältää 1997 julkaistussa 4.03-versiossaan 29 eri juurivarmennetta "tehdasasenteisena".

Tällä hetkellä käyttökelpoisimmilta turvasähköpostijärjestelmiltä vaikuttavat X.509-varmenteita hyödyntävät S/MIME (secure MIME) ja PGP:n päälle rakennettu PGP/MIME, tulevaisuudessa ehkä myös Open-PGP.

IETF:n (Internet Engineering Task Force, löyhä ja kaikille avoin organisaatio, spesifioi mm. uusia protokollia Internet-käyttöön), uusin ehdotus PKIX (Public Key Infrastructure (X.509)) yrittääkin yhden ison varmennepuun sijasta luoda yhteisiä pelisääntöjä X.509-pohjaisille varmennejärjestelmille. PKIX ottaa kantaa mm. ristiinvarmennukseen, varmennusohjesääntöihin ja varmenteiden välitysprotokolliin. Tavoitteena on paikata suurimpia X.509-varmennejärjestelmien yhteensopimattomuuksia. Yhtenä esimerkkinä olkoon varmennuspyyntöviestin syntaksi. Varmennuspyyntö eli uuden julkisen avaimen lähetys varmenteen allekirjoittajalle kulkee nykyään yleensä PKCS #10 -tyyppisenä [Kal98] viestinä. Tässä viestimuodossa on joitakin puutteita X.509 v3-laajennosten käsittelyssä [Tha98] ja PKIX yrittää paikata tilannetta. Yhteensopivuusongelmien riivaamalla alalla PKIX on otettu hyvin vastaan, joskin ylimenokaudella vanhojen järjestelmien ja PKIX:n tukeminen aiheuttaa lisätyötä. PKIX saattaa lyödä itsensä läpi jo parin vuoden sisällä.

Luottamusverkkojen ja hierarkkisen infrastruktuurin yhdistäminen. Käytännössä nykyään suosituinta mallia edustaa tyyppi, jota ICE-TEL-projekti [KCYGF96] tutki vuosina 1995-1997. Projektin tekijöiden tarkoituksena oli yhdistää hierarkkiseen infrastruktuuriin nk. luottamusverkkomalli [KCYC96] (ks. kappale 3.4), mutta käytännössä heidän mallinsa on lähinnä PEMiä vapaammalla politiikalla varustettu X.509-rakennelma. Järjestelmä yrittää "formalisoida" usean varmennepuun idean.

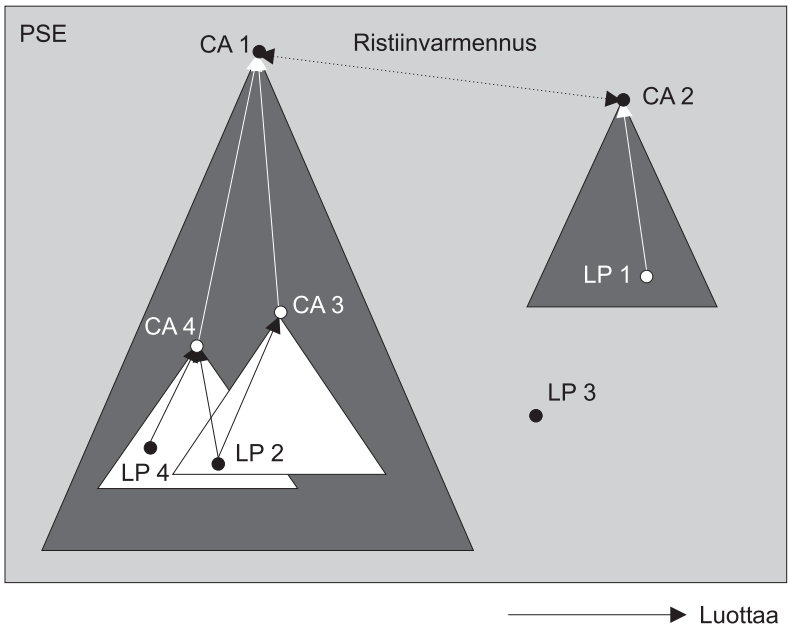
[12]MIME (Multipurpose Internet Mail Extensions) on laaja kehys, joka mahdollistaa eri tietotyyppien liittämisen samaan (tyypillisesti sähköpostitse lähetettävään) dokumenttiin ja näiden suhteiden ja siirtotavan kuvaamisen.

ICE-TELin järjestelmässä jokaisella taholla on *henkilökohtainen turvaympäristö* PSE (Personal Security Environment). Se koostuu *turva-alueista* (security domain). Jokainen PSE:hen lisätty varmenneviranomaisen julkinen avain virittää itselleen oman turva-alueen, johon sen varmentamat varmenteet kuuluvat. Turva-alueet voivat olla sisäkkäisiä (varmenneviranomaisen alaisuudessa toimiva toinen varmenneviranomainen) tai erillisiä (ristiinvarmennuksella linkitettyjä tai linkittämättömiä). Turva-alueen voi virittää vain sellainen julkinen avain, jonka varmentaneen tahon varmennusohjesääntö on arvioitu ja lisätty PSE:hen. Esimerkiksi tavallinen henkilövarmenne ei viritä omaa turva-aluetta. Turva-alueen virittävää julkista avainta kutsutaan mallissa *luotetuksi pisteeksi* (trusted point). Myös yksittäinen, turva-alueelle kuulumaton henkilövarmenne voi olla oma luotettu pisteensä.

Luottamusta arvioiva taho, todentaja, voi luottaa avaimeen vain, jos se sijaitsee todistajan omassa PSE:ssä. Viestiessään muiden tahojen kanssa lähettäjä liittää yleensä viestin mukaan varmennepolun omasta varmenteestaan valitsemaansa luotettuun pisteeseen, jonka lähettäjä uskoo olevan vastaanottajan PSE:ssä. Tämä helpottaa vastaanottajan toimintaa, koska vastaanottajan ei tarvitse tällöin itse hankkia varmennepolun varmenteita varmistuakseen lähettäjän varmenteen oikeellisuudesta. Jos jokin PSE:n varmenne kuuluu kahteen tai useampaan turva-alueeseen, voi lähettäjä liittää mukaan minkä tahansa näistä varmenneketjuista. Kyseisen luotetun pisteen varmennusohjesäännöstä ja vastaanottajan PSE:n rakenteesta riippuu, minkälainen luottamus lähetettyyn avaimeen todentajan (eli viestin vastaanottajan) PSE:ssä syntyy.

Kuvassa 8 on esitetty esimerkki käyttäjän PSE:stä. PSE:hen on lisätty neljän varmennusviranomaisen varmenteet (CA 1, ..., CA 4) sekä tieto siitä, että näiden varmennusohjesäännöt on hyväksytty ja saatavilla PSE:ssä. Jokainen varmenneviranomainen virittää oman turva-alueensa. Luotetut pisteet LP 1, LP 2 ja LP 4 voidaan siis lisätä PSE:hen ilman enempiä toimenpiteitä. Kun PSE:n käyttäjä lähettää viestin vastaanottajalle, hän voi valita sen varmenneketjun, joka viestiin liitetään mukaan. LP 2 kuuluu kolmeen turva-alueeseen ja lähetettävä varmenneketju voi kiertää joko CA 3:n tai CA 4:n kautta, mahdollisesti CA 2:een asti.

Itsevarmennus. Luottamuspuihin kuuluvien varmenteiden lisäksi esimerkkimme PSE:ssä on yksi varmenne (LP 3), joka on nk. *itsevarmennettu* (self-certified). Tällöin salaisen avaimen omistaja on allekirjoittanut oman julkisen avaimensa. Luottamus kyseiseen avaimeen on tällöin täysin PSE:n omistajan päätettävissä. Koska itsevarmennetulla luotetulla pisteellä ei ole varmennusohjesääntöä, se ei voi virittää turva-aluetta eikä naamioitua varmen-



Kuva 8: Henkilökohtainen turvaympäristö PSE

nusviranomaiseksi. Itsevarmennetut avaimet ovat käyttökelpoisia tilanteessa, jossa on avattava salattu tiedonsiirtokanava kahden osapuolen välillä ilman, että tiedon alkuperän todennuksella on suurta merkitystä, tai jos halutaan, että avaimen haltijan identiteetti tai julkinen avain eivät ole näkyvissä missään hakemistossa. Useat turvasähköpostiohjelmat tukevat varmenteiden itsevarmennusta.

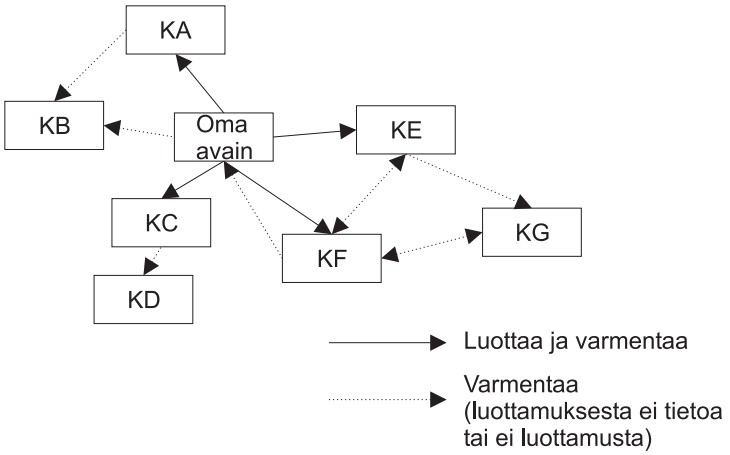
3.4 Luottamusverkot

Luottamus ihmisjoukossa. Kuvitellaan hypoteettinen kryptografia-alan konferenssi. Konferenssikeskuksen käytävillä kävelee toinen toistaan akateemisemman näköisiä asiantuntijoita ja seisahdan hetkeksi pieneen piiriin, jossa kuulen keskusteltavan mielenkiintoisesta aiheesta. Edellispäiväisessä grillijuhlassa tapaamani tutkijat A ja B tunnistavat minut ja esittelevät minut professorille P ja ohjelmistokehittäjille C ja D. Koska A ja B vaikuttivat grillijuhlissa päteviltä henkilöiltä, voin helposti uskoa, että esimerkiksi P on todella heidän nimeämänsä kuuluisa professori eikä esimerkiksi NSA:n agentti. Myös C ja D, joista C on vieläpä P:n entinen oppilas, puhuvat P:stä yhtenevästi, näin syventäen luottamustani P:n oikeaan identiteettiin. Kun P

myöhemmin henkilökohtaisesti antaa minulle sähköpostiosoitteensa (”minä, P, väitän, että tämä on oikea sähköpostiosoitteeni”) on sen alkuperän tarkistettu *luottamusverkon* (network of trust) avulla, jossa luottamusjuurena on oma luottamukseni $A, \ldots, D$:hen.

Luottamusverkko. Luottamusverkon jokainen solmu on oma luottamusjuurensa. Ainoa asia, johon avainparin omistaja uskoo *ehdoitta* (axiomatically) on hänen salaisen avaimensa sijainti ja eheys. Kun avainparin omistaja kohtaa julkisen avaimen, jonka alkuperään hän haluaa luottaa, hän allekirjoittaa kyseisen avaimen omalla salaisella avaimellaan. (Kryptokonferenssianalogiana henkilö painaa tapaamansa henkilön kasvonpiirteet ja nimen mieleensä ja päättää luottaa kyseiseen henkilöön.) Muut henkilöt ja avainparin omistajan käyttämät ohjelmat voivat tukeutua tähän allekirjoitukseen. Verkon solmu, vaikkapa solmu T, voi päättää luottaa toiseen solmuun täysin oman politiikkansa mukaan: kun solmu saa haltuunsa jonkin toisen solmun A julkisen avaimen, hän voi vaatia kommunikointikumppaniaan esittämään esimerkiksi todistuksen henkilöllisyydestään.

Jos luottamusverkon solmu T on päättänyt luottaa solmun A julkisen avaimen autenttisuuteen, se voi lisätä A:n avaimeen oman allekirjoituksensa, luoden näin varmenteen. Varmenne voidaan antaa kolmannelle osapuolelle B. Mikäli hänellä on ennestään olemassa luottamussuhde varmenteen antajan T kanssa, voi B (oman politiikkansa määräämissä rajoissa) muodostaa *välillisen luottamussuhteen* A:han T:n kautta.



Kuva 9: Luottamusverkko

Esimerkki luottamusverkosta. Kuvassa 9 on pieni luottamusverkko, jonka sydämenä on jonkin henkilön oma avainpari. Hän on varmentanut viisi

tuntemaansa julkista avainta. Samalla hän on ilmaissut luottavansa avaimiin KA, KC, KE ja KF niin paljon, että hän luottaa myös näiden julkisten avainten omistajien varmentamiin julkisiin avaimiin. Tämä oletus ei seuraa suoraan varmennuksesta: esimerkiksi avaimen KB alkuperään luotetaan, koska se on varmennettu, mutta tämän avaimen edelleen varmentamiin avaimiin ei luoteta.

Koska luotamme avaimen KE alkuperään ja myös sen varmentamien avainten alkuperään, kuvan tilanteessa voimme luottaa esimerkiksi avaimen KG alkuperään.

Muiden avainten välisistä luottamussuhteista näemme ainoastaan mahdolliset allekirjoitukset. Emme esimerkiksi tiedä, luottaako KG:n omistaja omaan avaimeemme välillisesti KF:n kautta vai ei.

Luottamusverkkomallissa on useita parametreja, jotka on määrättävä luottamuksen määrän arvioimiseksi: onko luottamus, tai sen puute, aina täydellistä (kuten kuvassamme) vai onko esimerkiksi mahdollista luottaa avaimeen osittain? Luotammeko enemmän KE:hen vai KF:ään? Luottaisimmeko KG:hen, ellei meillä ei olisi kahta erillistä varmennuspolkua (KE:n ja KF:n kautta) kyseiseen avaimeen? Luotammeko KD:hen, koska siihen kulkee vain yksi varmennuspolku? Kaikkiin kysymyksiin on mahdollista löytää esimerkkivastine reaalimaailman ihmissuhteista.

Luottamusverkot käytännössä. Tunnetuin tälle periaatteelle rakennettu sovellus on PGP (Pretty Good Privacy), alun perin ilmainen ja sittemmin myös kaupallistettu ohjelma. PGP:n varmenteet (joita PGP itse kutsuu vain ”avaimiksi”) sisältävät julkisen avaimen ja omistajan identifioivan tunnisteen lisäksi mielivaltaisen monta allekirjoitusta. Käyttäjä tallentaa avaimensa *avainrenkaisiin* (key ring). PGP:n käyttäjä voi tarkistaa varmenteessa olevan allekirjoituksen mikäli hänen avainrenkaassaan on vastaava julkinen avain. Mitä useampia avaimia käyttäjä on avainrenkaaseensa kerännyt, sitä useamman allekirjoituksen hän voi tarkistaa.

Jokaiseen avainrenkaan avaimeen on liitetty luottamusparametri, joka on käyttäjäkohtainen. Käyttäjä määrittelee luottamusparametrit sen mukaan, kuinka luotettavana hän itse henkilökohtaisesti pitää avaimen omistajaa kyseisellä sovellusalueella. PGP määrittelee avainta käytettäessä kyseisen avaimen kokonaisluottamuksen ja ilmoittaa sen käyttäjälle (”tämä avain on täysin luotettu” tai ”tämän avaimen luottamus on puutteellinen, koska varmenteen allekirjoituksia ei voitu tarkistaa”). PGP suorittaa tällöin *luottamuksenhallintaa* (trust management), josta myöhemmin.

PGP:n luottamusverkoista ei ohjelman suuresta suosiosta huolimatta ole kasvanut kovin kattavia. Vuoden 1997 lopussa tehdyn tutkimuksen [McB98] mukaan vain noin yhdessä kolmanneksessa julkisista hakemistopalveluista saatavista PGP-varmenteista oli jonkin toisen osapuolen allekirjoitus ja vain yksi kuudesosa oli useamman kuin kahden tahon allekirjoittamia.

Tutkimuksessa käytetyssä varmennetietokannassa oli 57582 PGP-varmennetta. Jos näiden muodostama luottamusverkko ajatellaan suunnatuksi graafiksi siten, että "julkisen avaimen A omistaja on varmentanut avaimen B" on nuoli solmusta A solmuun B, sen suurin vahvasti kytketty komponentti (mahdollisimman iso osa solmuista, jossa jokaisesta solmusta on polku kaikkiin muihin kyseisen komponentin solmuihin) oli vain 3100 varmenteen suuruinen. Jos mukaan lasketaan myös ne varmenteet, joihin on polku jostakin em. komponentin solmusta mutta jotka eivät itse kuulu ko. komponenttiin, suurimman toimivan luottamusverkon koko oli 6662 varmennetta.

3.5 Identiteetti- ja attribuuttivarmenteista

Identiteettivarmenteiden ongelmista. *Identiteettivarmenne* (identity certificate) tai *nimivarmenne* sitoo julkisen avaimen johonkin luonnolliseen henkilöön tai muuhun tahoon, jolla on nimi. Koska yleisimmät varmennetyypit, X.509- ja PGP-varmenteet, ovat identiteettivarmenteita, suurin osa varmenteiden käyttäjistä olettaa kaikkien varmenteiden olevan tätä tyyppiä.

Henkilön tunniste voi olla esimerkiksi sähköpostiosoite tai nimi. Useimmiten on kuitenkin tarkoituksenmukaista yrittää mallintaa todellisen maailman konsepteja myös tietokonemaailmassa aina kun mahdollista. Otetaan esimerkiksi seuraavanlainen skenaario: asunnon ovessa on lukko ja lukkoon sopii yksi avain. Mikään muu avain ei aukaise lukkoa. Lukko ei välitä siitä, kenen omistama avain lukossa on, kunhan sen kolot ovat oikeilla kohdillaan ja lukolla on edellytykset avautua (tällä on suora analogia haaste/vaste-protokollaan). Lukko ei myöskään välitä avainta kääntävän henkilön identiteetistä — lukko samaistaa avaimen sen omistajaan.

Miksi avaimen sitominen nimeen ei ole kannattavaa? SPKI (Simple Public Key Infrastructure) [EFL+98a] sekä [Ell96a] ja [Ell96b] esittävät useita hyviä näkökohtia. Ensinnäkään henkilön nimi ei ole yksikäsitteinen erottaja: saman nimisiä henkilöitä voi olla useita. [EFL+98a] esittää teorian, jonka mukaan ihmisten nimien luontaisen nimiavaruuden koko on juuri niin suuri,

että jokaiselle riittää yksikäsitteinen nimi agraarikulttuurissa ja pienissä kyläyhteisöissä, mutta ei esimerkiksi jättikokoisissa kaupungeissa tai Internetin kaltaisessa valtavassa kommunikaatiojärjestelmässä. [Ell96a] tutki keskisuuren yrityksen nimiavaruutta ja totesi, että kyseisessä yrityksessä noin joka 360. ihmisellä oli kaima (sama etu- ja sukunimi).

Jos nimeen liitetään muita tietoja (ikä, sähköpostiosoite, henkilötunnus), tilanne paranee, mutta mikään näistä ei voi taata yksikäsitteisyyttä — varsinkin kun kuka tahansa voi aina keksiä itselleen pseudonyymin, jonka turvin esiintyä (ja antaa vaikka kaikille pseudonyymeilleen oman sähköpostiosoitteen). Lisäksi esimerkiksi iän paljastaminen on identifiointitarkoituksia varten useimmiten täysin irrelevanttia ja henkilötunnus esimerkiksi USA:ssa on Suomen vapaasta normaalikäytännöstä poiketen hyvin luottamuksellista tietoa.

Kun tunnemme henkilön ja tiedämme hänen nimensä, on ihmiselle luontaista olettaa jotakin myös hänen olemuksestaan — onko hän samaa ammattiryhmää, mahdollisesti saman yhdistyksen jäsen, vanha ystävämme tai niin edelleen. Vertauksena mainittakoon jokin pieni kaupunki, jossa kaikki tuntevat toisensa ja pystyvät päättelemään nimen perusteella vaikkapa harrastuksesi tai kotiintuloaikasi.

Lopulta, nimen ja olemuksen perusteella, voimme tehdä joitakin johtopäätöksiä kyseisen henkilön identiteetistä — onko hän luotettava, voiko hänelle antaa pääsyn johonkin järjestelmään ja niin edelleen, vaikka näillä asioilla ei todellisuudessa pitäisi olla mitään merkitystä pääsynvalvonnan kannalta, olettaen, että henkilöllä on hallussaan hänelle kuuluva oikea avain.

Vaikka päättelyketju toimiikin mahdollisesti suljetussa tai pienessä yhteisössä, hajoaa se käsiin jos sitä yritetään laajentaa suuriin henkilömassoihin. Tämä on hyvä syy sille, miksi varmenteissa tulisi pyrkiä eroon nimistä identifioivana seikkana.

Palataksemme avain-lukkoanalogiaan, ajatellaan tilannetta, jossa avaimen omistaja on antanut avaimensa vartiointiliikkeelle. Vartiointiliike, vaikka sen palveluksessa onkin ihmisiä, ei ole henkilö vaan valtuutettu agentti, ja koska tavalliseen avaimeen ei ole sidottu henkilön identiteettiä, agentti voi toimia alkuperäisen avaimen omistajan valtuuttamana. Mikäli lukko olisi tietoinen avaimen omistajan identiteetistä, pelkkä avaimen luovutus vartiointiliikkeelle ei riittäisi: lukolle olisi kerrottava mahdollisesti kaikkien vartiointiliikkeen työntekijöiden nimet.

Nimen sitominen avaimeen saa hajautetuissa järjestelmissä vieläkin epämiellyttävämpiä piirteitä: avain saattaakin kuulua esimerkiksi jollekin ohjelmalle, jota ajetaan useita instansseja eri puolilla maailmaa. Ohjelmaa ei voi sitoa mihinkään yksittäiseen henkilöön, eikä sen ohjelmoijalla tai ajoympäristön (fyysisen tieto- tai virtuaalikoneen) omistajalla ei välttämättä ole mitään tekemistä ohjelman kanssa. Silti sen on pystyttävä identifioimaan itsensä muille osapuolille, jotka saattavat nekin olla ohjelmia ja näin ollen kykenemättömiä tekemään johtopäätöksiä inhimillisistä tekijöistä.

Vähemmän ajateltu seuraus nimen käytöstä varmenteissa on se, että pääsylistoista ja varmennelistoista muodostuu henkilörekistereitä. Henkilörekisterien ylläpitoon liittyvät lait ja esimerkiksi Suomen lain mukainen henkilörekisterin tarkastusoikeus saattavat aiheuttaa epämiellyttäviä yllätyksiä ohjelmiston ylläpitäjälle ja operaattorille.

Paikalliset nimiavaruudet. Avaimen omistajan identiteetin samaistaminen avaimeen kuulostaa erinomaiselta idealta: julkinen avain on jo määritelmänsä mukaan uniikki ja julkinen. Se voidaan siis turvallisesti tallettaa järjestelmään ja sitä tai vaikkapa sen kryptografisesti vahvaa tiivistettä voidaan käyttää tietokantajärjestelmissä suoraan yksikäsitteisenä hakuavaimena. Koska avaimella voi digitaalisesti allekirjoittaa valtuutuksia jonkin asian tekemiseksi, sitä kutsutaan *valtuuttajaksi* (principal).

Nimet ovat kuitenkin tärkeitä esimerkiksi sähköpostin salausprotokollien varmenteissa. Julkinen avain itsessään ei ole helposti ihmisen muistettavissa tai kirjoitettavissa, koska se on käytännössä vain iso (satoja numeroita pitkä) luku. Pienissä järjestelmissä myös pääsylistan ylläpito saattaa olla vielä realistista ja helppoa, jolloin nimen sitominen avaimeen on järkevää. Koko ajan on kuitenkin pidettävä mielessä, että luottamus on arvioitava avaimeen — ei nimeen — nähden.

Jotta vältettäisiin globaalin X.500-tyylisen nimiavaruuden hierarkian muodostamat ongelmat, (Simple Distributed Security Infrastructure) [RL96] tarjoaa ratkaisuksi *paikallista nimiavaruutta*, jossa avaimiin kyllä sidotaan nimi, mutta näiden ei edes oleteta olevan mitenkään uniikkeja globaalisti vaan ainoastaan paikallisen nimiavaruuden sisällä. Käytännössä esimerkiksi sähköpostiohjelman kontaktitietokanta, johon on tallennettu käyttäjän useimmiten tarvitsemat osoitteet, voi muodostaa paikallisen nimiavaruuden.

Paikallinen nimi voidaan muuttaa *yleiseksi nimeksi* liittämällä nimen mukaan paikallisen nimiavaruuden omistajan julkinen avain tai sen kryptografisesti

vahva tiiviste, jotka määritelmän mukaan ovat globaalisti uniikkeja ja näin tekevät paikallisesta nimestä myös globaalisti uniikin.

Oletetaan taho nimeltään `Käyttäjä`, jolla on tietokone nimeltään `tietokone`. Jollakin muullakin taholla saattaa olla nimiavaruudessaan `tietokone`, mutta `Käyttäjän` nimiavaruuden `tietokone` erotetaan kaikista muista merkitsemällä `(ref: Käyttäjä tietokone)`. Merkintä `ref:` tarkoittaa kulloisenkin todentajan omaa nimiavaruutta; jos todentajalla ei ole nimiavaruudessaan `Käyttäjää`, edellisellä esimerkillä ei ole järkevää tulkintaa.

Paikallisten nimien lisäksi tiettyihin erikoistarkoituksiin voidaan varata globaaleja nimiavaruuksia. Tällöin kyseessä on sovellus, jonka yhteydessä globaalin nimen käyttö on perusteltua tai jopa oleellista. SDSI-ehdotelmassa [RL96] yksi erityiskohtelua nauttivista nimiavaruuksista on Internetin nimipalvelu DNS, jota merkitään nimellä `DNS!!`. Edellisen esimerkin tietokoneen globaali nimi voisi olla siis esimerkiksi `(ref: DNS!! fi firma Käyttäjä tietokone)`, jos `Käyttäjän` sähköpostiosoite on `<Käyttäjä@firma.fi>`.

SDSI:ssä on määritelty myös nimien *ryhmät*, joissa jollekin nimelle voidaan antaa ominaisuus "kuuluu ryhmään". Ryhmiä on erityisesti pääsylistojen tapauksessa helpompi käsitellä kuin yksittäisiä nimiä.

Niin ikään SDSI:n tapauksessa on hyvin hyödyllistä luoda *rooleja*. Mikä tahansa taho voi luoda itselleen mielivaltaisen monta avainparia ja antaa niille nimiä omasta paikallisesta nimiavaruudestaan. Osa-aikaisesti työssä olevalla opiskelijalla `Teekkari` voisi mahdollisesti olla kaksi roolia, `(ref: Teekkari opiskelija)` ja `(ref: Teekkari työharjoittelija)`. Nostaessaan opintotukea `Teekkari` toimii ensimmäisessä roolissa ja päästäkseen työpaikan kulunvalvonnasta läpi hän voisi käyttää toista.

Viimeisenä erityistapauksena mainittakoon *delegointi*, jossa jokin taho antaa toiselle taholle oikeuden toimia itsenään. Tilanteesta riippuu, periytyykö delegointioikeus vielä eteenpäin oikeuden saaneeltakin taholta. Tämä ei ole sama asia kuin nimen kiinnitys valtuuttajaan; delegoinnissa jokin valtuuttaja antaa toiselle valtuuttajalle omia oikeuksiaan, ilman että valtuuttaja ja valtuutettu samaistuvat toisiinsa.

Nimien kanonisoinnista. Koska SDSI on *avainkeskeinen* järjestelmä, kaikki operaatiot ja luottamus on kasattu pelkkiin julkisiin avaimiin. Kuten aiemmin todettiin, allekirjoittavaa avainta kutsutaan *valtuuttajaksi*. Kun avain sidotaan nimeen, nimestä tulee *valtuutettu* ja kyseessä on *agenttirelaatio* (speaks-for relation) [Aba97]. Merkitään relaatiota nuolella ($\mapsto$). Suh-

de `Teekkari` $\mapsto K$, jossa K on julkinen avain, tarkoittaa tällöin "`Teekkari`lla on lupa toimia K:n nimissä".

Toinen tärkeä suhde on *väiterelaatio* (says relation). Esimerkiksi "`Teekkari` *väittää, että* `(ref: Teekkari opiskelija)` $\mapsto L$" ilmaisee, että `Teekkari` (eli oikeastaan K) väittää, että `opiskelija`lla (eli `Teekkarin` eräällä roolilla) on oikeus toimia L:n (roolia varten luodun avaimen) puolesta. Tyypillisesti väitteet tehdään allekirjoittamalla väite väitteen esittäjän salaisella avaimella. Väitteen uskottavuus on ekvivalentti vastaavan julkisen avaimen luottamuksen kanssa.

Kun avaimia käytetään, on yleensä tarpeen muuntaa nimiä nimiavaruudesta toiseen. Lopullisen luottamuksen arvioinnin vuoksi jokaiselle nimelle on löydettävä sitä vastaava valtuuttaja (julkinen avain), jolla allekirjoitukset voidaan tarkistaa. Nimiavaruuksille on määritelty seuraavat aksioomat [Aba97]:

$p \mapsto p$	Refleksiivisyys
$(p \mapsto q) \Rightarrow ((q \mapsto r) \rightarrow (p \mapsto r))$	Transitiivisuus
$(p \mapsto q) \Rightarrow (\texttt{(ref: } p\ r\texttt{)} \mapsto \texttt{(ref: } q\ r\texttt{)})$	Monotonisuus
$\texttt{(ref: (ref: } p\ q\texttt{) } r\texttt{)} \mapsto \texttt{(ref: } p\ \texttt{(ref: } q\ r\texttt{))}$	
$\texttt{(ref: } p\ \texttt{(ref: } q\ r\texttt{))} \mapsto \texttt{(ref: (ref: } p\ q\texttt{) } r\texttt{)}$	Assosiatiivisuus
Jos g on globaali nimi, $\texttt{(ref: } p\ g\texttt{)} \mapsto g$	Globaalisuus
Jos n ja m ovat paikallisia nimiä,	
(p väittää, että $(n \mapsto r)$) $\Rightarrow$ $(\texttt{(ref: } p\ n\texttt{)} \mapsto \texttt{(ref: } p\ r\texttt{))}$	Linkitys
$(p \mapsto q) \Rightarrow$ (p väittää, että s) $\rightarrow$ (q väittää, että s))	Agenttirelaatio

SDSI:ssä ei sulkulausekkeiden järjestyksellä ole merkitystä. Tämä on kirjattu assosiatiivisuusaksioomassa.

Refleksiivisyys on helppo ymmärtää: kukin oman nimiavaruutemme taho voi puhua omasta puolestaan. Transitiivisuus on myös varsin selkeä. Jos q:lla on oikeus puhua r:n puolesta ja p:llä q:n puolesta, p:llä on myös oikeus puhua r:n puolesta. Toisaalta, vaikka sekä q ja p olisivat valtuutettuja puhumaan r:n puolesta, ei se kerro mitään p:n ja q:n keskinäisestä suhteesta.

Monotonisuudella tarkoitetaan, että jos p:llä on oikeus puhua q:n puolesta (eli ne samaistetaan toisiinsa), samaistetaan toisiinsa myös saman nimiset nimet kummankin nimiavaruudessa.

Globaalisuus merkitsee, että globaali nimi on globaali kaikissa nimiavaruuksissa. Esimerkiksi `Valtion` nimiavaruudessa olevalla julkisella avaimella K on aina oikeus puhua K:n puolesta, koska julkinen avain on globaalisti uniikki.

K:n voi myös siirtää mihin tahansa muuhun nimiavaruuteen ja sen uniikkius säilyy.

Linkitys tarkoittaa, että p:hen luotetaan sen oman nimiavaruuden käsittelyssä; jos nimiavaruutemme p väittää, että sen omassa nimiavaruudessa $n \mapsto r$, voimme linkittää saman oletuksen omaan nimiavaruuteemme p:n kautta.

Viimeinen aksioomista, agenttirelaatio, määrittelee aiemmin mainitsemamme $\mapsto$ -merkinnän: jos $p \mapsto q$, p:llä on oikeus puhua (eli tehdä väitteitä) q:n puolesta.

Esimerkki: SPKI/SDSI. IETF (Internet Engineering Task Force) aloitti SPKI-varmenteiden spesifioinnin [EFL+98a], [Ell98], [EFL+98b], [EFL+98c] vuoden 1997 alussa. Myöhemmin aloitteeseen yhdistettiin SDSI [RL96], joka oli toinen samankaltaisista lähtökohdista lähtenyt projekti. Tässä käytämme tämänhetkisestä versiosta nimeä SPKI/SDSI [Riv97]. SPKI-pohjaisesta politiikkamanagerista on luotu prototyyppi Teknillisessä korkeakoulussa [LN98].

Poiketen PGP:stä ja X.509:stä, joissa varmenteeseen kuuluu myös digitaalinen allekirjoitus, SPKI/SDSI:ssä allekirjoitus ja varmenne on erotettu omiksi objekteikseen. Molemmat siirretään osana jotakin protokollaa. Muita tavallisia objektityyppejä SPKI/SDSI:ssä ovat julkiset avaimet, kahden tyyppiset varmennesulkulistat (näistä myöhemmin), voimassaoloajan muuttamispyynnöt ja operaatiopyynnöt. Viimeksi mainituilla ohjataan vastaanottajaa optimaaliseen varmennekäsittelyyn esimerkiksi ilmaisemalla, että kyseinen varmenne kannattaa ajaa tiivistealgoritmin läpi ja indeksoida tiivisteensä mukaan myöhempää viittausta varten. Objektityyppejä on olemassa myös salaisille avaimille (joskin niitä harvemmin siirrellään) sekä pääsylistoille.

SPKI/SDSI käyttää *S-lauseita* (S-expression) varmenteidensa kuvaamiseen. S-lause on Lispiltä tai Schemeltä näyttävä sulkulauseke, jossa tyhjiä listoja ei sallita ja jokaisen listan ensimmäinen alkio ilmaisee listan tyypin. Aloitteen nimen "simple" nimittäin alun perin tarkoitti myös järjestelmää, jossa varmenteita voisi lukea ja käsitellä ilman erikoistyökaluja. Sittemmin S-lauseille on tehty kompaktimpia ja hankalammin luettavampia siirtoformaatteja lähinnä siirtotehokkuussyistä.

SPKI/SDSI-varmenteissa turvallisuuden kannalta tärkeät elementit muodostavat *viisikkoja* (5-tuples), joiden jäsenet ovat varmenteen myöntäjä (allekirjoittaja), subjekti (eli valtuuttaja, jota varmennetaan), delegointilupa (saako varmennettu valtuuttaja propagoida oikeutensa eteenpäin), attribuutit, joita subjektilla on (oikeuksia, nimiä ja niin edelleen) sekä varmenteen voimassaoloaika. Esimerkiksi seuraavassa voisi olla opintotoimiston myöntämä

varmenne Teemu Teekkarille. Delegointilupaa ei ole, muutoin varmenteessa olisi myös kenttä `(propagate)`.

```
(cert
  (issuer <opintotoimiston-avain>)
  (subject <teemun-avain>)
  (not-before 1998-09-30)
  (not-after 2005-05-31)
  (tag (name ''Teemu Teekkari'')
       (email teemu@tut.hut.lut.fi)
       (student-id 1234567)))
```

Itse `tag`-kentän eli attribuuttikentän eri attribuuttien nimiä ei ole määrätty — ne ovat sovelluskohtaisia. Joitakin yleisimpiä on tosin kuvattu ja suositeltu. Edellisessä esimerkissä globaali nimi `<opintotoimiston-avain>` *väittää, että* `Teemu Teekkari` $\mapsto$ `<teemun-avain>` 30.9.1998-31.5.2005 välisenä aikana ja että avaimeen liittyvät myös tiedot nimestä, sähköpostiosoitteesta ja opiskelijanumerosta. "Kaikki oikeudet" kuvataan rakenteella `(tag (*))`.

Viisikkoredusointi. Oletetaan, että todentaja on kanonisoinut kaikki tarpeelliset varmenteiden nimet, jolloin hänellä on viisikkoja (I, S, D, A, V), jossa I on valtuuttajan globaali nimi (esimerkiksi siis julkinen avain) tai todentajaa itseään tarkoittava "`Self`", S subjektin eli varmenteen omistajan globaali nimi, D delegointilupa (`tosi` tai `vale`), A lista, jonka alkiot ovat alilistoja tai merkkijonoja sekä V voimassaoloaika (alku- ja loppuaika, tai palvelu, jolta avaimen voimassaolon voi tarkastaa). Varmenteiden luotettavuuden arvioinnissa tehdään *viisikkoredusointia* (5-tuple reduction), jossa pyritään saamaan aikaan viisikko $(\texttt{Self}, S, D, A, V)$. Tällainen viisikko kuvaa tilannetta, jossa todentaja itse uskoo varmenteen väitteeseen. Mikäli todentaja ei pysty enää redusoimaan viisikkoja, mutta ei ole päätynyt kelvolliseen tulokseen $(\texttt{Self}, S, D, A, V)$, jossa todistamisen ajan hetki $\texttt{nyt} \in V$, on redusoinnin tulos ja niin ikään luottamus määrittelemätön.

Osa viisikoista on peräisin todentajan omasta pääsylistasta. Se on yleensä ainoa paikka, jossa valtuuttajana on `Self`. Kaikkien viisikoiden ei välttämättä tarvitse olla SPKI/SDSI-varmenteita; muita varmenteita käytettäessä attribuutteja on vain tulkittava sopivasti. Usein myöskään oman pääsylistan viisikot eivät ole varmenteita siinä mielessä, että niitä ei ole allekirjoitettu (niiden turvallisuudesta pidetään huolta muulla tavalla). X.509-maailmassa

redusointi johtaa todennäköisesti tulokseen $(\texttt{CA}, S, D, A, V)$, jossa `CA` on varmenneviranomaisen julkinen avain, ja lopullinen luottamus rakennetaan sen tiedon perusteella, että todentaja hyväksyy kyseisen varmenneviranomaisen varmentamat avaimet (tämä voidaan tehdä *suosituksilla*, joista myöhemmin).

Perusredusointisääntö on

$$(I_1, S_1, D_1, A_1, V_1) + (I_2, S_2, D_2, A_2, V_2) \Rightarrow (I_1, S_2, D_2, A_1 \sqcap A_2, V_1 \cap V_2)$$

jos $S_1 = I_2$ ja $D_1 = \texttt{tosi}$. Operaatio "$\sqcap$" poikkeaa tavallisesta leikkauksesta: tulosjoukossa on elementti, jos molemmissa operandeissa on elementti. Jos operandin A_1 elementit löytyvät kaikki myös A_2:ta, on tulosjoukko A_2. Tämä siksi, että A_2:n ylimääräiset elementit tulkitaan tällöin A_1:n lisärajoitteiksi. Redusoinnissa jompi kumpi voimassaoloaika voi olla myös erityisarvo `nyt`, joka tulkitaan tarkasteluhetken aikaleimaksi. Tällöin leikkauksen tulosjoukko on joko `nyt` tai $\emptyset$.

3.6 Varmenteiden käytöstä poistaminen

Varmenteiden peruminen. Avainpari ei ole ikuinen. Tietokoneiden laskentateho ja edistysaskeleet kryptoanalyysissä saattavat muuttaa ennen turvallisena pidetyn avaimen turvattomaksi. Mitä useampi viesti avaimella on salattu, sitä suuremmaksi nousee mahdollisen tunnettuun (tai arvattuun) selväkieleen perustuvan hyökkäyksen onnistumismahdollisuus. Pitkä aikaväli antaa hyökkääjälle myös enemmän mahdollisuuksia yrittää saada salainen avain avaimen omistajan laitteistosta. Myös varmenteeseen sidotut attribuutit saattavat vanheta tai muuttua.

Näiden kaikkien syiden vuoksi varmenteen *perumiseen* (revocation) liittyvä ongelmakenttä on hyvin tärkeä. Perumiseen liittyvät yksityiskohdat vaihtelevat käytetyn avaintenhallintainfrastruktuurin mukaan, koska perumiseen tarvittava luottamus on vastaava kuin varmennukseenkin tarvittava.

Varmenteiden voimassaolokriteerien suunnitteluperiaatteita. [Riv98] listaa tiettyjä periaatteita, joiden avulla varmennejärjestelmään voidaan suunnitella hyvä voimassaolon valvontajärjestelmä. Ensimmäinen vaatimus on, että todentajan tulee pystyä asettamaan yläraja voimassaoloajalle. Jos todentaja vaatii korkeintaan päivän vanhan varmenteen, halutun palvelun saamiseksi ei ole muuta vaihtoehtoa kuin uuden varmenteen hankinta. Toisaalta valtuuttajan on pystyttävä antamaan transaktion

aikana kaikki voimassaolon tarkistamiseen tarvittava tieto. Tämä helpottaa todentajan toimintaa varsinkin jos todentaja on kuormitettu palvelin. Muita hyviä suunnittelukriteerejä ovat sulkulistojen välttäminen ja uusien varmenteiden luonti usein.

Sulkulistat. Ennen kuin kauppojen kassakoneisiin yleisesti liitettiin elektronisia kortinlukijoita, jotka tarkistavat kortin kelpoisuuden automaattisesti, useassa myymälässä otettiin pankkikorttimaksuja vastaan nk. mankelilla. Kun pankkikortti katoaa, vastuu sen käytöstä siirtyy pankille välittömästi katoamisilmoituksen yhteydessä. Mikäli kortti ilmoitettiin samalla varastetuksi, pankki lisäsi kortin numeron mustalle listalle. Mustaa listaa jaettiin säännöllisin väliajoin liikkeisiin. Korttia hyväksyvän kauppiaan olisi ennen mankelointia teoriassa pitänyt tarkistaa, ettei kortin numeroa ollut mustalla listalla. Oman korttini numeroa ei ole koskaan tarkastettu käsin, eikä listan selaaminen tuntunut kovin kätevältä myöskään tiskin toisella puolella seistessäni. Listat vanhenivat myös nopeasti, eikä varastettu kortti tietenkään päätynyt listalle kuin vähintään viikon viiveellä (ja varas tuskin odotti kuukautta ennen kuin hän alkoi käyttää korttia).

Korttinumeroiden musta lista on täysin analoginen varmenneinfrastruktuurien *sulkulistojen* (certificate revocation list, CRL) käsitteen kanssa. Jos varmenneviranomaisen myöntämä varmenne ei jostakin syystä enää ole käyttökelpoinen, varmenneviranomainen allekirjoittaa omalla avaimellaan tiedotteen tästä ja pyrkii levittämään sitä mahdollisimman laajalle. Varmenneinfrastruktuurin kaikkien osien pitäisi säännöllisesti (ideaalitapauksessa ennen jokaista todistusta) noutaa uusin sulkulista tai sen päivitys, nk. delta-CRL, käydä se läpi, tutkia kaikkien varmenteidensa voimassaoloaika ja mielellään vielä tallentaa CRL vastaisen varalle. Tuskin mikään nykyinen tuote toteuttaa tätä kunnollisesti.

PGP:ssä varmenteen voi perua vain salaisen avaimen omistaja eli varmenteen subjekti. Ongelmat ovat kuitenkin samat, elleivät vielä pahemmat, koska PGP:ssä salaisen avaimen hävittyä varmennetta ei voi perua. PGP:n osalta tilannetta voi osittain korjata luomalla tiedotteen avaimen perumisesta (*itsemurhaviestin* (suicide note), kuten [Riv98] sitä nimittää) tulevaisuuden käyttöä varten. Julkinen avain rekisteröidään *itsemurhaviranomaiselle* (suicide bureau), joka tarkkailee verkkoliikennettä etsien kyseisen avaimen itsemurhaviestejä. Mikäli se ei ole nähnyt kyseisen avaimen itsemurhaviestiä, voi itsemurhaviranomainen julkistaa *terveystodistuksen* (certificate of health), jossa se toteaa, että on hyvin luultavaa, että avain on vielä käyttökelpoinen. Itsemurhaviranomainen voi myös pitää itsemurhaviestiä tallessa luotetun kolmannen osapuolen tavoin.

Varmenteen elinaika. Ehdottomasti tärkein perumisen muoto on jokaiseen varmenteeseen kirjattu elinaika. Tämän vuoksi elinaika on yksi viidestä SPKI/SDSI-varmenteen luottamuksen arvioinnissa käytetystä parametrista. Erityishuomiona voimme todeta, että perinteisessä PGP-varmenteessa elinaikaa ei ole. Tämä on jo nyt aiheuttanut ongelmia tilanteissa, joissa avainparin salainen avain on päässyt hukkumaan, mutta avain pysyy voimassa "ikuisesti".

On ehdotettu [Ell96b], [Riv98] koko sulkulistakäsitteen hylkäämistä ja sen sijaan huomion kiinnittämistä voimassaoloaikaan. Tällöin kaikkien varmenteita jakelevien järjestelmien turvallisuusarviointien olisi pakko olettaa, että varmenne on *joka tapauksessa* voimassa sen alkuperäisen voimassaoloajan. Jos tämä ei turvallisuuspolitiikan mielestä ole hyväksyttävää, on politiikkaa korjattava tai on käytettävä lyhyempiä voimassaoloaikoja tai kertakäyttöisiä varmenteita.

Tilannetta voidaan helpottaa antamalla varmenteelle pitkä voimassaoloaika, mutta pakottamalla lyhyemmällä aikavälillä tapahtuvia voimassaolotarkistuksia. Mikäli voimassaolotarkistusta ei ole tehty ajoissa, ei varmenne ole voimassa. Tämä eroaa sulkulistan päivityksestä siten, että sulkulistan tapauksessa varmenne on oletusarvoisesti voimassa ellei sitä ole peruttu — voimassaolotarkistusten tapauksessa varmenne on oletusarvoisesti epäkelpo.

Elinajasta voidaan myös tehdä kolmiportainen: "avain on voimassa ajan hetkestä T_1 eteenpäin, takuuvarmasti T_2 asti ja jopa T_3 asti, jos teet voimassaolotarkistuksen Δt väliajoin".

Segmentoidut sulkulistat. Sulkulistojen tehokkuutta voidaan parantaa myös segmentoimalla ne eri käyttötarkoitusten tai ryhmien mukaan. Tämä vaatii varmenteen myöntövaiheessa jonkinlaisten varmennepuun sisäisten ryhmien luomista, mutta saattaa helpottaa tilannetta kun sulkulistoja pidetään yllä. Ryhmien tulee olla toisistaan riippumattomia niin, että ne eivät tarvitse toistensa alueella olevia varmenteita. Varmennepuun osa, jossa varmenteita poistetaan käytöstä suuria määriä, ei tällöin aiheuta turhaa kuormaa muille samaa varmennepuuta käyttäville. Yksi toteutusmekanismi on esimerkiksi lisätä uniikki kriittinen X.509v3-laajennos jokaiseen ryhmään, joka aiheuttaa varmenteiden epäyhteensopivuuden ryhmien välillä, mutta silti säilytetään yhteinen luottamusjuuri. Koska voidaan olla varmoja, että eri ryhmät eivät käytä toistensa varmenteita, myöskään sulkulistoissa ei tarvitse listata kuin kyseisen ryhmän omia varmenteita.

Rajoitettu käyttötarkoitus. Suuri osa avainten riskeistä voidaan myös

välttää luomalla joka tarkoitusta varten oma avain. X.509:n kaltainen järjestelmä on tällaiseen toimintaan hieman jäykkä, mutta SPKI/SDSI rooleineen lähtee juuri tästä olettamuksesta. Esimerkiksi sähköpostin allekirjoitusavaimen paljastuminen ei tällöin romuta sähköpostin salausavaimen turvallisuutta, puhumattakaan esimerkiksi mahdollisesti eri sovelluksessa käytetystä luottokorttinumeron salausavaimesta tai kulunvalvonnan avaimesta.

Rajatuilla käyttötarkoituksilla saavutetaan myös huomattavia etuja nykytilanteessa, jossa kryptografiaa pidetään useissa tapauksissa *kaksikäyttötuotteena* (dual-use product) ja sen vientiä ja käyttöä valvotaan maailmanmarkkinoilla. Valmistajien on helpompi saada lupia käyttää vahvaa salausta pelkkään digitaaliseen allekirjoitukseen kuin salaukseen. Mikäli kumpaankin käytettäisiin samaa avainta, johtaisi se heikkojen avainten tai algoritmien käyttöön kaikissa sovelluksissa.

3.7 Luotetut kolmannet osapuolet

Useissa maissa on esitetty viranomaisten taholta vaatimuksia salaisten *avainten luovutuksesta* (key escrow) viranomaisten tai *luotetun kolmannen osapuolen* (trusted third party, TTP) haltuun. Vaikka tämä on symmetrisen kryptografian maailmassa avaintenhallinnallisesti järkevää, ei se puolusta paikkaansa asymmetrisiä menetelmiä käytettäessä. Aihe on avaintenhallintainfrastruktuurien kannalta merkittävä, koska se liittyy suoraan siihen, voidaanko avaimen alkuperään enää luottaa varmenneinfrastruktuurin pohjalta lainkaan.

Yleisesti avainten luovutusta perustellaan tilanteella, jossa esimerkiksi yrityksen työntekijä lähtee pois yrityksestä tuhoten salaisen avaimensa. Yrityksen on päästävä käsiksi salattuun dataan. Tätä varten tapahtuvasta avainten-luovutuksesta käytetään usein termiä *avainten takaisinsaantijärjestely* (key recovery). Jotkut yritykset valvovat myös työntekijöidensä ulos lähettämiä tiedostoja teollisuusvakoilun varalta.

Niin helpoksi kuin tilanne valvonnan kannalta näennäisesti muodostuukin tuhoaa avainten luovutus kolmannelle osapuolelle sen perusperiaatteen, jolle avaintenhallintajärjestelmät perustuvat. Jos salainen avain ei enää ole ainoastaan avainparin omistajan hallussa, ei digitaalisten allekirjoitusten todennus voi enää olla sataprosenttisen varma. Riippuen kolmantena osapuolena toimivasta tahosta järjestelmään lisätty epävarmuustekijä saattaa olla turvallisuuspolitiikan rajoissa tai sitten ei. On hankalaa määritellä, milloin riski on hyväksyttävä ja milloin se ei ole.

Yhdysvaltain kansallisen turvallisuuspalvelun NSA:n raportti [NSA98] nimeää suurimmaksi riskiksi luotetun kolmannen osapuolen palveluksessa työskentelevän henkilön, joka saattaa vuotaa salaisia avaimia muille tahoille. Tämän hän tekee joko omaksi hyödykseen tai esimerkiksi kiristyksen uhrina. TTP-järjestelmässä, jossa käsitellään esimerkiksi kansallisen henkilökortin salaisia avaimia, salaisen avaimen joutuminen vääriin käsiin antaa käytännössä asiaankuulumattomalle osapuolelle jonkin toisen henkilön sähköisen identiteetin. Rikolliset olisivat varmasti erittäin kiinnostuneita tällaisista *alter egoista*. Kun TTP-järjestelmä kasvaa niin suureksi, että sen ylläpitoon tarvitaan useita henkilöitä, kasvaa riskien määrä samassa suhteessa [AAB+98].

Luotettujen kolmansien osapuolien käyttöön on ehdotettu tiettyjä parannuksia, kuten esimerkiksi salaisen avaimen jakoa salaisuuksienjakoprotokollan (secret sharing) [Sal96, ss. 187–190], [Sch96, ss. 528–531] avulla. Tällöin yhdellä TTP:n työntekijällä ei olisi pääsyä salaiseen avaimeen vaan siihen tarvittaisiin useita henkilöitä ja esimerkiksi kaksi eri TTP:tä. Tämä ehkäisee avainten väärinkäyttöä ulkopuolisten tahojen toimesta, mutta ei tarjoa edelleenkään suojaa sisäpuolelta tapahtuvaan väärinkäyttöön. Historia tuntee lukuisia tapauksia, joissa esimerkiksi poliisi on väärentänyt todistusaineistoa. Mikäli digitaalinen allekirjoitus on oikeudessa käypä, mahdollistaa tämä sähköisen todistusaineiston väärentämisen ei-toivottujen henkilöiden tuomitsemiseksi. Yleisesti voidaan todeta, että avaintenhallintainfrastruktuuri, jossa salaisten avainten luovutus kolmannelle osapuolelle on mukana, on käyttökelvoton mihinkään muuhun kuin sellaisiin järjestelmiin, joiden turvallisuuteen ei muutenkaan voida luottaa.

4 Luottamuksenhallinta

Luottamuksenhallintaongelma. Virusten ja troijalaisten maailmassa tyypillinen ongelma on varmistua riittävässä määrin siitä, että verkosta haettu ohjelma tai muu objekti on puhdas ja hyvin käyttäytyvä. Sataprosenttisen varmuuden saaminen ohjelman toiminnasta on hankalaa, koska ohjelman formaali todistaminen oikeaksi on vähänkään monimutkaisemmalle ohjelmalle varsin työlästä. Tähän on tarjottu osaratkaisuksi muun muassa nk. *itsetodistavaa koodia* (proof-carrying code, PCC) [FL97], jossa ohjelman on itse todistettava järjestelmälle olevansa hyvin käyttäytyvä [CFL+98].

Muita usein toistuvia ongelmia ovat erityisesti pääsynvalvonnassa, sähköisessä kaupankäynnissä ja lähdetodennuksessa esiintyvät kysymykset ”Onko

tämän pyynnön allekirjoittanut avain x oikeutettu pyyntöönsä?". Kysymys saattaa järjestelmästä riippuen muuntua esimerkiksi muotoon "Kuuluuko avain x henkilölle A?".

Sataprosenttisen varmuuden saavuttaminen näissä tapauksissa on yleensä käytännössä mahdotonta. Avainasemassa on tällöin järjestelmän politiikka: mikä on se luottamustaso, jolla voimme toimia samoin, kuin jos varmuus olisi täydellinen, ja kuinka suuren riskin tällöin hyväksymme? Riskien arviointi ja politiikan muodostaminen on sovellus- ja järjestelmäkohtainen ongelma, eikä siihen puututa tässä. Jos oletamme, että sopiva politiikka on olemassa, luottamuksen määrittely on mahdollisesti varsin monimutkainen sarja erilaisten väitteiden ja riippuvuussuhteiden (politiikkojen, valtuutuksien sekä luottamussuhteiden) tarkastelua. Ongelmaa kutsutaan *luottamuksenhallintaongelmaksi* (trust management problem). Luottamuksenhallintaongelman ratkaisu on "todistus" siitä, että pyyntö oli oikeutettu — tai todistuksen puute, jolloin oikeutuksesta ei ole tietoa [BFS98].

Todistus ei ole todistus sanan perinteisessä mielessä: luottamuksenhallintajärjestelmän todistuksessa on aina mukana riskitekijä, pieni mahdollisuus sille, että todistus ei pidä paikkaansa. Riskejä ovat muun muassa todistuksen ajallinen kesto ja todistukseen käytettävän ohjelmiston ja laitteiston luotettavuus. Järjestelmän suunnittelijan vastuulla on kehittää sellainen järjestelmän turvallisuuspolitiikka ja riskien hallintamenetelmä, joka tekee riskinoton hyväksyttäväksi.

Politiikka. Jokaisessa järjestelmässä on jokin säännöstö, joka sanelee sen, mikä on mahdollista ja sallittua. Tätä kutsutaan nimellä *politiikka* (policy). Se on sovelluskohtainen. Tietojärjestelmän politiikka voisi esimerkiksi todeta "ajan ohjelman O vain, jos käyttäjällä on oikea salainen avain". Pankin politiikka voisi olla "henkilö, jolla on oikeus kirjoittaa yhdistyksen nimi, voi nostaa yhdistyksen tililtä rahaa kerran viikossa. Jos nostojen määrä viikon aikana on yli 5000 mk, on nimenkirjoitusoikeuden omaavia henkilöitä kuitenkin oltava paikalla vähintään k kappaletta".

Aiemmin on kuvattu muutamia avaintenhallintamekanismeja. Kuten todettiin, PGP käyttää luottamusverkkoa, jossa avaintenhaltijat pystyvät varmentamaan toistensa avaimia. Allekirjoituksen alkuperälle ja luotettavuudelle voidaan antaa jonkinasteinen laatuleima, jonka perusteella luottamusketjun arvioinnin jälkeen voidaan päätellä jotakin avaimen sidonnasta tiettyyn nimeen.

PGP, X.509 ja muut henkilövarmennejärjestelmät kuitenkin lisäävät yhden

kerroksen epäsuoruutta esimerkkipolitiikkoihimme. Tietojärjestelmän on ylläpidettävä pääsylistaa, jossa ovat henkilöiden nimet ja heidän oikeutensa. Pankin on pidettävä kirjaa jokaisen yhdistyksen auktorisoiduista henkilöistä.

Valtuutus. Taho haluaa todistaa, että hänellä on oikeus tehdä järjestelmän politiikan mukainen teko. Tähän hän käyttää *valtuutusta* (credential). Valtuutus saattaa olla esimerkiksi avainparin salainen puolisko, jolla salaisen avaimen omistaja voi vastata hänelle lähetettyyn haasteeseen.

Kyseessä ei tietenkään aina tarvitse olla salainen avain: oikeassa elämässä henkilöllisyystodistus toimii usein valtuutuksena. Koska se kuitenkin on sidottu henkilön nimeen, on valtuutus epäsuora. Tavallisen lukon avain on myös valtuutus (avata ovi); sitä ei ole sidottu henkilöön vaan sitä voi käyttää kuka tahansa.

4.1 Yleistetty luottamuksenhallinta: PolicyMaker

Avoimen tietojärjestelmän luottamuksenhallinnan toteutuksessa kohdataan tyypillisesti seuraavia ongelmia:

- Politiikka on sovelluskohtainen. Politiikka saattaa jopa vaihdella ajan tai yhteydenottotavan mukaan. Luottamuksenhallintajärjestelmän on skaalauduttava moniin erilaisiin politiikkoihin.
- Jokaisella järjestelmällä on omia erityispiirteitä, joten luottamuksenhallinta on rakennettava aina uudelleen.
- X.509 ja PGP ovat kaupallisessa ja harrastemaailmassa *de facto*-standardeja, eikä näiden suosittuvuutta voi nopeasti pienentää attribuuttivarmenteiden hyväksi.
- Riippuen valtuutuksia tarjoavasta tahosta, jopa samankaltaisessa avaintenhallintainfrastruktuurissa voi olla erityyppisiä varmenteita — esimerkiksi X.509 ja WTLS-varmenteet.[13] Järjestelmän on oltava riippumaton varmennesyntaksista.
- Luottamuksenhallinta on monimutkainen ongelma ja hyvin määritellyn luottamuksenhallintajärjestelmän olemassaolo on eduksi.

[13]WTLS, Wireless Transport Layer Security, on langattomiin laitteisiin hitaalle tiedonsiirtoyhteydelle suunniteltu turvaprotokolla [WAP98]. Varmenne muistuttaa ajatusmaailmaltaan X.509:ää, mutta on optimoitu koon pienentämiseksi.

PolicyMaker. Näihin ongelmiin [BFL96] esittää PolicyMaker-nimistä järjestelmää. PolicyMakerin idea on käyttökelpoinen ymmärtää, joten se käydään tässä läpi varsin tarkasti. PolicyMaker on "musta laatikko", joka on täysin irrallinen sovellusalueesta. Se ei itse tarkista allekirjoituksia eikä suorita kryptografisia operaatioita. Sovellus itse hankkii varmenteet, tarkistaa allekirjoitukset ja varmenteiden voimassaolon. Politiikka, varmenteet ja pyyntö halutusta toiminnasta annetaan PolicyMakerille. PolicyMaker palauttaa arvon "toiminta hyväksytty", "toimintaa ei hyväksytty" tai "toiminta hyväksytty, jos lisäehto L on tosi".

PolicyMaker ei itse ota kantaa siihen, mikä *toimintapyyntö* (action string) on kyseessä. Toimintapyyntöjen onnistumista tarkastellaan *suodattimilla* (filter), jotka on rakennettu niin, että ne ovat sovelluskohtaisia.

Väitteet. Valtuuttaja, esimerkiksi avainpari, käytännössä siihen sidottu taho) voi esittää *väitteen*. Väite tarkoittaa, että väitteen tehnyt valtuuttaja, nimeltään *lähde*, luottaa joihinkin tiettyihin muihin valtuuttajiin — käytännössä julkisten avainten aitouteen. Väitteellä voi olla *ehto*, jonka on oltava tosi ennen kuin väite pitää paikkansa. Väitteen kohteena saattaa myös yksittäisen avaimen sijasta olla monimutkaisempi joukko kuten "vähintään 5 kappaletta seuraavista 10 avaimesta". Väitteelle käytetään merkintää (s, v), jossa s on suodatin ja v on valtuuttaja.

Ehto on suodatin, joka ottaa syötteekseen toimintapyynnön ja palauttaa "kyllä", "ei" tai "lisäehdolla L".

Politiikka kuvataan sarjana *politiikkaväitteitä*, joissa lähde on `policy`. Jos lähde on jokin muu kuin `policy`, on kyseessä *allekirjoitettu väite*, ja siihen liittyy lähteen luoma digitaalinen allekirjoitus. Allekirjoitettu väite on siis käsitteellisesti sama asia kuin varmenne.

Suodatinkieli. Koska suodattimet ovat sovelluskohtaisia ja itse PolicyMaker ei niiden toimintaa määrää, on hyvin luonnollista tehdä niistä ohjelmia. Suodatinohjelmaa ajetaan tilassa, jonka resursseilla on katto (syötteen ja tulostuksen pituuden sekä muistin ja prosessoritehon kulutuksen yläraja) eivätkä ne saa päästä käsiksi koneen muihin resursseihin tai muihin sovelluksiin. Mikäli ohjelma rikkoo rajoituksiaan, tulkitaan suodattimen automaattisesti palauttavan "ei". Tällä estetään palvelunriistohyökkäykset luottamuksenhallintajärjestelmää vastaan.

PolicyMakerin alkuperäisessä suunnitelmassa käytettiin AWKin [DR97] turvallista muunnelmaa, jonka nimi on AWKWARD. Käytännössä esimerkiksi Java [AG96] voisi olla erinomainen valinta suodattimien ohjelmointikieleksi,

koska Java-sovelmia ajetaan tyypillisesti resurssirajoitteisissa "hiekkalaatikoissa". Java on tavukoodiksi käännettynä myös suhteellisen nopeaa.

Suodatin saa syötteenään toimintapyynnön ja tietoja senhetkisestä *arviointiympäristöstä*. Tyypillisesti tässä arviointiympäristössä on ainakin aikaleima, sovelluksen nimi ja loput *väiteketjusta*, jossa kyseistä väitettä tarkastellaan.

Jos suodattimen on tarkistettava allekirjoituksia tai haettava esimerkiksi varmenteita verkosta, myös näiden tietojen oletetaan kuuluvan arviointiympäristöön. PolicyMakeria käyttävä sovellus voi joko tarkistaa allekirjoitukset valmiiksi tai sitten tehdä tarkistuksia PolicyMakerin pyynnöstä. Allekirjoitusten tarkistamiseen voidaan käyttää olemassaolevia järjestelmiä kuten PGP:tä. PolicyMaker ei sinänsä ota kantaa siihen, mitä avaintenhallintainfrastruktuuria tai minkä syntaksin mukaisia varmenteita käytetään. Myöhemmin mainittavissa esimerkeissä järjestelmänä on geneerinen `pubkey`.

Kyselyt. PolicyMakerilta voi tehdä *kyselyitä* (query). Kyselyssä PolicyMakerille annetaan yksi tai useampia valtuuttajia, jotka esittävät yhden toimintapyynnön.

Koska luottamus perustuu viime kädessä järjestelmän politiikkaan, PolicyMakerilla on oltava vähintään yksi väite, joka muodostaa politiikan (eli väite, jonka lähteenä on `policy`). Politiikkojen perusteella muodostetaan uskomus joidenkin toisten valtuuttajien aitouteen ja nämä taas voivat väittää jotakin käyttäen allekirjoitettuja väitteitä.

Noudattamistodistus. Kyselyn kulku on *noudattamistodistuksen* (proof of compliance, POC) etsimistä [BFS98]. Kyselyä kuvataan *kyselykolmikolla* (i, v_i, K_{ij}). PolicyMaker pitää yllä kyselykolmikkojen joukkoa. Tämä kyselykolmikkojen joukko syötetään tarkasteltavalle väitteelle, joka saattaa tuottaa uuden kyselykolmikon, joka lisätään joukkoon. i on kyselykolmikon tuottaneen väitteen järjestysnumero, v_i on toimintapyynnön hyväksynyt valtuuttaja ja K_{ij} on hyväksytty toimintapyyntö. On huomattava, että väitteelle syötetty kyselykolmikkojen joukko on aina vastauksena saadun kyselykolmikkojen joukon alijoukko, ts. väite voi lisätä joukkoon kyselykolmikkoja, mutta sieltä ei voida niitä poistaa.

Kysely alkaa kyselykolmikolla (Λ, Λ, K), joka sisältää alkuperäisen toimintapyynnön K, jota kukaan ei ole vielä hyväksynyt. PolicyMaker pyrkii löytämään kyselykolmikon, joka on muotoa $(0, \texttt{policy}, K')$. Jos $K = K'$, tulos on "toimintapyyntö K noudattaa todistettavasti järjestelmän politiikkaa". K' voi myös sisältää lisävaatimuksia, jolloin tulos tulkitaan "lisäehdoin varustettu toimintapyyntö K' noudattaa todistettavasti järjestelmän politiikkaa". Jos

K':n vaatimukset voidaan tyydyttää, myös K:n vaatimukset voidaan tyydyttää.

Yllä kuvatun noudattamistodistuksen, POCin, yleinen muoto on NP-kova ongelma [BFS98]. PolicyMakerin maailmassa ongelmalle asetetaan tiettyjä rajoitteita, jotta on mahdollista tehdä rajoitetun POC-ongelman ratkaiseva polynomiaikainen algoritmi.

PolicyMakerin noudattamistodistus (LBMAPOC). PolicyMakerissa käytetään algoritmia, joka ratkaisee *paikallisesti rajatun, monotonisen ja autenttisen POC-ongelman* (locally bounded, monotonic, authentic proof of compliance, LBMAPOC).

Paikallisesti rajattu POC (LBPOC) olettaa, että jokaisen väitteen tarkasteluun kuluu korkeintaan $O(N^c)$ aikaa (N on väitteen syötteen pituus), väitteitä tarkastellaan enintään l kappaletta (vaikka niitä olisi enemmänkin), kukin väite voi tuottaa korkeintaan m erilaista toimintapyyntöä ja väitteen palauttaman kolmikon koko voi olla korkeintaan s.[14]

Monotoninen POC (MPOC) olettaa, että kaikille kolmikkojoukoille A ja B, $A \subseteq B$, jos ne annetaan syötteenä väitteelle (s_i, v_i) (merkitään $(s_i, v_i)(A)$), pätee $(s_i, v_i)(A) \subseteq (s_i, v_i)(B)$.

Autenttisuus tarkoittaa, että kukin väite (s_i, v_i) tuottaa ainoastaan kolmikoita (i, v_i, K_{ij}) eikä teeskentele olevansa jokin toinen väite i' ja tuota kolmikkoa $(i', v_i', K_{i'j})$, joka näyttäisi jonkin toisen väitteen tulosteelta.

Kaikilla yllämainituilla rajoitteilla lisätty POC on LBMAPOC.

PolicyMakerin LBMAPOC-ongelman ratkaiseva algoritmi. [BFS98] esittää LBMAPOC-ongelmaan seuraavan algoritmin. Algoritmi käy läpi väitteitä ja jos jokin väite rikkoo LBMAPOC-ongelman rajoitteita, se tunnistetaan ja kyseinen väite siirretään algoritmin ajon loppuajaksi jäähylle. `ViallinenVäite` palauttaa toden, jos väitteen ajon aikana törmättiin rajoitteisiin (väitteen tulkitaan tällöin palauttavan syötteensä ilman lisäyksiä). Vialliset väitteet poistetaan algoritmin loppuosan ajaksi tutkittavien väitteiden joukosta. Väitteen viallisuus saattaa siis olla piilevää: viallisuus ei ehkä tule esille kuin tietyllä syötteellä.

Algoritmin pseudokoodi on esitetty kuvassa 10. Rivillä 2 asetetaan tehtäväkysely ensimmäiseksi kyselykolmikoksi ja rivillä 3 viallisten väitteiden joukko

[14] Kolmikon "koko" viittaa siihen, että toimintapyyntö tulkitaan merkkijonoksi.

```
1   LBMAPOC (r, {(s_0, policy), (s_1, v_1), ..., (s_{n-1}, v_{n-1})}, c, m, s)
2       A ← {(Λ, Λ, K)}
3       V ← {}
4       For j ← 1 to mn
5           For i ← n − 1 to 0
6               If (s_i, v_i) ∉ V
7                   then A' ← (s_i, v_i)(A)
8               endif
9               If ViallinenVäite((s_i, v_i))
10                  then V ← V ∪ {(s_i, v_i)}
11                  else A ← A ∪ A'
12              endif
13          endfor
14      endfor
15      If ∃K' : K ⊆ K' ∧ (0, policy, K') ∈ A
16          then palauta(T, K')
17          else palauta(F)
18      endif
```

Kuva 10: LBMAPOC-ongelman ratkaiseva algoritmi

tyhjäksi. Rivillä 6 tutkitaan, onko väite merkitty aiemmin vialliseksi; jos ei, kysely ajetaan suodattimen läpi rivillä 7. Jos suodattimen ajon aikana kävi ilmi, että väite oli viallinen, se havaitaan rivillä 9 ja rivillä 10 väite lisätään viallisten väitteiden joukkoon, eikä väitettä tarkastella tulevilla kierroksilla. Muutoin hyväksytään suodattimen palauttama kyselykolmikko rivillä 11. Lopuksi tutkitaan, onnistuiko algoritmi löytämään kyselykolmikon, jossa `policy` hyväksyy kyselyn, joka sisältää alkuperäisen kyselyn K. Jos näin kävi, palautetaan T (hyväksy) sekä hyväksytty kysely K', muutoin F (hylkää).

Sisempi silmukka käy läpi väitejonoa lopusta alkuun päin ja ulompi silmukka toistaa sen mn kertaa. Koska väitteitä on n ja jokainen voi tuottaa korkeintaan m erilaista uutta kyselykolmikkoa, erilaisia kyselykolmikkojen joukkoja A_l voi olla korkeintaan mn kappaletta. Täten väiteketjun läpikäynti mn kertaa riittää.[15] Väitteitä tarkastellaan siis mn^2 kertaa, mikä ei kuitenkaan

[15] Tässä oletetaan, että jokainen väite palauttaa aina syötteelle x ja ajoympäristölle y

käytännön sovelluksissa ole nykyään kovin suuri määrä, koska varmenneketjut ovat lyhyitä ja niiden määrä on vähäinen (n pieni) ja X.509-varmenteiden lisäehdot ovat hyvin jäykkiä (m pieni).

Kun väite lisää joukkoon A uusia kyselykolmikkoja, väitteen suodatinta kutsutaan *kommentaattoriksi* (annotator).

PolicyMakerin käyttökelpoisuus. Verrattaessa PolicyMakerin ratkaisemia ongelmia edellä mainittuihin yleistetyn luottamuksenhallinnan ongelmiin, voidaan havaita sen täyttävän kaikki vaatimukset tavalla tai toisella. Sovellusriippumattomuus saavutetaan ohjelmoitavilla suodattimilla, olemassaolevia varmennehierarkioita voidaan käyttää, mikäli ne tulkitaan väitteiksi, attribuuttivarmenteisiin voidaan siirtyä varsin helposti, ja mikä tärkeintä, luottamuksenhallintajärjestelmä suorittaa aina sovellusriippumattoman arvioinnin ottamatta kantaa varmennettavan asian merkitykseen. Näin vastuu tulosten tulkitsemisesta ja varsinaisesta päätöksenteosta jää PolicyMakeria käyttävälle sovellukselle ja PolicyMakerin toteutus voidaan pitää puhtaana.

Esimerkki: PolicyMaker ohjelman aitouden varmennuksessa. Oletetaan laite, johon on mahdollista hakea uusia ohjelmia Internetistä. Ohjelmille annetaan järjestelmässä vapaa pääsy järjestelmään tallennettuihin tietoihin, jolloin pelko troijalaisen hevosen asentamisesta tulee aiheelliseksi. Jotta esimerkki pysyisi riittävän lyhyenä, pidetään väitteiden joukko pienenä.

Ongelmien välttämiseksi laitteen valmistaja on määritellyt ohjelmien siirrossa käytetyn formaatin sellaiseksi, että ohjelman tekijä käy siitä ilmi. Olkoon kyseessä esimerkiksi kenttä `Creator`, joka saa arvokseen tekijän nimen, esimerkiksi `''Creator=Acme Plc.''`.

Jotta kuka tahansa ei voi esiintyä Acmen nimissä, laitteen ROM-muistiin on tehtaalla tallennettu Acmen julkinen avain `0xe1ee7d00dcafebabe`[16] sekä politiikka, joka ilmoittaa, että kyseinen valtuuttaja liittyy Acmeen:

```
policy ASSERTS
  pubkey:''0xe1ee7d00dcafebabe''
  WHERE PREDICATE=regexp:''Creator=Acme Plc.''
```

saman vastauksen $x' = f(x, y)$. Mikäli vastauksen laskennassa on mukana esimerkiksi kellonaika, aika on pysäytetty algoritmin ajon ajaksi — muuttuvia suureita ei siis syötteessä ja ympäristössä ole.

[16] Merkitsemme 16-kantaiset luvut `0x`-etuliitteellä. Sen lisäksi, että esimerkeissä käytetyt avaimet on valittu niiden heksadesimaalilukuesityksen kauneuden perusteella, ne ovat turvallisuuden takaamiseksi auttamattoman lyhyitä. Todellisissa järjestelmissä avaimiin on myös yleensä liitetty tieto algoritmista, jonka kanssa niitä on tarkoitus käyttää.

Kun ohjelma saapuu laitteeseen, käy ilmi, että se on allekirjoitettu avaimella `0xe1ee7d00dcafebabe` — tämän sovellus tarkistaa ennen PolicyMakerin kutsumista — ja sen otsikkotiedoissa lukee `''Creator=Acme Plc.''`. PolicyMakerille tehdään seuraava kysely:

```
pubkey:''0xe1ee7d00dcafebabe''
  REQUESTS ''Creator=Acme Plc.''
```

eli sovellus haluaa tietää, onko laitteen politiikan mukaista hyväksyä kyseisen avaimen k allekirjoittamalta ohjelmapaketilta otsikkorivi, joka ilmoittaa sen tekijäksi Acmen.

Väitteiden joukossa on vain yksi väite, joka on politiikkaväite ($n = 1$). Tämä väite voi palauttaa kaikilla syötteillään vain yhden erilaisen kyselykolmikon ($m = 1$), joka on lisäehdoton hyväksyntä. Algoritmi tekee siis ainoastaan yhden testin.

Toimintapyyntö on `''Creator=Acme Plc.''` ja tämä annetaan AWKWARD-ohjelmalle s_1, joka on kaikessa yksinkertaisuudessaan `regexp:''Creator=Acme Plc.''`.[17] Väite palauttaa kyselykolmikon $(0, \texttt{policy}, \texttt{Creator} = \texttt{AcmePlc.})$, joka lisätään A:han.

Algoritmin lopussa testataan, löytyykö haluttua kyselykolmikkoa A:sta, ja koska se löytyi, PolicyMaker hyväksyy toimintapyynnön.

4.2 KeyNote-luottamuksenhallintajärjestelmä

PolicyMaker, vaikka se varmasti muuntuukin joustavasti eri tarkoituksiin, on kiistatta hieman raskaan oloinen. Erityisesti PolicyMakerin suodatinkieli on ajettavine virtuaalikoneineen raskas. Koska toimintapyynnöille ei ole määritelty syntaksia, on jokaisen PolicyMakeria käyttävän myös suodattimien kirjoituksen lisäksi määriteltävä toimintapyyntöjen sisältö ja merkitys. PolicyMakerin idea oli toimia "kytke-ja-käytä"-tyyppisenä ratkaisuna, mutta liika konfiguroitavuus toimii alkuperäistä ideaa vastaan.

KeyNote [BFK98a], [BFK98b] on luottamuksenhallinnan teorialtaan sama kuin PolicyMaker, mutta sen käyttöalue on rajatumpi. KeyNoten tarkoituksena on pitää suodattimien ja tehtäväpyyntöjen kuvaus yksinkertaisena

[17] Tämä ohjelma vertaa syötettään annettuun *säännölliseen lausekkeeseen* (regular expression) ja palauttaa totuusarvon vertailun onnistumisen mukaisesti.

ja toteutus helppona. Ensinnäkin se palauttaa aina vain tietyn paluuarvon (esimerkiksi hyväksytty/hylätty)[18] ilman PolicyMakerista tuttuja lisäehtoja, luotetut toiminnat ja toimintapyynnöt kuvataan attribuutti-arvopareina ja järjestelmä käyttää RFC 822:stä [Cro82] ja C-kielestä periytyvää syntaksia.

KeyNote on suunnattu olemassaolevien julkisen avainten infrastruktuurien luottamushallintakäyttöön, erityisesti esimerkiksi turvasähköpostisovelluksiin.

KeyNoten osat. Kuten PolicyMakerin AWKWARD-ohjelmia ajettiin virtuaalikoneessa, KeyNoten väitteitä tutkitaan *toimintaympäristössä* (action environment). Kun KeyNotelle annetaan toimintapyyntö, se kuvataan ympäristömuuttujan kaltaisina rakenteina (attribuutti-arvopareina) toimintaympäristöön. KeyNote palauttaa paluuarvon sekä *paluuympäristön* (write-back environment), jossa on mahdollisesti lisätietoja KeyNoten päätelmistä. Paluuympäristö ei kuitenkaan sisällä lisäehtoja, joten KeyNotea kutsunut sovellus voi käyttää palautettua paluuarvoa sellaisenaan. Paluuympäristö voi toisaalta antaa sovellukselle vihjeitä siitä, mikä aiheutti KeyNote-ajon epäonnistumisen.

KeyNoten väitteet vastaavat idealtaan PolicyMakerin väitteitä. Kuten kantaisässään, niitä on kahdenlaisia: politiikkaväitteitä ja allekirjoitettuja väitteitä. PolicyMakerista tuttuun tapaan ainakin yksi politiikkaväite on pakollinen KeyNote-ajon onnistumiseksi. KeyNoten väitteessä on kahdeksan kenttää. Väitteessä kerrotaan asia, jonka teko väitteellä valtuutetaan. Lisäksi väitteessä on digitaalinen allekirjoitus (paitsi politiikkaväitteissä) ja valtuutetun tahon tunniste tai julkinen avain sekä allekirjoituksen tehnyt julkinen avain (tai `policy`)). Näiden lisäksi väitteissä on versionumero, kommenttikenttä, toimintaympäristön alustustiedot (kiinteäarvoisia attribuutti-arvopareja, jotka lisätään toimintaympäristöön ennen väitteen tutkimista) ja tieto siitä, mitkä toimintaympäristön attribuutti-arvoparit palautetaan kutsuneelle sovellukselle paluuympäristössä.

KeyNote-ajo. Kun sovellus haluaa tutkia jonkin teon sallittavuutta, se asettaa toimintaympäristöönsä avaimet, joiden luotettavuutta tutkitaan, teon, jonka sallittavuutta tutkitaan, kontekstin, jossa KeyNote toimii (esim. `RFC822-EMAIL`, [Cro82]) ja kontekstin mukaan muita aloitustietoja. KeyNoten väitteet on rakennettu yksinkertaisella lähinnä vertailuoperaatioita

[18] Paluuarvoja voi olla mielivaltaisen monta, kunhan ne voidaan laittaa suuruusjärjestykseen ja niiden määrä rajataan KeyNoten käynnistyshetkellä. Alin arvo tarkoittaa vähiten luotettua (kaksiarvologiikassa epätosi) ja ylin arvo eniten luotettua (kaksiarvologiikassa tosi) paluuarvoa.

sisältävällä kielellä, joka venyy useimpiin käytännön tarpeisiin. Kontekstilla voidaan ilmoittaa KeyNotelle tiettyjä erityispiirteitä juuri tällä hetkellä käsillä olevasta tehtävästä.

KeyNote tutkii väitteitä yksi kerrallaan ja pyrkii rakentamaan graafin, jonka juuri on politiikkaväite ja josta kulkee allekirjoitettuja väitteitä sisältävä ketju johonkin niistä avaimista, joiden luotettavuutta tutkitaan. Perusidea on sama kuin PolicyMakerissa.

Paluuarvon saatuaan sovellus voi KeyNoten määritelmän mukaan tehdä sille mitä vain. KeyNote ei ota kantaa sovellusalueeseen (muuten kuin kontekstin muodossa) ja KeyNote ei voi määrätä, että sovellus noudattaisi järjestelmän turvallisuuspolitiikkaa. KeyNote, kuten PolicyMakerkin, on siis vain työkalu luottamuksen arviointiin, ei sen politiikan pakottamiseen.

4.3 Suositukset

Tähän asti on puhuttu pääasiassa kahdesta suureesta julkisten avainten hallinnassa: alkuperätodennuksesta, joka saadaan aikaan allekirjoituksilla ja varmenteilla, ja jonka tarkoitus on todistaa julkisen avaimen salaisen avaimen puoliskon haltija, sekä luottamuksesta siihen, onko avaimen haltijalla lupa tehdä jotain. [Mau96] lisää vielä yhden suureen: uskon siihen, että jokin taho toimii odotusten mukaisesti.

Luottamuksen on useimmiten ajateltu olevan joko keskusjohtoista (kaikilla sama luottamus) tai yksityistä, jolloin itse luottamussuhteen olemassaolokin saattaa olla arkaluontoista informaatiota. Yksityisyysnäkökohtien vuoksi on joskus käyttökelpoista ajatella, että luottamusta välittävät allekirjoitetut väitteet ovat luottamuksellisia. [Mau96] kutsuu näitä varmenteen kaltaisia mutta mahdollisesti luottamuksellisia väitteitä *suosituksiksi* (recommendation).

Suositukset ovat varmenteita monimutkaisempia, koska niihin on lisätty tietoa niiden välittämän luottamuksen ”tasosta”. Esimerkiksi kakkostason suositus voi suositella, että luotamme jonkin toisen tahon suositukseen. Kuten [Mau96] mainitsee, suosituksia ketjutettaessa luottamus suosituksien antajaan vähenee hyvin nopeasti ketjun pituuden kasvaessa. Myös suosituksien annon ajalla on merkitystä ja suosituksilla onkin niiden käytöstäpoiston kanssa samankaltainen ongelma-alue kuin varmenteilla.

Suosituksien suurin merkitys on luottamusjuuren luottamuksen määrittelyssä. Suositukset vetoavat usein ”inhimillisiin” luottamusta herättäviin asioihin ja niitä tulkitsemaan tarvitaan yleensä järjestelmän turvallisuuspolitiikan tunteva ihminen. Suositukset siis palvelevat lähinnä järjestelmän luottamushallinnan ”juurruttamista” (bootstrapping). Suosituksia voidaan myös käyttää paikkaamaan puuttuvia paloja varmenneketjusta, jos turvallisuuspolitiikka antaa myöten. Monet tahot, jotka nykyään esimerkiksi tarkkailevat suoramarkkinointia ja puolustavat kuluttajien oikeuksia, voivat löytää luonnollisen ekolokeron Internet-maailmasta suosituksien myöntäjinä. [JY98] Näillä tahoilla on pitkällä aikavälillä hankittua luottamusta ihmisten mielissä.

Tulevaisuudessa suosituksia voidaan mahdollisesti arvioida tekoälysovelluksilla ja asiantuntijajärjestelmillä. Tällöin inhimilliselle epävarmuudelle annetaan todennäköisyysarvioita ja esimerkiksi luottamushallintajärjestelmän tulos saattaa olla ”hyväksytään, todennäköisyydellä p”. Järjestelmän turvallisuuspolitiikka määrää rajan, jota todennäköisemmät tulokset sitten lopulta hyväksytään.

Kollektiiviset suositukset. Yksittäisten suosituksia myöntävien tahojen lisäksi suuri joukko ihmisiä (tai ohjelmia) voi myös myöntää suosituksia kollektiivisesti. Suosituksen kelvollisuus riippuisi tällöin sekä suositusta puolustavien tahojen määrästä ja ajasta, joka suosituksen antamisesta on kulunut. Jos suositusten paikkansapitävyystodennäköisyyttä arvioi jokin ohjelma, se voi painottaa suosituksia mm. sen mukaan, kuinka tunnollisesti eri tahot ovat noudattaneet kommunikointiprotokollaa, kuinka nopea vasteaika eri tahoilla on, miten vahvaa todennusta tahot käyttävät, sekä muiden mitattavissa olevien suureiden perusteella. Mikäli jokin taho havaitsee suosituksen olevan kelvoton, hälytys tästä voi propagoitua suosituksen kaltaisena tiedotteena koko järjestelmään, jolloin se aiheuttaa paikkansapitävyystodennäköisyyksien laskun kaikkialla (sitä nopeammin, mitä useampi järjestelmän taho julkistaa hälytyksen omana suosituksenaan).

Kollektiivinen suositus voi muodostua *kokemuksista* (experience) [BBK94]. Kokemuksia on sekä myönteisiä että kielteisiä. Mitä enemmän myönteisiä kokemuksia jokin taho saa kuulla jostakin tahosta, sitä luotettavampi suositus tälle voidaan rakentaa. Kokemukset luokitellaan ja suosituksen hyvyyteen vaikuttaa vain sellainen kokemus, jonka luokka on relevantti sillä sovellusalueella, jossa luottamusta tarvitaan. Esimerkiksi digitaalisissa allekirjoituksissa [BBK94] nimeää luokiksi avainten generoinnin, kyvyn assosioida avaimen ja avaimen omistajan toisiinsa, salaisuuksien pitämisen, muiden asioihin sekaantumattomuuden, aikasynkronoinnin ja algoritmien askelten suorittami-

sen. Myönteiset kokemukset näillä alueilla auttavat muita tahoja muodostamaan kollektiivisen suosituksen, jonka mukaan kyseiseen tahoon voi luottaa digitaalisten allekirjoitusten suhteen.

4.4 Metatieto

On hyvin yleistä, että varsinaista järjestelmään tallennettua informaatiota ei ole jäsennelty tai se on yksinkertaisesti niin monimutkaista, ettei sen automaattinen käsittely ole mahdollista. Automaattisen indeksoinnin, haun ja muun käsittelyn helpottamiseksi on usein järkevää liittää informaatioobjektiin "tietoa tiedosta" eli kuvailevaa lisäinformaatiota. Tätä kutsutaan *metatiedoksi* (metadata) ja se muodostaa enenevässä määrin tärkeän osan esimerkiksi elektronisessa julkaisutoiminnassa. Metatietoon voi myös liittää metatietoa, koska metatieto on informaatiota siinä missä alkuperäinen informaatiokin.

Metatieto on sovellusalueesta riippuen erilaista. Tyypillisesti indeksointia ja hakua varten suunnitellusta *kaavasta* (schema) löytyy dokumentin nimi, aiheluokitus jonkin järjestelmän mukaan, tiivistelmä, tekijä, tekoaika, voimassaoloaika ja muita hyödyllisiä perustietoja. Kun metatietoon liitetään mukaan metatiedon luoja, voidaan metatieto tulkita väitteeksi luottamuksenhallinnan näkökulmasta (tosin ilman luottamusta, jos metatiedon eheyttä ja alkuperää ei voida todentaa).

Esimerkki: Dublin Core. Dublin Core [Tuo98] on verkkojulkaisemista varten kehitetty metatietokaava, joka on tarkoitettu lähinnä kirjastokäyttöön. Sen avulla on yksinkertaista antaa esimerkki siitä, mitä metatieto käytännössä tarkoittaa. Dublin Core -määrittelyt voidaan sijoittaa esimerkiksi HTML-dokumentin [RLHJ98] otsikkotietojen metatietokenttään esimerkiksi kuten kuvassa 11. Kuvassa on kaksi metatietoelementtiä (riveillä 3–5): dokumentin kieli (suomi, `fi`) ja tekijä. Tekijän nimen kieli on annettu vielä erikseen metatietoelementin yhteydessä.

Esimerkki: PICS. Internetin levittyä kansan syviin riveihin alkoi esiintyä huolestumista Internetissä (lähinnä seitissä) julkaistun materiaalin poliittisesta ja moraalisesta puhtaudesta. Eri kansalaisjärjestöt näkivät seitin valtavana paheellisen materiaalin levityskoneistona. Pääasiassa tästä syystä World Wide Web Consortium käynnisti PICS-hankkeen (Platform for Internet Content Selection) [RM96], [MRS96], [KMRT96], joka on pohjimmiltaan metatietokuvausmenetelmä seittisivuja varten. PICSin periaate on itsesensuuri: tiedon julkaisija määrittelee dokumenttiin liitetyin *PICS-leimoin* (PICS label)

```
1    <!DOCTYPE HTML PUBLIC ''-//W3C//DTD HTML 4.0//EN''>
2    <HTML><HEAD><TITLE>Dublin Core -esimerkki</TITLE>
3    <META name=''DC.creator'' lang=''fi''
4    content=''Antti Vähä-Sipilä''>
5    <META name=''DC.language'' content=''fi''>
6    </HEAD><BODY>Esimerkkidokumentti.</BODY></HTML>
```

Kuva 11: Dublin Core HTML-dokumentissa

dokumentin sisältämän paheen määrän osa-aluekohtaisesti. Eri tarkoituksia varten on olemassa eri kaavoja. Eräs verraten laajasti käytetyistä kaavoista, RSACi (Recreational Software Advisory Council on the Internet), sisältää kentät väkivallan, rumien sanojen, alastomuuden ja seksin määrän arviointia varten.

PICS-leimat voidaan levittää HTML-dokumenttien metatietokentissä aikaisemman Dublin Core -esimerkin mukaisesti tai vaikkapa sähköpostin otsikkotiedoissa ja HTTP-protokollan vasteissa.

4.5 Luottamuksenhallinta PICS-leimoilla

REFEREE. PICSin sensuurimalli on herättänyt voimakkaita tunteita sekä puolesta että vastaan, mutta totuus on, että se on tiettyihin tarkoituksiin hyödyllinen keksintö. PICS-leimoilla voi luokitella myös esimerkiksi ohjelmiston puhtautta viruksista.

PICS-leiman merkitys on "jokin taho on luokitellut sisällön seuraavasti". Tämä voidaan tulkita myös "jokin taho väittää, että x", jossa x on jokin attribuutti-arvopari. Tämä kuulostaa hyvin paljon PolicyMakerin väitteeltä ja näin onkin luonnollista soveltaa luottamuksenhallinnan menetelmiä myös PICS-leimojen hallintaan. PolicyMakerin ja AWKWARDin käyttöä PICS-leimojen luottamuksenhallinnassa käsitellään tarkemmin lähteessä [BFRS96], mutta tässä esitellään uusi järjestelmä nimeltään REFEREE, Rule-controlled Environment For Evaluation of Rules, and Everything Else [CFL+98]. Nimi viittaa siihen, että REFEREE pyrkii ratkaisemaan ongelman,

joka PolicyMakerissa ja KeyNotessa on jätetty syrjään: myös luottamuksenhallintakoneistolle syötettävän tiedon hakeminen on tehtävä jonkin politiikan mukaan.

Syy tähän on se, että jos hyökkääjällä on mahdollisuus tarkkailla luottamusta arvioivan tahon verkkoliikennettä, voi siitä päätellä hyvinkin paljon siitä ongelmasta, johon sovellus pyytää luottamuksenhallintajärjestelmän kannanottoa. Tätä kutsutaan *piilokanavaksi* (covert channel). Esimerkiksi julkisia avaimia jakava hakemistopalvelin voi haettujen avainten omistajien nimien perusteella muodostaa varsin kattavan profiilin tahoista, joiden kanssa sovellus toimii, tai ainakin joiden kanssa se harkitsee toimivansa.

REFEREE**n ominaisuuksia.** REFEREE tekee kaikki päätökset valtuutusten noutamisesta itse politiikan ohjaamana. REFEREE erottelee järjestelmän pääosiksi *ohjelmat* (program), *lausunnot* (statement list) sekä paluuarvot. Ohjelmat vastaavat PolicyMakerin suodattimia ja lausunnot väitteitä.[19] Ohjelmilla ei kuitenkaan ole suoritusrajoitteita ja ne voivat ajaa toisia ohjelmia. Ajokelpoiset ohjelmat on tallennettu erilliseen tietokantaan, jonne voidaan lisätä uusia ohjelmia politiikan ohjaamana. Ohjelmat kirjoitetaan profiles-0.92 -nimisellä kielellä [Chu97].

Kuten muutkin esitellyt luottamuksenhallintajärjestelmät, REFEREE (tai oikeammin jokainen järjestelmän ohjelma) ottaa syötteenään sarjan lausuntoja sekä vapaavalintaisia argumentteja. Jokainen ohjelma voi palauttaa sekä uuden lausuntojoukon että totuusarvon. Totuusarvo on kolmitilainen (tosi, vale, määrittelemätön). Ohjelma voi totuusarvolla "vale" ilmaista, että ohjelman suoritus epäonnistui (esimerkiksi pääsy julkisten avainten jakopalvelimelle ei onnistunut tai koneen resurssit ovat lopussa). Myös politiikkaa valvova ohjelma voi palauttaa vastaavat totuusarvot, eli aiemmista järjestelmistä poiketen REFEREE voi siis myös antaa hylkäävän tuomion (jolloin on löytynyt todisteita sille, että toimintapyyntö oli itse asiassa laiton ja hälytyskelloja saattaisi olla aiheellista soittaa).

Esimerkki: REFEREE**-ajo.** Määritellään esimerkkipolitiikka siten, että tietyn seittisivun katsominen on sallittua vain, jos sivun `paheellisuus` on pienempi kuin 2. Paheellisuuden määrän arvioi kuvitteellisen Tarkastamon työryhmä.

Emme kuitenkaan halua paljastaa mahdollisesti paheellista sisältöä levittävän seittisivun ylläpitäjälle olevamme kiinnostuneita sivus-

[19] REFEREEstä puhuessamme käytämme sen omia termejä, jotta lukijalla ei olisi vääriä ennakko-oletuksia väitteiden ja suodattimien olemuksesta.

ta, mikäli se sattuisikin olemaan liian paheellinen makuumme. Tämän vuoksi haemme PICS-leimat (myöskin kuvitteellisesta) osoitteesta `http://www.tarkastamo.fi/pics/`. Politiikan mukaan kyseisellä leimalla on myös oltava oikea Tarkastamon digitaalinen allekirjoitus. Tiedossamme on Tarkastamon julkinen avain `0x31337d00dabba`.

Kuvassa 12 on esitetty tämä politiikka profiles-0.92 -kielellä.

```
1    (invoke ''load-label'' STATEMENT-LIST URL
2        ''http://www.tut.hut.lut.fi/''
3        (''http://www.tarkastamo.fi/pics/''))
4    (invoke ''check-signature'' STATEMENT-LIST ''0x31337d00dabba'')
5    (false-if-unknown
6        (match ((''load-label'' ''check-signature'' *)
7            ((version ''PICS-1.1'') *
8            (service ''http://www.tut.hut.lut.fi/'') *
9            (ratings * (paheellisuus < 2) * )))
10           STATEMENT-LIST))
```

Kuva 12: profiles-0.92 -kielellä kirjoitettu politiikka

Kun REFEREE alkaa käydä politiikkaa läpi, se käynnistää rivillä 1 ensin toisen ohjelman `load-label`, joka hakee tutkittavaa URLia kuvaavat PICS-leimat Tarkastamolta. Mikäli leimoja löytyy, ne lisätään lausuntojen joukkoon (`STATEMENT-LIST`), joka aluksi oli tyhjä. Lausuntoihin lisätään myös ohjelman nimi, joka lausunnon tuotti.

Tämän jälkeen suoritetaan rivillä 4 toinen ohjelma, `check-signature`. Se saa argumenttinaan lausunnot ja politiikkaan kirjatun Tarkastamon julkisen avaimen, tarkistaa allekirjoituksen ja allekirjoituksen hyväksyttyään palauttaa lausunnot omalla nimellään lisättynä. PICS-leimat allekirjoitetaan käytännössä World Wide Web Consortiumin kehittelemää DSig-allekirjoitusjärjestelmää [CDLL98] käyttäen.

Politiikan viimeinen, riviltä 5 alkava vaihe tekee vertailun (`match`) lausuntojen joukon ja hyväksyttävään tulokseen oikeuttavan lausuntojen joukon välillä. Hyväksyttävältä lausuntojoukolta vaaditaan ensinnäkin sekä

`load-label`in että `check-signature`n hyväksytty suoritus. Tämän lisäksi PICS-leiman version on oltava oikea, sen on kuvattava haluamaamme palvelua ja paheellisuuden on oltava kakkosta pienempi. Lausunnot ovat sulkulausekkeita [Chu97] ja `match`-lauseen ensimmäisessä parametrissa on yllä käytetty jokerimerkkinä tähteä ("`*`") tarkoittamaan mitä tahansa merkkijonoa.

Jos vertailu onnistuu, politiikka palauttaa sovellukselle "tosi". Jos vertailu epäonnistuu tai tulos jää epäselväksi, palautetaan "vale". Sovellus voi toimia tämän tuloksen pohjalta parhaaksi katsomallaan tavalla (esimerkissämme "tosi" aiheuttaa sivun haun palvelimelta).

PICSRules. World Wide Web Consortiumin PICS- ja DSig-hankkeiden [CDLL98] jälkeläisenä syntynyt PICSRules [PEF+97], [Chu97] on menetelmä yksinkertaisten PICS-leimoihin perustuvien politiikkojen kuvaamiseen. Se ei ole varsinaisesti luottamuksenhallintajärjestelmä vaan ainoastaan politiikkojen kuvauskieli. PICSRulesissa saattaa olla ainesta yleiseksi politiikkojen siirtotavaksi luottamuksenhallintajärjestelmien välillä. REFEREE-järjestelmä voi käyttää myös PICSRulesilla määriteltyä politiikkaa ja REFEREEn esimerkkitoteutus ymmärtääkin PICSRulesia [Chu97].

4.6 Luottamuksenhallinta XML-dokumenteissa

Dokumenttien rakennekuvauskielistä ehkä suurelle yleisölle tutuin esimerkki on HTML (Hypertext Mark-up Language) [RLHJ98], jota käytetään seittisivuja kirjoitettaessa. Rakennekuvauskielellä erotellaan dokumentin loogiset osat niin, että dokumentin automaattinen jäsennys on mahdollista. HTML:ssä tyypilliset rakenteet ovat tekstikappaleita, otsikkoja ja alaotsikkoja, listoja ja listojen alakohtia ja niin edelleen. Rakennekuvausta kutsutaan DTD:ksi (Document Type Definition). Rakenneosia kutsutaan *elementeiksi*.

HTML on kuvattu SGML:llä (Single Generalized Mark-up Language), jolla voidaan kuvata myös muihin tarkoituksiin sopivia rakennekuvauskieliä. SGML:n rinnalle ja erityisesti Internetin tarpeisiin on nousemassa yksinkertaisempi XML (Extensible Mark-up Language) [BPSM98]. XML:llä, kuten SGML:llä, voidaan siis määritellä rakennekuvauskieli lähes mitä tahansa olemassaolevaa rakenteellista tietoa varten. Tiedon ei suinkaan tarvitse olla ihmisen luettavaksi tarkoitettu dokumentti.

Allekirjoitukset XML-dokumenteissa. XML-dokumentteihin voidaan myös liittää metatietoa. XML-maailmassa metatiedon välitykseen on luo-

tu infrastruktuuri nimeltä RDF (Resource Description Framework) [LS98]. RDF:n käyttäminen digitaalisten allekirjoituksien siirtoon on täysin luonnollinen seuraava askel, ja koska XML-dokumentit RDF-metatietoineen on tarkoitettu automaattisesti indeksoitaviksi, jäsennettäviksi ja muuten käsiteltäviksi, tulee myös automaattisen luottamuksenhallintajärjestelmän mahdollisuuden tutkiminen ajankohtaiseksi.

Tätä kirjoitettaessa digitaalisten allekirjoitusten lisääminen XML-dokumentteihin on vasta tutkimusasteella. Sen lisäksi, että allekirjoitus käyttäisi RDF:ää, on esitetty myös erillisen allekirjoitus-DTD:n luomista [Bro99]. Kun XML-dokumentissa halutaan allekirjoittaa jokin sen elementti, ilmaistaan XML-dokumentin alussa, että dokumentissa voidaan käyttää allekirjoitus-DTD:ssä määriteltyjä rakenteita. Allekirjoituselementti sisältää viitteen allekirjoitettavaan XML-elementtiin ja itse allekirjoituksen. Idea on sama kuin RDF:ssä ja on varsin todennäköistä, että ehdotus alkaa käyttää RDF:ää myöhemmässä vaiheessa ja että allekirjoitus-DTD:stä tulee RDF-*kaava* (schema).

XML-dokumenttien luottamuksenhallinta. Käsitteellisesti XML:llä määritellyillä dokumenttityypeillä kirjoitetut dokumentit ja RDF ovat täysin HTML:ää ja PICSiä vastaavia (itse asiassa HTML voidaan määritellä myös XML:llä). Täten lähes kaikki luottamuksenhallinnalliset näkökohdat, jotka pätevät HTML:ään ja PICSiin, ovat paikkansapitäviä myös XML-maailmassa.

XML:llä kuvatut varmenteet. *Kykykortit* (capability cards) ovat eräs hiljattain julkaistu ehdotus SPKI/SDSI-varmenteiden kaltaiseksi varmennejärjestelmäksi [OSM98]. Kykykortit on kuvattu XML:llä ja niiden maailmassa on kolmenlaisia tahoja: *subjekti*, johon kortin määrittelemä *kyky* (capability) liittyy, kortin *haltija* (holder), jolla kortti on (ja joka ei välttämättä ole subjekti), sekä *varmentaja* (verifier), joka pystyy varmentamaan kykykortin oikeellisuuden. Kykykortti on tavallaan todistus siitä, että subjektilla on oikeus johonkin tekoon; jos subjekti on anonyymi, kortin kulloinenkin haltija rinnastetaan sen subjektiin.

Maininnan arvoiseksi ehdotuksen tekee se, että [OSM98] esittää useita käyttötilanteita, jotka ovat hyödyllisiä Internetissä. Ehkä mielenkiintoisin ominaisuus on kortin delegointi. Tällöin delegoija luo uuden kykykortin, joka sisältää kopion vanhasta kortista. Alkuperäisen kortin luoja voi asettaa delegoinnille rajoituksia mm. delegoinnin ”syvyyden” (kuinka monta peräkkäistä delegointia sallitaan) ja ”leveyden” (kuinka monelle taholle voi yksi taho kortin delegoida) suhteen. Delegointiominaisuus mahdollistaa kykykorttien

käyttämisen esimerkiksi alennuskuponkeina tai maksupoletteina, jotka voidaan luovuttaa kolmannelle osapuolelle, mutta jotka eivät kuitenkaan vastaa varsinaista rahaa, koska niiden levitys on rajoitettua. Tällaisten rahankaltaisten maksuinstrumenttien luonti on huomattavasti selkeämpää kuin varsinaisen sähkörahan luonti, koska rahan tai maksuvälineiden liikkeelle laskenta on useasti laeilla rajoitettua.

5 Yhteenveto

Työn alussa kuvattu julkisen avaimen salauksen käsite on suuresti helpottanut salausavainten sopimista kahden kommunikoivan osapuolen välillä. Salaus on kuitenkin vain yksi tarpeellisista turvallisuuspalveluista. Kommunikoivien osapuolien todennus on myös yleensä tarpeen. Työssä tarkasteltiin avaintenhallintamenetelmiä, joilla julkisen avaimen alkuperästä voidaan varmistautua ja näin saada aikaan todennettu kommunikointikanava.

Tärkeimmät avaintenhallintamallit ovat hierarkkinen eli keskusjohtoinen varmennusjärjestelmä sekä luottamusverkko, jossa luottamus keskitetään kunkin todentajan omaan salaiseen avaimeen. Molemmat mallit ovat nykyään käytössä. Hierarkkista ajattelua edustaa X.509 ja luottamusverkkomallia PGP. Näiden varmennetyyppien osalta tarkasteltiin ongelmia, joita seuraa julkisen avaimen ja nimen sitomisesta toisiinsa sekä esiteltiin järjestelmä, jossa avaimen omistaja samaistetaan avaimeensa (SPKI/SDSI).

Kun avaimen alkuperästä on varmistuttu, seuraava vaihe on varmistua avaimen omistaman tahon oikeudesta hänen haluamaansa palveluun. Työssä esiteltiin luottamuksenhallintaongelma sekä tapoja, joilla ongelmaa voidaan yrittää automaattisesti ratkaista. Teoriapohjan luottamuksenhallinnalle loi suurelta osin PolicyMaker. Myös muita luottamuksenhallintajärjestelmiä käytiin lyhyesti läpi. Työn lopuksi esiteltiin metatiedon esitystapoja ja luottamuksenhallinnan suhdetta näihin.

5.1 Tulevaisuuden kehitysnäkymiä

1980-luku oli nykyperspektiivistä nähden Internetin esihistoriaa. Seitti otti ensiaskeleensa 1990-luvun alussa ja räjähti kaiken kansan tietoisuuteen sen puolivälin jälkeen. Internetin turvaprotokollat ovat edelleen osin määrittelyvaiheessa ja niiden toteutuksia on tullut loppukäyttäjien saataville vasta

parin viime vuoden aikana. Salausteknologian vientivalvontamääräykset sekä eri maiden käyttörajoitteet ovat osaltaan hidastaneet prosessia, lähinnä vaikeuttamalla kaikkea alan standardointi- ja toteutustyötä. Parannusta ei valitettavasti ole lähiaikoina näkyvissä. Onneksi kryptografisiin menetelmiin perustuva todennus (ilman salausta) on kuitenkin valtaosin rajoituksista vapaata.

Lähitulevaisuudessa Internetin avaintenhallinta perustunee X.509:ään, koska suurin osa määrittelyvaiheessa olevista protokollista tulee käyttämään sitä hyväkseen. Internetin peruspiirteenä on aina ollut, että teknistä kehitystä ei ohjaa välttämättä paras teoreettinen pohja vaan ensimmäinen toimiva ja laajalle levinnyt toteutus (tästä myös tunnettu lausahdus "rough concensus and working code"). SPKI/SDSI:n ideat tulevat todennäköisesti hyödynnetyksi jossakin vaiheessa joissakin X.509:n tai PGP:n ympärille rakennetuissa ratkaisuissa, mutta on epätodennäköistä, että kyseinen aloite olisi sellaisenaan käytössä ainakaan kovin nopealla aikataululla. Jo nyt on kuitenkin merkkejä siitä, että tietyissä sovelluksissa ajatellaan luottamusjuurena käytettävän käyttäjän tai laitteen omaa avainta, vaikka varmennehallintajärjestelmänä olisikin X.509.

Internetin tulevaisuuden sovelluksiin sisältyy varmasti automaattisen tiedon hakuun ja jalostukseen erikoistuneita henkilökohtaisia agentteja. Tiedon jäsentelyssä on tapahduttava huomattavaa kehitystä, mutta XML ja sen metatietojärjestelmä RDF tuntuvat lupaavalta tekniikalta. Tiedon etsinnässä digitaaliset allekirjoitukset ja luottamuksenhallinta ovat oleellisessa asemassa väärän tiedon suodattamisessa. Agenttien pääsynvalvonta on myös oma ongelmakenttänsä, koska ne toimivat pääosin autonomisesti, eikä esimerkiksi käyttäjä voi (tai saa) olla vastuussa siitä, mitä agentti yrittää tehdä. Luottamuksenhallinnan osalta perusperiaatteet luotiin PolicyMakerissa ja se luonee henkisen pohjan tulevaisuuden kehitykselle pieniä ideologisia eroja lukuunottamatta (esimerkiksi REFEREEn idea asettaa kaikki luottamuksen tarkastelun vaiheet politiikan valvontaan).

Nopeasti etenevän alan tulevaisuutta on vaikea ennustaa, mutta Internetin leviäminen kaikkialle on lähes varmaa. Vielä nykyään käyttäjä yleensä tiedostaa käyttävänsä Internetiä esimerkiksi selaillessaan seittiä tai lähettäessään sähköpostia, mutta jo nyt jotkin palvelut käyttävät Internetiä loppukäyttäjälle läpinäkyvästi — esimerkkinä Internetin kautta välitettävät puhelut, jotka kuitenkin reititetään tavalliseen puhelinverkkoon. Internet, joka suuren yleisön mielestä on edelleen sama asia kuin seitti, muuttuu vähä vähältä vain "töpseliksi seinässä", johon laitteet on kytkettävä samoin kuin sähköverkkoonkin, elleivät satu olemaan langattomia.

Hyvänä esimerkkinä tästä ajattelumallista on Sun Microsystemsin Jini [Wal98], jossa esimerkiksi kodinkoneet sisältävät Jini-agentteja. Kahvinkeitin tunnistaa keittiön verkkoympäristöstä laitteet, jotka ovat kykeneviä tarjoamaan käyttäjälle käyttöliittymän kahvinkeittoa varten. Keitin välittää oman käyttöliittymänsä näille komponenttina, jonka ne voivat näyttää käyttäjälle. Esimerkiksi keittiöön astelevan ihmisen Jini-rannekello voisi alkaa näyttää kellonajan sijasta kahvinkeittimen käyttöliittymää kun se saapuu kahvinkeittimen langattoman verkkoyhteyden kantomatkan päähän — tietenkin todennettuaan itsensä kahvinkeittimelle ja hyväksyttyään keittiön haltijan todistuksen kahvinkeittimen rehellisyydestä. Muita palveluita voisivat olla esimerkiksi tietojen haku ja tallennus Jini-agentin hallitsemalle kovalevylle, jolloin kovalevyn fyysinen sijainti muuttuu merkityksettömäksi.

Laitteet voivat myös sulautua käyttäjänsä vaatteisiin tai vaikkapa elimistöön ja tieto, jota ne välittävät, vaihtelee kättelemällä siirtyvistä käyntikorteista reaaliaikaiseen sydän- tai aivokäyrään. Jotta nämä visiot voidaan missään määrin toteuttaa, on turvaprotokollien kehityttävä huomattavasti niiden käytettävyyden ja huomaamattomuuden kannalta. Verkkoympäristö, johon koneet liittyvät langattomasti, esimeriksi infrapunalla (IrLAN) tai pienitehoisella radiolähetteellä (LPRF, low power radio frequency) toimiva paikallisverkko ja jossa ne tarjoavat jatkuvasti omia ohjelmakomponenttejaan ajettavaksi toisilleen, suorastaan huutaa läpinäkyvää ja automatisoitua luottamuksenhallintajärjestelmää. Itse salaus voidaan julkisten avainten menetelmiä käyttäen tehdä jo nyt täysin kommunikoivien tahojen kannalta huomaamattomasti. Avainasemassa käytettävyydessä ovat täten avaintenhallinta, luottamuksen tarkastelu sekä loppukäyttäjän (ihmisen) luotettava todentaminen, sillä ne ovat vielä tällä hetkellä ainoat asiat, joita ei automaattisesti, ilman jotakin ennakkotietoa, voida hoitaa.

Viitteet

[AAB+98] Hal Abelson, Ross Anderson, Steven M. Bellovin, Josh Benaloh, Matt Blaze, Whitfield Diffie, John Gilmore, Peter G. Neumann, Ronald L. Rivest, Jeffrey I. Schiller, and Bruce Schneier. The risks of key recovery, key escrow and trusted third party encryption, 1998. http://www.cdt.org/crypto/risks98/ (1998-07-30/1998-12-30).

[Aba97] Martín Abadi. On SDSI's linked local name spaces. In *PCSFW: Proceedings of The 10th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1997.

[AG96] Ken Arnold and James Gosling. *The Java programming language*. Addison-Wesley, 1996.

[Bal93] D. Balenson. Privacy enhancement for Internet electronic mail: Part III: Algorithms, modes, and identifiers. *RFC 1423*, February 1993. ftp://ftp.nordu.net/rfc/rfc1423.txt.

[BBK94] Thomas Beth, Malte Borcherding, and Birgit Klein. Valuation of trust in open networks. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS)*, number 875 in Lecture Notes in Computer Science, pages 3–18. Springer-Verlag, 1994.

[BFK98a] Matt Blaze, Joan Feigenbaum, and Angelos D. Keromytis. KeyNote: Trust management for public key infrastructures. In *Proc. of the Cambridge University Workshop on Trust and Delegation*, April 1998.

[BFK98b] Matt Blaze, Joan Feigenbaum, and Angelos D. Keromytis. The KeyNote trust-management system. *Internet-Draft (Work in progress)*, November 1998. http://www.ietf.org/internet-drafts/draft-blaze-ietf-trustmgt-keynote-00.txt.

[BFL96] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. Technical Report 96-17, DIMACS, June 28 1996.

[BFRS96] Matt Blaze, Joan Feigenbaum, P. Resnick, and Martin Strauss. Managing trust in an information-labeling system. Technical Report TR 96.15.1, AT&T Labs Research, November 1996.

[BFS98] M. Blaze, J. Feigenbaum, and M. Strauss. Compliance checking in the Policy Maker trust management system. In *Financial Cryptography*, volume 1465 of *Lecture Notes in Computer Science*, pages 254–. Springer-Verlag, 1998. Also published as AT&T Labs Research technical report TR 98.3.2.

[BPSM98] Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen. Extensible markup language (XML) 1.0. *W3C Recommendation REC-xml-19980210*, February 1998. http://www.w3.org/TR/REC-xml.

[Bro99] Richard D. Brown. Digital signatures for XML. *Internet-Draft (Work in progress)*, January 1999. http://www.ietf.org/internet-drafts/draft-brown-xml-dsig-00.txt.

[CDK95] George Coulouris, Jean Dollimore, and Tim Kindberg. *Distributed Systems: Concepts and Design.* Addison-Wesley, second edition, 1995.

[CDLL98] Yang-Hua Chu, Philip DesAutels, Brian LaMacchia, and Peter Lipp. PICS signed labels (DSig) 1.0 specification. *W3C Recommendation REC-DSig-label-19980527*, May 1998. http://www.w3.org/TR/REC-DSig-label/.

[CFL+98] Yang-Hua Chu, Joan Feigenbaum, Brian LaMacchia, Paul Resnick, and Martin Strauss. REFEREE: Trust management for web applications, 1998. http://www.farcaster.com/papers/www6-referee/www6-referee.html (1998-12-03/1998-12-29).

[Chu97] Yang-Hua Chu. Trust management for the World Wide Web. Master's thesis, Dep. of Elect. Engineering and Computer Science of MIT, June 1997.

[Cro82] David H. Crocker. Standard for the format of ARPA Internet text messages. *RFC 822, Internet Standard 0011*, August 1982. ftp://ftp.nordu.net/rfc/rfc822.txt.

[DA99] Tim Dierks and Christopher Allen. The TLS protocol version 1.0. *RFC 2246*, January 1999. ftp://ftp.nordu.net/rfc/rfc2246.txt.

[DH76] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[DHRW98] S. Dusse, P. Hoffman, B. Ramsdell, and J. Weinstein. S/MIME version 2 certificate handling. *RFC 2312*, March 1998. ftp://ftp.nordu.net/rfc/rfc2312.txt.

[DR97] Dale Dougherty and Arnold Robbins. *sed & awk*. O'Reilly & Associates, second edition, 1997.

[EFL+98a] Carl M. Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tatu Ylönen. SPKI certificate theory. *Internet-Draft (Work in progress)*, November 1998. http://www.ietf.org/internet-drafts/draft-ietf-spki-cert-theory-04.txt.

[EFL+98b] Carl M. Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tatu Ylönen. Simple public key certificate. *Internet-Draft (Work in progress)*, March 1998. http://www.ietf.org/internet-drafts/draft-ietf-spki-cert-structure-05.txt.

[EFL+98c] Carl M. Ellison, Bill Frantz, Butler Lampson, Ron Rivest, Brian M. Thomas, and Tatu Ylönen. SPKI examples. *Internet-Draft (Work in progress)*, March 1998. http://www.ietf.org/internet-drafts/draft-ietf-spki-cert-examples-01.txt.

[Ell87] James Ellis. The story of non-secret encryption. *Communications-Electronics Security Group*, 1987. Released 1998. http://jya.com/ellisdoc.htm (1998-02-26/1998-12-29).

[Ell96a] Carl Ellison. Establishing identity without certification authorities. In *6th USENIX Security Symposium, July 22–25, 1996. San Jose, CA*, pages 67–76, Berkeley, CA, USA, July 1996. USENIX.

[Ell96b] Carl M. Ellison. Generalized certificates, 1996. http://www.clark.net/pub/cme/html/cert.html (1998-04-23/1998-12-30).

[Ell98] Carl M. Ellison. SPKI requirements. *Internet-Draft (Work in progress)*, October 1998. http://www.ietf.org/internet-drafts/draft-ietf-spki-cert-req-02.txt.

[FL97] Joan Feigenbaum and Peter Lee. Trust management and proof-carrying code in secure mobile-code applications: A position paper. In *Proc. of the DARPA Workshop on Foundations for Secure Mobile Code*, pages 48–55, March 1997.

[Gut98] Peter Gutmann. X.509 style guide, August 1998. http://www.cs.auckland.ac.nz/%7Epgut001/pubs/x509guide.txt (1998-10-09/1998-12-29).

[ITU93] ITU-T Recommendation X.509 (Information Technology – Open Systems Interconnection – The Directory: Authentication Framework), ISO/IEC International Standard 9594-8, 1993. International Telecommunication Union.

[JY98] Markus Jakobsson and Moti Yung. On assurance structures for WWW commerce. In *Financial Cryptography*, volume 1465 of *Lecture Notes in Computer Science*, pages 141–. Springer-Verlag, 1998.

[KA98] S. Kent and R. Atkinson. Security architecture for the Internet protocol. *RFC 2401*, November 1998. ftp://ftp.nordu.net/rfc/rfc2401.txt.

[Kah96] David Kahn. *The Codebreakers: The Story of Secret Writing.* Scribner, revised and updated edition, 1996.

[Kal93] B. Kaliski. Privacy enhancement for Internet electronic mail: Part IV: Key certification and related services. *RFC 1424*, February 1993. ftp://ftp.nordu.net/rfc/rfc1424.txt.

[Kal98] B. Kaliski. PKCS 10: Certification request syntax version 1-5. *RFC 2314*, March 1998. ftp://ftp.nordu.net/rfc/rfc2314.txt.

[KCYC96] Nada Kapidzic Cicovic, Andrew Young, and D. W. Chadwick. Merging and extending the PGP and PEM trust models - the ICE-TEL trust model, 1996.

[KCYGF96] Nada Kapidzic Cicovic, Andrew Young, Petra Glöckner, and Stephen Farell. Architecture and general specifications of the

public key infrastructure. Public Deliverable D1, ICE-TEL – Interworking Public Key Certification Infrastructure for Europe, TELEMATICS Project RE 1005, September 1996.

[Ken93] S. Kent. Privacy enhancement for Internet electronic mail: Part II: Certificate-based key management. *RFC 1422*, February 1993. ftp://ftp.nordu.net/rfc/rfc1422.txt.

[KMRT96] Tim Krauskopf, Jim Miller, Paul Resnick, and Win Treese. PICS label distribution: Label syntax and communication protocols: Version 1.1. *W3C Recommendation REC-PICS-labels-961031*, October 1996. http://www.w3.org/TR/REC-PICS-labels.

[Koh78] Loren M. Kohnfelder. Towards a practical public-key cryptosystem. B.S. Thesis, supervised by L. Adleman, May 1978.

[Lin93] J. Linn. Privacy enhancement for Internet electronic mail: Part I: Message encryption and authentication procedures. *RFC 1421*, February 1993. ftp://ftp.nordu.net/rfc/rfc1421.txt.

[LN98] Ilari Lehti and Pekka Nikander. Certifying trust. In *Public Key Cryptography*, volume 1431 of *Lecture Notes in Computer Science*, pages 83–. Springer-Verlag, 1998.

[LS98] Ora Lassila and Ralph R. Swick. Resource description framework (RDF) model and syntax specification. *W3C Working Draft WD-rdf-syntax-19981008*, October 1998. http://www.w3.org/TR/WD-rdf-syntax/.

[Mau96] Ueli Maurer. Modelling a public-key infrastructure. In *Proceedings 1996 of the European Symposium on Research in Computer Security (ESORICS '96)*, number 1146 in Lecture Notes in Computer Science. Springer-Verlag, September 1996.

[McB98] Neal McBurnett. PGP web of trust statistics, 1998. http://bcn.boulder.co.us/%7Eneal/pgpstat/ (-/1998-12-29).

[MRS96] Jim Miller, Paul Resnick, and David Singer. Rating services and rating systems (and their machine readable descriptions). *W3C Recommendation REC-PICS-services-961031*, October 1996. http://www.w3.org/TR/REC-PICS-services.

[MVOV96] Alfred J. Menezes, Paul C. Van Oorschot, and Scott A. Vanstone. *Handbook of Applied Cryptography*. CRC Press Series on Discrete Mathematics and Its Applications. CRC Press, 1996.

[NSA98] Threat and vulnerability model for key recovery (KR), February 1998. National Security Agency.

[OSM98] Koji Otani, Hiroyasu Sugano, and Madoka Mitsuoka. Capability card: An attribute certificate in XML. *Internet-Draft (Work in progress)*, November 1998. http://www.ietf.org/internet-drafts/draft-otani-ccard-00.txt.

[PEF+97] Martin Presler, Christopher Evans, Clive D. W. Feather, Alex Hopmann, and Paul Resnick. PICSRules 1.1. *W3C Recommendation REC-PICSRules-971229*, December 1997. http://www.w3.org/TR/REC-PICSRules.

[Pre96] Oliver Pretzel. *Error-Correcting Codes and Finite Fields*. Oxford applied mathematics and computing science series. Oxford University Press, 1996.

[Riv97] Ronald L. Rivest. SPKI/SDSI 2.0: A simple distributed security infrastructure. A presentation for Maryland Theoretical Computer Science Day, April 1997. http://theory.lcs.mit.edu/%7Erivest/sdsi20-maryland.ppt (1997-05-27/1998-12-30).

[Riv98] R. L. Rivest. Can we eliminate certificate revocation lists? In *Financial Cryptography*, volume 1465 of *Lecture Notes in Computer Science*, pages 178–. Springer-Verlag, 1998.

[RL96] Ronald L. Rivest and Butler Lampson. SDSI – A simple distributed security infrastructure, version 1.1. Originally (version 1.0) presented at CRYPTO'96 Rump session, April 1996. http://theory.lcs.mit.edu/%7Erivest/sdsi11.html (1997-03-25/1998-12-29).

[RLHJ98] Dave Raggett, Arnaud Le Hors, and Ian Jacobs. HTML 4.0 specification. *W3C Recommendation REC-html40-19980424*, April 1998. http://www.w3.org/TR/REC-html40/.

[RM96] Paul Resnick and James Miller. PICS: Internet access controls without censorship. *Communications of the ACM*, 39(10):87–93, 1996.

[Ruo93] Keijo Ruohonen. *Koodaus- ja informaatioteoria.* TTKK opintomoniste 168. Tampere University of Technology, 1993.

[Sal96] Arto Salomaa. *Public Key Cryptography.* EATCS. Springer-Verlag, second edition, 1996.

[Sch96] Bruce Schneier. *Applied Cryptography: Protocols, Algorithms and Source Code in C.* Wiley and Sons, second edition, 1996.

[SET97] SET secure transaction specification, book 1: Business description, version 1.0, May 1997. SET Secure Electronic Transaction LLC. http://www.setco.org/download/set_bk1.pdf (1998-01-20/1999-02-10).

[Sta94] William Stallings. *Data and Computer Communications.* Prentice Hall, fourth edition, 1994.

[Sti95] Douglas R. Stinson. *Cryptography: theory and practice.* CRC Press Series on Discrete Mathematics and Its Applications. CRC Press, 1995.

[Tha98] Rodney Thayer. PKI requirements for IP security. *Internet-Draft (Work in progress)*, September 1998. http://www.ietf.org/internet-drafts/draft-ietf-ipsec-pki-req-01.txt.

[Tuo98] Kimmo Tuominen. Lisää julkaisujen näkyvyyttä verkossa: käytä metadataa. *@CSC*, lokakuu 1998.

[Ver97] VeriSign, Inc. *VeriSign certification practice statement*, May 1997. Version 1.2.

[Wal98] Jim Waldo. Jini architecture overview, 1998. Sun Microsystems, Inc. http://java.sun.com/products/jini/whitepapers/architectureoverview.pdf (1999-01-17/1999-01-18).

[WAP98] Wireless application protocol: Wireless transport layer security specification, version 1.0, April 1998. Wireless Application Protocol Forum Ltd.

Hakemisto