# Käyttäjätunnistuksen ongelmia ja illuusio tietoturvasta

Niko Santala

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Tietojenkäsittelyoppi
Pro gradu -tutkielma
Ohjaaja: Jyrki Nummenmaa
Kesäkuu 2008

Tampereen yliopisto
Tietojenkäsittelytieteiden laitos
Tietojenkäsittelyoppi
Niko Santala: Käyttäjätunnistuksen ongelmia ja illuusio tietoturvasta
Pro gradu -tutkielma, 110 sivua
Kesäkuu 2008

---

Tutkielmani käsittelee tietoturvan osa-alueista ennen kaikkea käyttäjätunnistusta, kryptologiaa ja tiedon luottamuksellisuutta. Keskityn tekniseen tietoturvaan, joten käsittelen vain hieman esimerkiksi hallinnollista tietoturvaa. Tarkastelen ongelmia konservatiivisella eli vainoharhaisella ajatusmallilla. Osoitan useilla esimerkeillä, miten vaikeaa tai mahdotonta tietoturvan takaaminen on edistyneimpiä hyökkääjiä kuten tiedustelupalveluita vastaan. Kerron useita mahdollisia tapoja vakoilla yrityssalaisuuksia, ja esitän kehittyneitä hyökkäystapoja perinteisiä lukkoja ja ohjelmistoja vastaan. Käsittelen käyttäjätunnistuksen nykyisiä tapoja ja niihin liittyviä ongelmia.

Epäsatunnaisuus salasanoissa ja kryptologiaa käyttävissä sovelluksissa on suuri ongelma, mikä helpottaa merkittävästi kryptoanalyysia ja tietojärjestelmiin murtautumista. Useita aidosti satunnaisia salasanoja kun on käytännössä vaikea muistaa. Esitän seitsemän suunnittelemaani pääosin erillisellä laitteistolla toteutettavaa kryptologiaa käyttävää ratkaisua erilaisiin käytännön ongelmiin.

Internetiin kytketyn työ- tai kotikoneen turvallisuuden takaaminen on lähes mahdotonta. Voidaan olettaa, että tiedustelupalveluilla on listoja julkiselle yhteisölle ja valmistajalle tuntemattomista tietoturva-aukoista. Yrityksen palvelinkaan ei ole välttämättä turvassa erityiseltä ohjelmapäivityksiin kohdistetulta hyökkäykseltä, vaikka sille haettaisiin vain digitaalisesti allekirjoitettuja päivityksiä. Lisäksi itse laitteita kuten tietokoneita, näppäimistöjä tai näyttöjä on helppo vakoilla tai manipuloida huomaamattomasti.

Avainsanat ja -sanonnat: käyttäjätunnistus, tiedon luottamuksellisuus, satunnaisuus, kryptologia, kryptoanalyysi, tiedon salaus, tiedustelu, vakoilu, lukot, tamper resistant, biometrinen käyttäjätunnistus .

## Sisällys

# 1. Johdanto

Tutkielmani käsittelee erilaisia käyttäjätunnistuksen menetelmiä ja ongelmia pääosin teknisen tietoturvan kannalta. Keskityn tietoturvan osa-alueista ennen kaikkea tiedon luottamuksellisuuteen. Käsittelen melko paljon kryptologiaa, joten sen perusteista on hyvä tietää osassa tekstiä. Aluksi esittelen fyysisen turvallisuuden kuten lukkojen ongelmia ja sitten ohjelmistoturvallisuuden. Painotan myös tarvetta aidolle satunnaisuudelle salasanoissa ja salausalgoritmien avaimissa. Ilman riittävää satunnaisuutta turvallisinkin salausalgoritmi voidaan murtaa helposti.

Esitän esimerkkejä mahdollisista tiedustelupalveluiden käyttämistä kehittyneistä menetelmistä. Kerron myös erilaisista vakavista tietoturvaongelmista, joita ohjelmistojen yhteydessä usein esiintyy. Ohjelmistojen suunnittelussa kannattaa olla tietoinen tunnetuista huonoista ratkaisuista ja asioista, joista hyökkääjät yrittävät löytää tietoturva-aukkoja. Fyysinen turvallisuus, ohjelmointitavat, käyttöjärjestelmät, palomuurit, henkilökunta ja monet muut asiat vaikuttavat kokonaisuuden turvallisuuteen. Vaikka käyttäjätunnistus ja salausalgoritmit olisivat huipputasoa, niin yksi päivittämätön ohjelmistoversio vaarantaa huonoimmassa tilanteessa kaiken. Yritysten ylläpito sekä yksittäiset käyttäjät oppivat tämän valitettavan usein kantapään kautta.

Esitän seitsemän suunnittelemaani laitteiston, ohjelmiston ja kryptologian yhdistävää ratkaisua. Entistä kehittyneemmät tietoturvaratkaisut tulevat myös osaksi perinteisiä sovelluksia, minkä voi huomioida ohjelmistojen suunnitteluvaiheessa. Kuluttajakäyttöön on ilmestynyt uusia kohtuuhintaisia ja jopa ilmaisia teknisiä ratkaisuja. Ihmiset tekevät yhä enemmän ostoksia verkossa, käyttävät web-pankkeja ja tarvitsevat tietoturvatuotteita. Tutkielmani on tehty melko konservatiivisella eli vainoharhaisella ajatusmallilla. Käytän kuitenkin hyökkäysmuodoissa pääasiassa vain sellaista tekniikkaa, joka on jo todistetusti olemassa tai on mahdollista toteuttaa nykytekniikalla. Esimerkkini osoittavat, miten mahdotonta järjestelmien tietoturvan takaaminen lopulta on.

Esittämäni hyökkäystekniikat ovat melko edullisia ja siten realistisia. Tiedustelupalvelun huippusalaista elektronista vaatteisiin tarttuvaa tiedustelupölyä, hammaspaikkojen radiolähettimiä, nanobioteknistä haittakoodia tai vastaavia en siis käsittele. Mielikuvitus on kuitenkin tärkeimpiä työkaluja uusien hyökkäystapojen kehittämisessä ja niiltä suojautumisessa. Ei liene sattumaa, että NSA:n Yhdysvaltalaisille opiskelijoille suunnatussa rekrytointimainoksessa on mainittu ainakin vuoteen 2007 asti kahteen kertaan peräkkäin sanat intelligence ja imagination.

# 2. Illuusio turvallisuudesta

Tässä luvussa romutan hieman lukijan turvallisuuden tunnetta melko turvallisena pidettyihin tekniikoihin. Tämä on tarpeellista, jotta turvatekniikoiden parantaminen olisi perusteltua. Aluksi käsittelen perinteisiä tekniikoita kuten lukkoja ja myöhemmin tietokoneita ja niiden ohjelmistoja.

## 2.1. Fyysisen turvallisuuden ongelmia

Lukoilla, valvontakameroilla ja muilla perinteisillä turvatekniikoilla pyritään estämään tai vähentämään todennäköisyyttä muun muassa varkauksille, ilkivallalle ja tiedustelulle. Käsittelen pääosin tiedon luottamuksellisuutta, joten keskityn ennen kaikkea lukkoihin ja muuhun tekniikkaan, jolla varsinaisesti rajataan käyttäjän pääsy tiettyyn tilaan. Näin yritetään taata tiedon eheys ja luottamuksellisuus. Jos fyysinen turvallisuus on heikkoa, niin ohjelmistoja ja laitteistoja voidaan manipuloida, ja tiloihin voidaan asentaa vakoilulaitteita.

### 2.1.1. Turvattomat lukot

Perinteinen 1800-luvun puolivälissä kehitetty [Yale, 2008] *haittatappeihin perustuva lukkotyyppi* (pin tumbler lock) on käytössä hyvin monessa paikassa kuten toimistokalusteiden lukoissa, autoissa ja marketeissa myytävissä edullisissa riippulukoissa. Tällaisessa lukossa on jousia, jotka pitävät pientä ääntä, kun avaimen työntää lukkoon. Avain on yleensä rakenteeltaan melko ohut. Kokeilin viime joululomalla tiirikoida erästä tyypillistä riippulukkoa, jonka omistan. Valitsin yhden pienen ja yhden suuremman ruuvitaltan vanhasta edullisesta kuuden elektroniikkataltan sarjasta. Yllätyin melkoisesti, miten helppoa lukon avaaminen oli näin amatöörimäisillä työvälineillä. Aikaa kului ehkä muutamia kymmeniä sekunteja. Kokeilin tiirikoida sen muutaman kerran uudestaan. Parhaimmillaan sain sen auki kahdeksassa sekunnissa. Kokeilin myös toista erimerkkistä riippulukkoa, joka aukesi myös usein alle minuutissa. Kolmannen erimerkkisen lukon sain kerran auki muutamassa minuutissa. Vanhan disketilaatikon lukon saan auki usein noin muutamassa sekunnissa siitä, kun laitan ruuvitaltat lukon suulle.

Yhdestä autosta sain oven auki teelusikan varrella. Lähtipä auto myös käyntiin samalla kehittyneellä välineellä. Toisen auton oven sain auki erimerkkisen auton avaimella. Molemmat testit olivat tietenkin luvallisia. Turvallisempi lukko saadaan auki ikkunan tiivisteiden kautta nostamalla oven tappi tai rikkomalla lasi. Näistä syistä vakuutusyhtiöt eivät erityisesti pidä siitä, että rahaa tai arvotavaraa kuten äänentoistolaitteita säilytetään autossa. Uusien autojen varsinainen varastaminen on todennäköisesti vaikeampaa ja lukot

aiempaa laadukkaampia. Kaikki nämä ja monet muut haittatappilukot aukeaisivat todennäköisesti helpommin erikoisliikkeistä ostettavilla tiirikoilla, tarkemmalla perehtymisellä eri tiirikointitekniikoihin ja harjoittelulla. Tämä sopii erikoiseksi tarkkuutta vaativaksi harrastukseksi, kunhan ei riko lakia avaamalla luvatta muiden lukkoja tai kantamalla koko ajan mukanaan murtautumisvälineitä.

Jos tiirikointia alle tunnin harjoitellut saa tiirikoitua vaikka tavallisen reppuun mahtuvan kassalippaan auki alle viidessä minuutissa ilman varsinaisia tiirikoita, niin kyseistä lukkotekniikkaa voi pitää turvattomana. Toki osa lukoista vaatii käytännössä varsinaisia tiirikoita. Euroopassa valmistetuissa lukoissa on usein kapeampi ura [Blaze, 2003], minkä vuoksi en saanut yhtä pientä Abus-merkkistä riippulukkoa auki noilla välineillä. Todennäköisesti lukko on myös työstetty laadukkaammin tarkemmalla toleranssilla, jolloin sen avaaminen on vaikeampaa. Lukkomekanismia voi lisäksi parantaa monilla tiirikointia vaikeuttavilla tekniikoilla [Tobias, 2007] [Blaze, 2003], mutta silti en suosittelisi tätä lukkotyyppiä tärkeisiin kohteisiin. Tunnettujen lukkovalmistajien lukot ovat kuitenkin hieman turvallisempia kuin melkein nimettömät Kaukoidässä valmistetut, joita näkee marketeissa.

Lukkotyypin ongelma on siinä, että näitä halpoja lukkoja on niin monessa paikassa. Viimeksi näin niitä kaupan kassan tupakka- ja arpalokerikoissa. Myös eräässä pankin toimipisteessä näytti olevan rahaa tai muuta arvokasta haittatappilukkojen takana, mutta pankin lukot ovat todennäköisesti laadukkaita ja vaikeammin tiirikoitavia. Toisessa saman pankkiketjun suuremmassa toimipisteessä käytettiin turvallisempia Abloyn lukkopesiä. Jos asut kerrostalossa, niin monessako kellari- tai ullakkovaraston ovessa on perinteinen messinkirunkoinen marketista ostettu riippulukko? En mainitse tässä omia havaintojani poliittisen korrektiuden nimissä. Ongelmia aiheutuu myös silloin, jos haittatapin jousi menee poikki, jäätyy tai jää jumiin liasta kuten autoissa talvella. Silloin lukko ei avaudu.

Luottamuksellisuuden kannalta lukkotyyppi on ongelmallinen sen suhteen, että suljettuun tilaan pääsee toistuvasti yksinkertaisilla välineillä melkein kuin varsinaisella avaimella. Esimerkiksi Yhdysvalloissa suurin osa kotiovien lukoista perustuu tähän tekniikkaan [Blaze, 2003]. Koska avainprofiileja on käytössä vain rajallinen määrä, niin viranomaiset ja tiedustelupalvelut pääsevät melkein kaikista ovista vain muutamalla erityisellä *lyöntiavaimella* (bump key). Avain on tehty siten, että se aiheuttaa haittatappien ponnahtelun holtittomasti urassaan, jolloin ovea ei tarvitse edes osata tiirikoida perinteisesti kuten yllä kerroin.

Turvaton lukkopesä voidaan korvata toisella, jos luottamuksellisuutta tarvitaan enemmän. Joskus on murtoturvallisuuttakin tärkeämpää tietää, että kukaan ei ole käynyt tilassa luvattomasti. Omatekoinen viritys on lisätä esimerkiksi koukut tai hakaset arkistokaappiin tai oveen. Koukkuun laitetaan turvallisempaan tekniikkaan perustuva riippulukko. Tiedustelupalvelu voi kyllä periaatteessa rikkoa lukon ja laittaa tilalle dummy-lukon, jonka sarjoitukseen sopii enemmän tai vähemmän mikä tahansa saman profiilin avain. Tätä voi kutsua nimellä Dummy Lock Replace Attack. Tässä on se ongelma, että turvallisinkin mekaaninen lukko on altis tälle hyökkäykselle, jos lukkopesää ei testata säännöllisesti satunnaisella joukolla muita saman profiilin avaimia, joiden ei kuulu tietenkään avata lukkoa. Tällä välin alkuperäisestä rikotusta lukosta on selvitetty sarjoitus, luotu vain alkuperäisellä avaimella avattava kopio ja vaihdettu lukko uudestaan. Lukkoa pitäisi siis testata päivittäin. Hyökkäystapa on helppo riippulukkoa vastaan, mutta kiinteän lukon vaihtaminen vaatii enemmän osaamista. Tällainen hyökkäys ei toimi kassakaappeja vastaan, joissa lukkopesä on suojassa poralta.

Myös elektromekaanisen lukon avaimen täytyisi varmistua lukkopesän luomalla digitaalisella allekirjoituksella siitä, että lukko on aito. Jos se ei ole aito, niin avain piippaa ja vilkuttaa merkkivaloa. Elektromekaanisen lukossa täytyy siten olla tamper resistant -tekniikkaa. Lukko ylikirjoittaa salaiset avaimensa silloin, jos sitä yritetään avata. Yleisesti luottamuksellisuuden kannalta oleellisinta on lukkopesän mekanismi. Riippu- tai muun lukon lujuus parantaa vastaavasti murtoturvallisuutta, joten lukon fyysinen koko täytyy valita käyttötarkoituksen mukaan. Murtoturvallisuuden suhteen kiinnityksen täytyy vastata lukon turvallisuutta. Muuten esimerkiksi murtovarkaat tuhoavat kiinnityksen tai saranat eikä lukkoa.

### 2.1.2. Turvallisten lukkojen ongelmia

Suomalaiset saavat olla ylpeitä kiinteistöissä käytettävistä Abloy-lukoistaan. Ne perustuvat *haittalevyihin* (disk tumbler), joten mitään jousia ei käytetä. Tällaisia lukkoja on paljon vaikeampi tiirikoida ja ne kestävät paremmin likaa, vettä ja pakkasta. Vaikka lukkoseppä saisikin lukon tiirikoitua, niin lukon tiirikointi vaatii huomattavasti enemmän ammattitaitoa kuin haittatappilukot. En löytänyt Internetistä varmoja ohjeita edes nykyaikaisten Abloy Classic -lukkopesien avaamiseen. Lukon poraamista tai muuta murtamista ei lasketa tähän, koska pankkiholvinkin saa kyllä lopulta auki, jos on tarpeeksi aikaa. Tiirikoinnin estäminen on luottamuksellisuuden kannalta oleellista.

Korkeaa turvatasoa vaativissa kohteissa voidaan käyttää esimerkiksi Abloy Protec -lukkopesiä, joissa haittalevyjä ei voi liikutella yksitellen vaan ne luk-

kiutuvat [Abloy, 2008]. Näiden lukkojen tiirikointia pidetäänkin tällä hetkellä yleisesti melkein mahdottomana. Vielä turvallisempi vaihtoehto on Abloy Protec Cliq, joka on muuten samanlainen, mutta se sisältää lisäksi elektronisen avaimen. Esitteessä mainitaan, että avaimessa käytetään DES-salausalgoritmia kaksisuuntaiseen liikennöintiin. Olisi mielenkiintoista arvioida tuotteen kryptologisen toteutuksen turvallisuutta. 56-bittistä DES-algoritmia voi pitää sinällään nykyään turvattomana, mutta jos salattavaa tietoa olisi yhteen suuntaan vain yhden lohkon verran, ja salauksessa käytettävä 56-bittinen avain sekä 64-bittinen selväkieliteksti olisivat aidosti satunnaisia, niin kryptoanalyysin suorittaja ei tietäisi mitä etsiä. Hän kun ei keksisi, mikä selväkieliteksti on oikea.

Kryptoanalyysia helpottavaa epäsatunnaisuutta voi olla esimerkiksi avaimen tai lukon muodostaman sessioavaimen satunnaisuudessa eli satunnaislukugeneraattorissa, avaimen ID:ssä tai salatun protokollan otsikkotiedoissa. Heikkouksia voi löytyä siitä, miten DES-sessioavain välitetään toiselle osapuolelle epäsymmetrisellä salausalgoritmilla kuten RSA:lla. Jos sessioavaimen brute force -hyökkäys yhdistetään protokollan heikkouteen sessioavaimen välityksessä, niin sessioavain ja siten DES-paketin sisältö saattaa selvitä. Jos fyysisen avaimen epäsymmetrinen julkinen avain voidaan olettaa olevan tiedossa, niin sillä voidaan salata esimerkiksi jokainen sessioavaimen kombinaatio, kunnes löydetään oikea sessioavain. Jos epäsymmetristä salauskertaa pidettäisiin muun muassa Montgomery multiplierin sisältävillä erikoispiireillä vaikka 4000 kertaa vastaavaa symmetristä salauskertaa hitaampana, niin tämä aiheuttaisi lisälaskentaa 2^12 eli yhteensä enintään 2^68 suhteutettua DES-salauskertaa.

Jos avaimen tai lukon lähettämä salattu selväkieliteksti sisältää edes jotain tunnettua epäsatunnaista dataa ja käytössä olisi vain 56-bittinen DES, niin hyökkäys helpottuisi merkittävästi. Tarvitaan vain ohjelmoitaviin FPGA (field programmable gate array)-siruihin perustuva alle 10000 dollarin pöytäkone [COPACOBANA, 2008] ilman erityistä tallennustilaa ja enintään 2^56 DES-salausoperaatiota. Keskimäärin oikea avain löytyy 2^55 DES-salausoperaatiolla. Tiedustelupalveluilla on taatusti vielä huomattavasti nopeampia ja kalliimpia laitteita kyseiseen tarkoitukseen. Jos käytössä on 3DES kuten muutamassa paikassa väitetään [Chen, 2008] ja esimerkiksi protokollassa ei olisi heikkouksia, niin sen murtaminen vaatii meet-in-the-middle-hyökkäyksen [Schneier, 1996] vuoksi 2^112 kaksinkertaista DES-salausoperaatiota sekä 2^56 yksinkertaista DES-salausoperaatiota. Tämä vaatii kuitenkin sen, että yhden 3DES-lohkon selväkieliteksti tiedetään kokonaisuudessaan. Näin voisi olla, jos protokolla lähettäisi esimerkiksi otsikkotietoja. Lisäksi tarvittava hakutaulu vaatisi tilaa ilman rainbow tablen [Oechslin, 2003] tai muun

kaltaisia tunnettuja tai julkaisemattomia tekniikoita 2^56 * (64 / 8) / 10^12 = 576461 teratavua pelkän tallennetun datan osalta ilman hakutaulun ja muun kirjanpidon vaatimaa ylimääräistä tilaa.

Ajatellaan että DES:in sijaan käytettäisiin esimerkiksi salausta, jossa selväkieliteksti olisi salattu ensin kahtena 64-bittisenä lohkona IDEA:lla (International Data Encryption Algorithm) ja tulos yhtenä lohkona AES:llä (Advanced Encryption Standard). IDEA:n salausavain on 128-bittinen, joten sen kokoinen hakutaulu tarvitaan. Molemmat salatut lohkot eli 128-bittiä salakielitekstiä tallennetaan hakutauluun, jos tunnetut selväkielitekstiä sisältävät lohkot eivät ole identtisiä. Muuten 128-bittisestä avainavaruudesta tulisi järjettömän paljon eli keskimäärin 2^64 = 18446744073709551616 viittausta yksittäiseen 64-bittisen hakuavaruuden alkioon, koska lohkosalaajien ja hash-funktioiden suunnittelussa *lumivyöryvaikutus* (avalanche effect) on keskeistä. Schneier esittää kirjassaan [Schneier, 1996] havainnon, joka pätee myös tähän esimerkkiin. Jos olisi jotenkin teknisesti mahdollista tallentaa yksi bitti tietoa jokaiseen alumiiniatomiin, niin tällainen tallennuslaite vaatisi pelkästään puhdasta alumiinia yli kuutiokilometrin. Toisin sanoen poralle tai aina niin tehokkaalle *kumiletkukryptoanalyysille* (rubber-hose cryptanalysis) tulisi käyttöä.

Palataan 3DES-esimerkkiin. Uudemman rainbow table -tekniikan [Oechslin, 2003] avulla voidaan vähentää merkittävästi hakutaulun muistinkulutusta, mutta se vaatisi liikaa ylimääräisiä salausoperaatiota siinä vaiheessa, kun tehdään hakutauluun enintään 2^112 hakua. Toki jos tarvitaan kompromissi 168-bitin ja 112-bitin brute force -hyökkäyksen sekä muistinkulutuksen välillä, niin lyhyitä ketjuja voisi miettiä. Kyseistä tekniikkaa käytetään nykyään erityisesti hash-funktioihin perustuvien salasanojen murtamiseen. Sitä voi käyttää myös symmetristen salausalgoritmien murtamiseen kuten alkuperäisessä artikkelissa. Tekniikka säätää merkittävästi muistia laskentatehon kustannuksella käyttämällä pitkiä rekursiivisia ”ketjuja”, jotka muodostetaan symmetrisen salausalgoritmin tapauksessa salaamalla sama selväkieliteksti yhä uudelleen aina seuraavalla salausavaimella. Salausavain muodostetaan rekursiivisesti edellisen salakielitekstin biteistä vähennysfunktiolla, jossa tässä tapauksessa 64-bittisestä salakielitekstistä muodostetaan 56-bittinen avain. Ketjusta tallennetaan vain ensimmäinen ja viimeinen alkio, jolloin sen välissä olevat alkiot voidaan laskea myöhemmin ensimmäisen alkion perusteella uudelleen täysin deterministisesti aina, kun on löydetty osuma viimeisestä alkiosta.

Ehkä koodissa on buffer overflow -bugi, jolla koko protokolla voidaan ohittaa ja siten esimerkiksi avata ovi ilman oikeaa avainta ja lokikirjoitusta. Tällaiset keinot ovat aina ensisijaisia ennen kryptoanalyysia. Ehkä lukko tai avain on altis TEMPEST-hyökkäykselle [NSA, 1982], jolloin esimerkiksi osa sa-

lausavaimesta saadaan selville. Pienikin virhe jollain toteutuksen tai suunnittelun osa-alueella helpottaa mahdollisesti kryptoanalyysia merkittävästi ja siten esimerkiksi avaimen kopiointia ilman laitteiden manipulointia eli pelkän salakuuntelun avulla. Nämä olivat esimerkkejä siitä, miten perusteellisesti järjestelmiä yritetään murtaa.

Tiedustelupalveluilla on aikaa, tietämystä, motivaatiota ja resursseja purkaa laite tai ohjelmisto osiin kehittyneimmillä tekniikoilla ja kehittää kiertotie sen mahdollisimman helpolle avaamiselle. Muun muassa suojelupoliisi, lukko- ja laitevalmistajat sekä salaus- ja virustorjuntaohjelmistojen valmistajat saavat leikkiä loputonta kissa ja hiiri -leikkiä erilaisten poliittisesti epäkorrektien organisaatioiden kanssa. Suomessa tapahtuvaa tiedustelua vähentää varmasti se, että yhteiskuntamme on vakaa ja emme tavoittele esimerkiksi ydinaseita. Jos asuisimme Pohjois-Koreassa, niin jokainen kaapattu viestimme analysoitaisiin tarkasti. Tiedustelua lisää maamme teknologiakeskeisyys. Vähintään yhden tiedustelupalvelun työntekijä käy melko varmasti läpi tämän dokumentin siten, että kiinnostavat avainsanat on korostettu esimerkiksi punaisella. Tutkielmassani kun on melkoisesti kansainvälisiä avainsanoja. Jos dokumentti on tarpeeksi kiinnostava, niin se käännetään kyseiselle kielelle.

Abloyn lisäksi muita turvallisempia lukkoja ovat esimerkiksi osa Abuksen malleista ja erityisellä varauksella Medecon lukot [Tobias, 2007]. Jos hyökkääjällä on aikaa ja tarpeeksi asiantuntemusta, niin ehkä hän onnistuu poraamaan lukon auki, kopioi salaiset tiedot, purkaa rikotun lukon ja selvittää sarjoituksen, asentaa uuden lukon samalla sarjoituksella ja siivoaa jälkensä. Aiemmin kuvaamani dummy-lukko on helpompi tapa. Yleisavaimien olemassaoloon liittyy myös omat riskinsä, jolloin esimerkiksi kahdesta eri avaimesta tai lukkopesä irrottamalla voidaan päätellä yleisavain. Onkohan tiedustelupalveluilla käytössään tekniikkaa, jolla mekaanisen lukkopesän sarjoitus on mahdollista selvittää läpivalaisulla?

Turvallisinkaan puhtaasti mekaaninen lukkomekanismi ei ole kuitenkaan täysin turvassa tiirikointia muistuttavilta tekniikoilta. Mallisuoja estää mahdollisesti vaikka työntekijää teettämästä avaimesta kopiota. Voidaan kuitenkin olettaa, että esimerkiksi rikollis- ja tiedusteluorganisaatioilla on osaamista ja resursseja valmistaa avain vaikka pelkän valokuvan perusteella. Joskus näkee sanomalehdissä ihmisten haastatteluja, joissa näkyy kaulassa tai muualla avaimen sarjoitus melko tarkasti. Avaimen kopioiminenhan vaatii metallin tarkkaa työstämistä, mitä ei voitane pitää ylivoimaisena esteenä suurille organisaatioille. Avaimesta voisi ottaa yhdellä tai useammalla lukon lähelle sijoitetulla vakoilukameralla salaa videokuvaa siitä kohtaa, jossa käyttäjä laittaa avaimen lukkoon. Pimeässä kyseisen kriittisen kohdan voisi valaista voimak-

kaalla infrapunavalolla tai muulla valon ihmissilmälle näkymättömällä aallonpituudella. Tällöin avaimen profiili tulee valaistua sivusuunnasta. Hyökkäystä voi kutsua nimellä Photographic Spy Cam Key Copy Attack.

Hyökkäystapa mahdollistaa myös kassakaappien ja pankkiholvien avaamisen, jos käytetään esimerkiksi avainta, elektronista numerokoodia tai pyöritettävää lukkoa. Pankkiholveissa onkin syytä käyttää vähintään aikalukkoa, jotta holviin ei pääse keskellä yötä. Lisäksi voidaan käyttää esimerkiksi biometristä tunnistinta ja elektronista avainta. Vainoharhainen voi peittää avaimen tai lukon vaikka paksulla kankaalla silloin, kun avaa lukon. Vakoilukameroita voi löytää muun muassa niiden pienestäkin linssistä tapahtuvan heijastuksen avulla. Eräs tällainen tuote on nimeltään SpyFinder. Yksikään tällainen pelkästään optiikkaan perustuva laite ei toimi silloin, jos tiedustelupalvelu on muuttanut kameran toimintaa yksinkertaisella tavalla. Kun laitteen ei ole tarvitse olla käytössä, niin linssin eteen tulee yksinkertainen läppä tai muu vastaava este kuten kokonaan suljettu perinteisistä kameroista tuttu valotusaukko. Näin valo ei pääse linssiin asti. Jos laitteeseen saadaan mukaan kuvantunnistusta ja tekoälyä, niin olisi mahdollista tunnistaa vastavakoilulaite ohjelmallisesti kirkkaana pisteenä, jolloin läppä sulkeutuu välittömästi tietyksi ajaksi.

### 2.1.3. Hälytys- ja kulunvalvontajärjestelmien ongelmia

Hälytys- ja kulunvalvontajärjestelmien valmistajat eivät yleensä mainitse mainonnassaan käyttävänsä tuotteissaan kryptologiaa, jolloin sitä ei myöskään todennäköisesti käytetä. Poikkeus tästä on osa sähköisistä avaimista. Eniten epäilyksiä turvallisuusaukoista herättää se, että järjestelmiä ei voi yleensä arvioida vapaasti kuka tahansa kuten esimerkiksi salausalgoritmeja. Miten voidaan olettaa hälytysjärjestelmän olevan turvallinen, kun usein turvallisilta vaikuttavat ja huomattavalla tarkkuudella kehitetyt salausalgoritmitkin osoitetaan myöhemmin turvattomiksi? Jos mainonnassa tai web-sivuilla ei ole mainittu tuotteiden kehittyneitä piirteitä, niin niitä ei todennäköisesti myöskään ole. Esimerkiksi liiketunnistin yleensä katkaisee tai yhdistää virran johtimessa, mikä aiheuttaa hälytyksen. Jos johtimia manipuloi, niin hälytystä ei tule kyseisestä tunnistimesta. Jos kiinteistöstä katkaistaan kaikki tietoliikenneyhteydet ja häiritään GSM-taajuuksia, niin hälytys ei lähde kiinteistöstä eteenpäin. Jos taas hälytysjärjestelmästä on jatkuva yhteys vartiointiliikkeeseen, niin yhteyttä voidaan manipuloida. Tällaisessa yhteydessä kannattaisikin käyttää kryptologian keinoja.

Osassa kulunvalvontajärjestelmiä avaimet on periaatteessa mahdollista kopioida. Jos avaimessa ei ole mikrosirua kryptologisia funktioita varten, niin avain voi käytännössä vain luovuttaa salaisuuden sellaisenaan kysyttäessä.

Suuri osa avaintyypeistä toimii vieläpä langattomasti, joten avain voidaan kopioida vaikka taskun läpi väenpaljoudessa. Esimerkiksi RFID-tekniikkaan perustuvat avaimet ja vaikka bussikortit ovat yleensä kopioitavissa. Turvallisissa avaimissa on esimerkiksi epäsymmetrinen salainen avain, jolla allekirjoitetaan kulunvalvontajärjestelmän lähettämä satunnainen data, kunhan siihen on ensin lisätty padding-dataa. Sirun olemassaolo ei tietenkään takaa avaimen turvallisuutta. Toteutuksessa, käytetyissä algoritmeissa, protokollissa tai satunnaisuudessa voi olla heikkouksia, jotka tekevät avaimista turvattomia. Ehkä TEMPEST-hyökkäys on mahdollista toteuttaa avaimeen tai kulunvalvontajärjestelmään. Juuri näistä syistä arviointi olisi tärkeää myös fyysisissä turvatuotteissa. Etäluettavia avaimia pystyy lukemaan onneksi vain periaatteessa lyhyeltä etäisyydeltä, mikä vaikeuttaa avainten laitonta kopiointia. Ehkä suunnatulla antennilla lukuetäisyyttä voidaan kasvattaa. Tiedustelupalvelulla on mahdollisesti oma aliorganisaatio tai alihankkijoita, jotka kehittävät markkinoilla oleviin hälytysjärjestelmiin kiertoteitä.

Avainlukijat ovat havaintojeni mukaan yleensä näkyvillä organisaatioiden ulko-ovissa. Tämä on loogista, sillä juuri oven avaamiseenhan ne avaimet on tarkoitettu. Jos avain on kopioitavissa, niin toisen huomaamattoman lukijalaitteen asentaminen alkuperäisen lukijalaitteen yhteyteen on erityisen vaarallinen hyökkäys. Laiton laite lukee avaimen ennen tai jälkeen laillisen lukijalaitteen. Näin saadaan avaimesta kopio. Mahdollinen tunnusluku saadaan vakoilukameroilla. Biometristä tunnistetta ei saada tällä tavoin.

Mahdollisen numeronäppäimistön johtimiin voisi kokeilla syöttää jotain muuta kuin numeroita eli odottamatonta dataa, jolloin hälytysjärjestelmä saattaa jopa kaatua. Lukijalaitteen tai numeronäppäimistön päälle voisi myös asentaa identtisen näköisen mutta muutaman millimetrin suuremman valelaitteen. Kun hälytyksen laittaa päälle, niin laite käyttäytyy täsmälleen kuin alkuperäinen. Käyttäjä luulee, että hälytys on mennyt päälle. Hyökkäystä voi kutsua nimellä Alarm System Panel Clone Attack. Hyökkäys on mahdollista tunnistaa työntömitalla, jota kukaan ei ole päässyt manipuloimaan.

### 2.1.4. Satunnainen väkivalta ja terrorismi

Kaupunkien keskustat, kauppakeskukset, asemat, kadut ja monet muut tilat ovat täynnä valvontakameroita, lentokentillä ollaan vainoharhaisia, siellä täällä parveilee vartijoita ja Internetiä täytyy sensuroida. Mistä tulee kaikki tämä epäluottamus?

Otetaan esimerkkinä lentoturvallisuus. Lentoturvallisuuden voidaan olettaa olevan hyvällä tasolla, koska sitä on painotettu vahvasti jo vuosia. Siitä huolimatta koneet on mahdollista ampua alas maalta tai mereltä erityisesti sil-

loin, kun ne nousevat tai laskeutuvat. Niitä on myös mahdollista sabotoida lentokentällä lastauksen tai muun operaation ohessa. Aseiden tai räjähteiden salakuljettaminen koneisiin ei ole muutenkaan mahdotonta. Jos joku todella ottaa asiakseen lentokoneen tuhoamisen, niin hän voi kätkeä ne geelimäiset räjähteet kengänpohjiin ja huolella valmistettuihin vartaloa myötäileviin ohuisiin geelipusseihin, jotka eivät näy vaatteiden läpi. Vaikka matkustajat riisuttaisiin kokonaan tai nähtäisiin vaatteiden alta epäilyttävät esineet uusilla tekniikoilla kuten ThruVision T5000:lla, niin järjestelmä ei ole silti aukoton. Äärimmäisyyksiin menevä terroristi voisi vaikka kätkeä geelipussit leikkauksilla ihonsa alle, mutta laitteita pystyy todennäköisesti hämäämään muutenkin. Tärkeintä on se, että suurta konetta ei ole mahdollista kaapata ja käyttää aseena erityisesti ydinvoimalaa vastaan. Tavalliset rakennukset eivät ole kriittisiä, vaikka niillä olisi suurikin symbolinen arvo.

Koneen ampuminen alas armeijan toimesta olisi pieni hinta siitä, että satojen neliökilometrien alue altistuisi radioaktiivisuudelle. Vaikeinta on tietenkin varmistua siitä, että kone on yleensä kaapattu ja määränpäänä ydinvoimala. Alas ammutusta koneesta kun tulisi joka tapauksessa melkoinen mediamyrsky. Kaikenlaiset viiveet viranomaisten organisaatioissa ja niiden välillä hidastavat reaktioaikaa. Kone poikkeaa yleensä merkittävästi reitiltään todennäköisesti muusta syystä, ja radio voi olla mykkä tai väärennetty. Koska radioon ei voi täysin luottaa, niin sen yli voitaisiin käyttää yksinkertaisiakin protokollia kuten kertakäyttöistä lentäjäkohtaista väärää ja oikeaa jaettua salaisuutta, jolloin kukaan muu kuin lentäjä ja lentokenttä eivät tiedä onko salaisuus oikea vai väärä. Oletko unohtanut koskaan salasanaa tai pankkikortin tunnuslukua? Lisäksi mykkä radio ja salasanaparin pysyminen salassa lentokentän ja lentäjän kesken on yksi ongelma. Ehkä lentokoneen muista järjestelmistä kokonaan eriytetty satelliittipuhelin tarjoaisi yhden vaihtoehtoisen kommunikaatiokanavan.

Jos oletetaan että lentoturvallisuus olisi aukoton ja kouluissa ei olisi enää edes teoreettista mahdollisuutta ammuskella, niin miten muu yhteiskunta olisi suojattu? Keksitkö yhtäkään julkisuuden henkilöä mistään maasta, joka olisi täydellisesti eli teoreettisesti suojassa murhalta? Fidel Castro on julkisen tiedon valossa selviytynyt murhayrityksiltä, mutta hän lienee syystäkin erityisen vainoharhainen. Miten tällä hetkellä on varauduttu esimerkiksi siihen, että joku menee satunnaisesti valittuun päiväkotiin, teatteriesitykseen, hiihtokeskukseen, yökerhoon tai uimarannalle ja alkaa ammuskella? Miten pystytään estämään henkilö, joka kävelee kahden kotitekoisia räjähteitä täynnä olevan matkalaukun kanssa ostoskeskukseen erityisen vilkkaan alennuspäivän aikana. Suljetussa tilassa räjähdysaalto, metallinsirpaleet, lentävät esineet ja rakenteiden romahtaminen aiheuttaisivat niin paljon ruumiita, että lentokone jäisi nopeasti

toiseksi. Ihmiset kasautuisivat ja tukehtuisivat paniikissa ulosmenoteihin. Tällaisissa tilanteissa valvontakameroita voisi käyttää vain tilanteen hahmottamiseen.

Kun kyseiset hyökkäykset tehtäisiin mahdottomiksi, niin aina löytyy jokin siviilikohde, jota ei ole suojattu yhtä hyvin puhumattakaan täydellisesti. Miksi hyökätä turvalliseen kohteeseen, jos löytyy helpompikin? Näin äärimmäisiin tekoihin ryhtyviä ihmisiä on kuitenkin mitä ilmeisimmin niin vähän, että he eivät saa onnistuessaankaan aikaan ihmislajin tai kansan kannalta merkittävää tuhoa, vaikka kaikki teot laskettaisiin yhteen. Poikkeus on henkilöt, jotka pääsevät tavalla tai toisella käsiksi ydintekniikkaan tai bioaseisiin. Historiallisesti suurin riski on suurten valtioiden tai muut mielipuoliset päättäjät, joilla on paljon valtaa. Montako ihmistä on kuollut muun muassa enemmän tai vähemmän turhissa sodissa, uskontojen nimissä, orjuudessa, kansanmurhissa ja keskitysleireillä? Mikä valtio on räjäyttänyt ydinaseita toisen valtion kaupungeissa? Lasketaan koko ihmiskunnan historiasta yhteen kaikki terroristien sekä massa- ja sarjamurhaajien uhrit. Verrataan lukua siihen lukuun, jonka päättäjät ovat saaneet aikaan. Suhdeluvussa on aivan tarpeeksi nollia johtopäätösten tueksi.

Suomessa poliisi, keskusrikospoliisi, suojelupoliisi ja muut viranomaiset ovat vallitsevaan vainoharhaisuuteen suhteutettuna kohtuullisessa tai hyvässä tasapainossa erilaisten uhkien suhteen. Ainakin tavallisella poliisilla on jalat maassa. Yhdysvaltojen ja Englannin vainoharhaisuus muistuttaa puolestaan jo ristiretkiä tai noitavainoja. Suomalaisen kiertueella olevan artistin käännyttäminen rajalta potentiaalisena turvallisuusuhkana on jo melkoista utopiaa. Liiallinen laskelmointi, arviointi ja profilointi tuottavat juuri tällaisia ongelmia.

Yhteiskunnan on mahdoton saavuttaa täysin turvallista tilaa, koska kaaos aiheuttaa aina tapahtumia, joita ei ole mahdollista ennakoida. Koko yhteiskunta perustuu hyvinkin paljon toisten luottamiseen. Millaisia ihmisiä kasvaisi ympäristössä, jossa lapsi kysyisi koulussa: ”Opettaja, miksi ikkunoissa on kalterit, koulun pihamaata ympäröivät muurit ja kotiin mennään panssaroidulla autolla?”. Kannattaa muistaa, että turvallisuusala on nykyään suuri bisnes. Pelko myy tuotteina ja palstoina mediassa. Suomessa korostetaan kovia arvoja suuryritysten vaatimuksesta, mutta toisaalta kuoleman suhteen ollaan pumpulilinnassa. Jos muutama ihminen kuolee hieman eksoottisella tavalla vaikka ulkomailla, niin ainakin valtamedian mukaan pitäisi muuttaa heti lakia eli liioitella. Tarpeeksi pitkällä aikavälillä jokaisen henkiinjäämisprosentti putoaa nollaan [Palahniuk, 1996].

Montako suomalaista kuolee vuosittain liikenneonnettomuuksissa, niin yleiseen alkoholismiin, kotiväkivallan uhreina ja lääkärien virheisiin? Miten

teollisuuden kuluttajille tarjoamat lukuisat kemikaalit kertyvät ympäristöön nykyisiä ja tulevia sukupolvia ajatellen? Esimerkiksi markettien shampoot ovat keskimäärin varsinaista ongelmajätettä. Moniko suomalainen elää laitoksessa tai kotona tarpeettomasti jatkuvassa pienessä pillerihumalassa eli lääkärin määräämässä laillisessa huumausaineriippuvuudessa? Eikö näin rikkaalla yhteiskunnalla ole varaa kodittomien asuntoihin, vaikka kaupungit ovat iltaisin ja öisin täynnä tyhjiä lämmitettyjä kiinteistöjä? Ai niin mutta tilat eivät sovi määräysten mukaan asunnoiksi.

Puolustusvoimat on yhteiskunnan turvallisuuden suhteen mielestäni täysin välttämätön instituutio, koska naapurivaltioiden ja Suomen päättäjät eli siten poliittinen tilanne vaihtuu jatkuvasti. En oikein usko, että joku toinen maa tulisi puolustamaan meitä puolestamme, jos emme itse tekisi mitään. Suomelle yksi uhka on ulkomainen järjestäytynyt rikollisuus.

Esimerkiksi havaintojeni mukaan naiset kuljettavat melkoisen huoletta käsilaukkujaan eivätkä huomaa, jos joku kävelee melkein heidän kantapäillään. Laukun pohjan voi viiltää huomaamattomasti mattopuukolla, jolloin sen sisältö putoaa varkaalle. Lisäksi laukkujen hihnat katkeavat samalla välineellä, jolloin toinen varkaista odottaa mopon äärellä pakomatkaa. Tehokas tapa on myös ajaa mopolla ohi ja riuhtaista laukku mukaan, jolloin hihnan kiinnitys pettää. Varkaita ei välttämättä kiinnosta se, murtuuko uhrin olkapää. Kun jäin eräänä päivänä bussista pois, niin olisin voinut ottaa yhdestä avoimesta käsilaukusta kukkaron sormenpäillä kenenkään huomaamatta. Ammattilaiset avaavat laukun vetoketjun tai napit huomaamatta. Muutama päivä sitten tuli vastaan valokuvaaja, jonka ammattimainen järjestelmäkamera olisi lähtenyt mukaan erityisen helposti.

Näitä tekniikoita käytetään maailmalla ammattimaiseen varasteluun. Rannekellojen, lompakoiden ja muun omaisuuden varastamiseen löytyy omat tekniikkansa. Osa varkaista on äärimmäisen taitavia ja harjaantuneita. Myös esimerkiksi yksittäisen nuoren naisten kaappaaminen pakettiautoon keskellä päivää on todennäköisesti suhteellisen helppoa. Ilman poliisia, keskusrikospoliisia, tullia ja muita viranomaisia kansainvälinen ihmiskauppa ja muu vastaava törky olisi liiankin helppoa.

Dramaattisten esimerkkieni tarkoitus oli esittää, miten käytännössä mahdotonta on taata fyysistä turvallisuutta. Liiallinen ihmisten valvonta eli epäluottamuksen ilmapiiri, ihmisten pitäminen pelkkinä tuotantoyksiköinä ja itsekeskeisyyden ihannoiminen aiheuttavat todennäköisesti pitkällä aikavälillä jo itsessään vakavia turvallisuusuhkia. Täydellistä turvallisuutta voi toki lähestyä mutta ei saavuttaa. Tietokoneiden turvallisuuden takaaminen on vähintään yhtä vaikeaa. Seuraavaksi keskityn lähinnä tietokoneiden ja ohjelmistojen tur-

vallisuuteen. Ei saa kuitenkaan unohtaa sitä, että molemmat aihealueet ovat vahvasti kytköksissä toisiinsa.

### 2.2. Ohjelmistoturvallisuuden ongelmia

Keskustelin muutama kuukausi sitten tietoturvasta erään asiakkaan kanssa. Mainitsin yhtenä uhkana erityisesti ulkomailta tulevan yritysvakoilun. Mainitsin, että kehittyvät suurvallat saattavat varsinaisesti kouluttaa ihmisiä tähän tehtävään. Vakoojat soluttautuvat korkean teknologian yrityksiin tavallisina työntekijöinä. Kehittyviä suurvaltoja kun tuskin kiinnostaa länsimaiden yritys-salaisuudet, vaan lopulta vain oma kansallinen etu. Olin päätellyt tuon itsenäisesti ja en vielä ollut kovin tietoinen, että myös kehittyneimmät teollisuusmaat yrittävät värvätä erityisesti henkilöitä, jotka asuvat jo täällä. Sattumalta noin kuukauden päästä suojelupoliisi ilmoittikin julkisuudessa, että he kouluttavat suuria yrityksiä välttämään yritysvakoilua. Heillä ei ole kuitenkaan resursseja suojella yksitellen kaikkia maan yrityksiä varsinkaan Internetistä tulevia hyökkäyksiä vastaan, joten koulutus on varsin hyvä ratkaisu. Yritykset voivat näin rakentaa itse omia vastakeinojaan. Suojelupoliisin vuosikertomuksien [Suojelupoliisi, 2008] mukaan he ovat varsin hyvin tietoisia tiedustelun tilanteesta. Raporteista näkee, että tiedusteluyritykset ovat tarkasti suunniteltuja ja kohdennettuja.

Tiedustelua tehtailevat käytännössä kaikki suurimmat valtiot. Pienemmät valtiot keskittyvät vastavakoiluun. Erityisen kuuluisia organisaatioita ovat CIA, NSA (National Security Agency) ja MI6. NSA:ta pidetään kryptoanalyysin kirjallisuudessa referenssinä monestakin syystä. Se on muun muassa maailman suurin matemaatikkojen työllistäjä [Schneier, 1996]. Lisäksi he keskittyvät juuri muun muassa kryptoanalyysiin. Nämä ja monet muut vastaavat organisaatiot tekevät kaikenlaisen tiedustelun lisäksi kohdennettuja hyökkäyksiä mahdollisesti yhteistyössä toisten organisaatioiden kanssa silloin, kun jokin salaisuus täytyy saada selville. Kohdennetut hyökkäykset ovat vaarallisimpia, koska esimerkiksi haittaohjelmia torjuvien ohjelmien tietokannassa ei ole näiden hyökkäysten sormenjälkiä. Haittaohjelmat kun on kirjoitettu organisaation sisällä. Tilanne on sama krakkerien kanssa. Hyökkäys voi kohdistua esimerkiksi palomuurin tai sähköpostipalvelimen hyökkäyksiä tunnistavan osan buffer overflow -haavoittuvuuteen. Oulun yliopistossa julkaistiin juuri tutkimus, jonka mukaan suuri osa pakkausalgoritmeja parsivista sovelluksista saattaa sisältää haavoittuvuuksia [OUSPG, 2008]. Toisin sanoen kun haavoittuva virustorjuntaohjelma tutkii sähköpostiviestin, niin palvelin tai työntekijän tietokone vaarantuu.

Tiedustelupalveluilla on todennäköisesti listoja haavoittuvuuksista, joista julkinen yhteisö ei ole tietoinen. Vaikka he käyttäisivätkin vain tunnettuja tietoturva-aukkoja, niin maailma on silti täynnä tavalla tai toisella päivittämättömiä tietokoneita, reitittimiä, palomuureja ja muita laitteita. Olen miettinyt myös mahdollisuutta sille, että he analysoivat verkkoliikennettä, puheluita ja muuta viestintää melko automaattisesti ja tekevät ihmisistä profiileja. Jokainen ihminen ja yritys luokitellaan johonkin muutamista luokista sen mukaan, miten kiinnostavaa tietoa hän käsittelee työssään. Suuri osa maapallon ihmisistä kuuluu luokkaan, joiden tekeminen ei kiinnosta heitä lainkaan. Osa kuuluu luokkaan, joiden työssä saattaa olla jotain kiinnostavaa. Vain pieni osa ihmisistä kuuluu luokkaan, joiden tekemistä seurataan mahdollisimman tarkasti. Rajat menevät todennäköisesti melko pitkälle henkilön työpaikan ja roolin mukaan. Toisin sanoen he keskittyvät oleelliseen ja suodattavat turhan tiedon pois. Heillä on kuitenkin rajallinen budjetti.

Tietokoneelle asennettavissa tietoturvatuotteissa kuten salasanatietokannoissa ja kiintolevyn salaukseen käytetyissä ohjelmistoissa mainostetaan usein salausalgoritmin bittimääriä. Se ei ole kuitenkaan useinkaan niin tärkeää kuin annetaan olettaa. Tietoliikenne on poikkeus, koska erityisesti langattomia verkkoja ja Internetiä yhdyskäytävänä käyttäviä salattuja yhteyksiä voidaan aina salakuunnella. Jotta salasanatietokantaan päästään käsiksi, niin täytyy usein joka tapauksessa manipuloida tai salakuunnella kohdeympäristöä. Näin voidaan asentaa koneelle esimerkiksi ohjelmistoon tai laitteistoon perustuva laite, joka tallentaa näppäinpainallukset (key logger). Jos tietomurto ei onnistu tai tietokone ei ole koskaan verkossa, niin seuraavaksi helpointa on tunkeutua kiinteistöön siten, että siitä ei jää jälkiä. Näin voidaan kopioida kiintolevyn tiedot, manipuloida tietokonetta tai asentaa vakoilulaitteita. Tästä syystä fyysinen turvallisuus on yrityksissä hyvin tärkeää tiedon luottamuksellisuuden kannalta.

Vasta viimeisenä vaihtoehtona voidaan laskea salauksen murtaminen. Harvoin tarvitsee silloinkaan tehdä brute force -hyökkäystä esimerkiksi heikkojen salasanojen vuoksi. Varkaat taas ovat todennäköisesti kiinnostuneita pääsemään eroon laitteista, jos joku vain maksaa niistä. He eivät välttämättä välitä tietokoneiden sisällöstä, ja heikkokin salaus voi olla heille turhan työläs tai asiantuntemusta vaativa toimenpide. Kannattaa huomata, että tiedustelu ei välttämättä kohdistu IT- tai mobiilialan yrityksiin. Se saattaa kohdistua myös esimerkiksi insinööritoimistoihin ja biotekniikan yrityksiin. Kaikki uudet keksinnöt ja tiedot julkaisemattomista tuotteista ovat potentiaalisia kohteita.

Seuraavissa alakohdissa ei ole kiinnitetty niinkään huomiota tietoturva-aukkoihin, joita löytyy jatkuvasti muun muassa web-selaimista sekä sähkö-

posti-, toimisto-, pikaviesti- ja vertaisverkko-ohjelmista. Ne tarjoavat usein helpoimman tavan ohittaa esimerkiksi organisaation palomuurit. Niinpä salaisimmat projektit täytyisi tehdä vähintään koneilla, joita ei ole yhdistetty millään tavalla Internetiin. Vierellä voi olla toinen kone, jota käytetään esimerkiksi sähköpostin käsittelyyn ja web-selailuun. Parhaassa tapauksessa sen vaarantuminen ei ole kovin kriittistä.

### 2.2.1. Viisi esimerkkiä tiedustelusta

Esitän viisi esimerkkiä kehittyneistä tavoista, joilla tiedustelupalvelu saattaa saada käsiinsä rahanarvoista salaista tietoa. Suurilla resursseilla voidaan rakentaa tällaisia kehittyneitä kohdistettuja hyökkäyksiä.

Ensimmäisessä tavassa kiinnostava biotekniikka-alan yritys tilaa työntekijöilleen uusia näppäimistöjä tai kannettavia tietokoneita. Tilaus rekisteröidään tietokantaan. Kaukoidässä sijaitseva tehdas saa tilauksen tietokannasta. Paikallinen tai joku muu tiedustelupalvelu on soluttautunut tehtaaseen tai muuhun sellaiseen asemaan, että laitteiden manipulointi on mahdollista. He huomaavat tämän kiinnostavaksi profiloidun yrityksen tilauksesta. Näppäimistöön tai kannettavan tietokoneen sisälle asennetaan muutettu piiri, joka näyttää painatuksineen kaikkineen ulospäin identtiseltä alkuperäiseen verrattuna. Piiri kerää muistipiiriin kaikki näppäinpainallukset. Se aktivoi myös siihen rakennetun radiolähettimen ja lähettää kerätyt näppäinpainallukset radioteitse sen jälkeen, kun radiovastaanotin vastaanottaa tietyn salasanan.

Sitten tarvitsee vain jättää vaikka tavallinen henkilöauto yrityksen parkkipaikalle, jossa tieto kerätään automaattisesti talteen kannettavalla tietokoneella. Samalla periaatteella voisi manipuloida esimerkiksi pöytäkoneen USB-portteja, piirisarjaa, jolloin kaikkien USB-näppäimistölaitteiden näppäinpainallukset tallennettaisiin. Näppäimistön vaihtaminen ei siis auttaisi yhtään. Esimerkiksi joko PS/2- tai USB-liittimeen asennettavan melko huomaamattoman Keyshark Key Logger -laitteen voi ostaa muutamasta suomalaisesta verkkokaupasta 65 - 80 eurolla. Jos löytyy osaamista elektroniikasta, niin kotikonstein voi avata näppäimistön ja lisätä sen sisälle sopivia komponentteja.

Tiedustelu ei kohdistu välttämättä uusiin tuotteisiin tai keksintöihin. Tiedustelupalvelu voi ottaa kohteekseen minkä tahansa pörssiyhtiön, joka julkaisee lähiaikoina esimerkiksi osavuosikatsauksen. Tiedustelupalvelu on seurannut esimerkiksi metsäyhtiön johtokunnan käyttäytymistä jo aiempina kertoina ennen osavuosikatsauksia. He tietävät, että johtokunta kokoontuu aina edellisenä iltana yhteen muutamista neuvotteluhuoneista. Tämän jälkeen he laskevat kustannuksia asentaa mikrofoneja todennäköisimpään huoneeseen. Yhtiöstä värvätty työntekijä tai väärennetty siivooja asentaa mikrofonit. He voivat jopa

porata yöllä seinän tai katon läpi. Heidän ei tarvitse käyttää edes mikrofoneja, jos esimerkiksi TEMPEST on mahdollinen siksi, että johtokunta tarkastelee osavuosikatsausta turvattomalla näyttölaitteella. Tavallisia näyttöjä kun on varsin mahdollista vakoilla seinien läpi [Kuhn, 2003]. Tällöin he voivat istua turvallisesti toisessa kiinteistössä tai autossa erikoiskahvia tankaten.

Yritysjohtajilla on usein käytössään markkinoiden monipuolisimmat matkapuhelimet, joita he käyttävät vähintään sähköpostien lukemiseen, web-selailuun, dokumenttien lukemiseen sekä kalenterina. Muun muassa nämä ja muut rajapinnat tarjoavat mitä ilmeisimmin tietoturva-aukkoja. Yksikin riittää. Sen avulla markkinoiden monipuolisin eli tiedustelun kannalta turvattomin mahdollinen matkapuhelin muuttuu kuin taikatempusta mikrofoniksi. Muistiakin noissa puhelimissa riittää, ja puhe ei vie nykymittapuussa paljon tilaa, joten se voidaan kätkeä muistialueen tyhjiin varausyksiköihin.

Nyt tarvitaan tietoturva-yhtiön edustaja selittämään, että tämä on epätodennäköistä. Se on epätodennäköistä satunnaisesti valitulle kansalaiselle, mutta huomattavasti todennäköisempää kohdistetussa hyökkäyksessä. Vastassa on määrätietoinen organisaatio, jossa työskentelee runsaasti kryptoanalyytikkoja, matemaatikkoja ja ohjelmoijia. Hyökkäystapa, sen ohjelmakoodi ja tarkat käyttöohjeet on kehitetty etukäteen rauhassa organisaation sisällä yleiskäyttöiseksi muutamalle saman valmistajan puhelinmallille ja tallennettu salaiseen tietokantaan. Matkapuhelinvalmistajia on käytännössä vain muutamia, joten tämä lienee kustannustehokas hyökkäystapa. Sisäpiiritiedon avulla ostetaan tai myydään osakkeita ennen tulosten julkistamista. Toiminnalla voidaan rahoittaa esimerkiksi tiedustelutoimintaa.

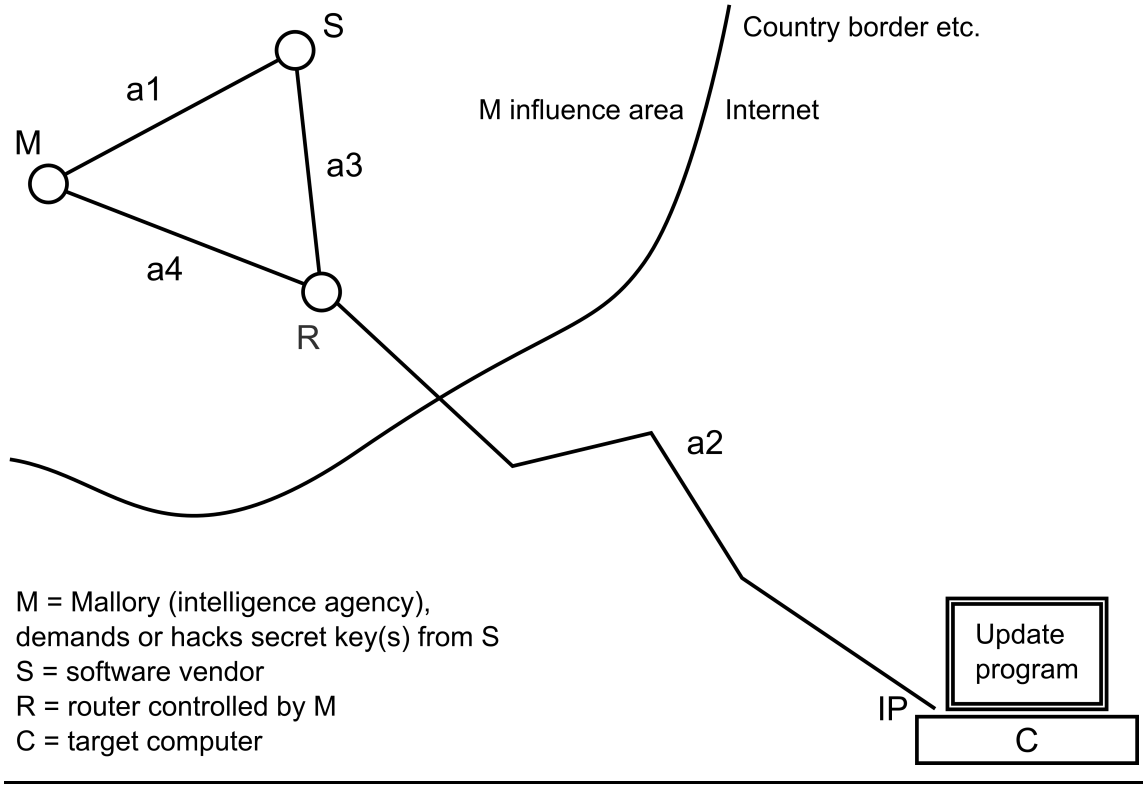
Kannettavien tietokoneiden mikrofoneja on myös mahdollista käyttää salakuunteluun, mutta tällainen tilanne ei ole luottamuksellisuuden suhteen muutenkaan kovin kehuttava. Oletko muuten käyttänyt kannettavaa tietokonetta julkisissa kulkuvälineissä? Jos Windowsin tehtäväpalkki on näkyvissä ja näytössä ei käytetä erillistä tietoturvasuojaa, niin sen kuvakkeista voi mahdollisesti päätellä mitä virustorjuntaohjelmistoa ja palomuuria käytät. Tämä tieto helpottaa myöhempiä hyökkäyksiä ja työkoneen tapauksessa mahdollisesti koko organisaatiota vastaan.

Kolmas hyökkäystapa kohdistuu ohjelmistopäivitysten sertifikaatteihin. Kyseessä voi olla minkä tahansa Internetin yli päivitettävä ohjelmisto mukaan lukien käyttöjärjestelmä. Hyökkäys soveltuu myös tavallisiin web-sivuihin. Esimerkiksi Yhdysvallat on vaatinut useita yrityksiä rajoittamaan salausalgoritmien turvallisuutta vientituotteissaan [Schneier, 1996]. Ehkä NSA tai jokin muu organisaatio on vaatinut lakiin nojaten ohjelmistoyrityksen digitaaliseen allekirjoitukseen käyttämät salaiset avaimet ja luvannut, että niitä ei käytetä

kotimaisia yrityksiä tai yksityishenkilöitä vastaan. Avaimet on voitu saada haltuun myös murtautumalla palvelimelle.

Kuvassa 2.2 on esitetty hyökkäyksen periaate. Tiedustelupalvelu M (Mallory) saa tavalla tai toisella haltuunsa palvelimen ja ohjelmapäivitysten salaiset allekirjoitusavaimet ohjelmistoyritykseltä S (software vendor), mitä kuvaa reitti a1. Kun tietokone C päivittää yrityksen ohjelmiston, niin normaalisti tietoliikenne kulkee Internetissä reittiä a2 + R + a3 päivityspalvelimelle S. Nyt M kohdistaa hyökkäyksen vain tähän yhteen tietokoneeseen. M muuttaa reitittimen R reitityksen siten, että ainoastaan koneen C IP-osoitteesta liikenne kulkeekin reittiä a2 + R + a4 valepalvelimelle M. Kohdepalvelimen IP-osoite pysyy samana ja verkkoviiveetkin voidaan simuloida, jolloin valepalvelin voi sijaita jopa samassa tilassa reitittimen kanssa.

Palvelimen S salaista allekirjoitusavainta käytetään luomaan palvelimen M digitaalinen allekirjoitus. Manipuloidut ohjelmistopäivitykset allekirjoitetaan vastaavasti ohjelmistopäivitysten salaisella allekirjoiusvaimella. Näin väärennettyä palvelinta tai sen sisältämiä päivityksiä on todella vaikea erottaa alkuperäisistä. Certificate authorityn (CA) sertifikaattikin näyttää olevan kunnossa. Käyttäjä tuntee olonsa turvalliseksi, kun ohjelmisto on taas päivitetty, vaikka todellisuudessa koneen tai koko lähiverkon kautta koko yrityksen tiedot ovat vaarantuneet. Hyökkäyksestä voi käyttää nimeä Targeted (Update) Server Certificate and Router Attack.




Kuva 2.2. Targeted (Update) Server Certificate and Router Attack.

Perinteisesti krakkerit saastuttavat tietokoneen C käyttämän nimipalvelimen hakutaulun, jolloin päivitys- tai muun palvelimen IP-osoite voidaan väärentää. Krakkerin suurin ongelma on tällöin sertifikaattien väärentäminen. Hän voi turvautua käyttäjän epätietoisuuteen tai käyttää hyväksi sertifikaattien tarkistukseen liittyvää tietoturva-aukkoa. Tällaisia aukkoja löytyy jatkuvasti eri ohjelmistoista [SecurityFocus, 2008]. Suurin osa ohjelmista ja ohjelmapäivityksistä haetaan kuitenkin vain jostain palvelimelta ilman sertifikaatteja, jolloin hyökkäys yksinkertaistuu. Esittämääni hyökkäystapaa voidaankin käyttää tiedustelussa esimerkiksi sellaisia palvelimia kohtaan, joihin tehdään vain käyttöjärjestelmäpäivityksiä ilman muuta liikennettä Internetiin.

NSA on muuten jäänyt ainakin kerran kiinni verkkoliikenteen keräämisestä Internetin tärkeästä Yhdysvalloissa sijaitsevasta solmukohdasta nimeltään Room 641A. Siellä oli monta laiteräkkiä laitteita ja muun muassa Narus-nimisen yrityksen todella kehittynyttä laitteistoa, jolla pystyy analysoimaan alustavasti käytännössä kaiken valokaapelissa liikkuvan liikenteen reaaliajassa. Mitä kaikkia muita laitteita noissa räkeissä oli? Montako apuohjelmaa tai päivitystä olet asentanut koneellesi viimeisen vuoden aikana?

Neljäs esimerkki liittyy aiemmin mainitsemaani ihmisten profilointiin. Suuria kaupallisia hakukoneita on epäilty keräävän tietoa käyttäjistä. Jos verkkoliikennettä kerätään verkon tärkeässä solmukohdassa esimerkiksi Naruksen laitteilla, niin eihän siihen tarvita minkään yrityksen suostumusta. Useimmat näistä laitevalmistajista toimivat todennäköisesti salaisina alihankkijoina suurille valtioille. Hakuehdot voidaan parsia pelkästä URL-osoitteesta, joka sisältää yleensä hakulauseen selväkielisenä. Kun käytät hakukonetta töissä, niin hakulauseet saattavat kertoa hyvinkin paljon ammatistasi ja projektista, jossa työskentelet. Tietoja saattaa ponnahtaa esiin myös esimerkiksi mahdollisista ECHELON-järjestelmän tiedustelutiedoista kuten matkapuhelimesi numerosta ja sijainnista. Ehkä luet web-sähköpostia sekä töissä että kotona. Tämän jälkeen kotikoneesi ja työkoneesi liitetään samaan profiiliin. Tiedonmurusista luodaan anonyymi hierarkkinen profiili, jota täydennetään vähitellen. Ennemmin tai myöhemmin profiili saa todennäköisen henkilöllisyyden. Ehkä osalla tietoa on ajallinen konteksti. Jos olet esimerkiksi sairaana, niin et käytä konettasi töissä ja matkapuhelimesi on kotona. Ehkä olet helpottanut profiiliverkoston luontia listaamalla kaikki tuttavasi sosiaalisessa verkostopalvelussa.

Viides esimerkki on mahdollinen tiedusteluyritys. Kun kirjoitin tätä tutkielmaa, niin etsin verkosta paljon muun muassa lohkosalaajiin ja satunnaisuuteen liittyvää aineistoa. Tiedon salaukseen ja aseteknologiaan liittyvät tuotteet ovat aina tiedustelupalveluille erityisen kiinnostavia. Sen voi päätellä tiedusteluhistorian perusteella ja suojelupoliisin raportista [Suojelupoliisi,

2008]. Työpaikkani muuten sijaitsee samoissa koordinaateissa kuin eräs turvallisuusteknologiaa kehittävä yritys. Minulle tuli puolenyön aikaan venäläisestä numerosta kaksisanainen tekstiviesti, jonka sisältö on käännettynä ”Olemme kotona”. Väärä vastaanottaja on yksi mahdollinen selitys, mutta en ole ennen sitä tai sen jälkeen saanut Venäjältä tai yleensäkään venäjänkielisiä viestejä.

Viestin lyhyys on jo sinänsä epäilyttävää, mutta tiedustelupalvelut eivät kyllä lähetä clandestine- tai covert-operaatioiden koodiviestejä vääriin numeroihin. Tällainen HUMINT (HUMan INTelligence)- ja SIGINT (SIGnals INTelligence) -tiedustelun yhdistelmä on sinänsä yksinkertainen mutta varsin välkky, ja se muistuttaa kryptoanalyysin keinoja. Jos tutkin kryptologiaan liittyvää aineistoa, niin he voivat todennäköisen psykologisen profiilini perusteella päätellä, että haluan tietää uteliaisuuttani tuon viestin suomennoksen. Lisäksi he laskevat, että olen tuohon aikaan väsyneempi ja siten alttiimpi manipuloinnille. Jos syötän esimerkiksi hakusanat ”russian” ja ”translat*” tai vastaavasti hakusanoja puhelinnumeron maakoodin selvittämiseksi, niin todennäköisesti kukaan muu Suomessa ei tee samoja hakuja varsinkaan yöllä. Näin he voivat varmistua, että IP-osoitteeni, puhelinnumeroni ja siten henkilöllisyyteni täsmäävät. Samalla he saavat tietää osaanko venäjän kieltä. Pelkästään kyrillisten aakkosten kanssa saa tehdä töitä, jotta suomennos onnistuu Internetin käännöspalveluissa. Viestissä kun melkein kaikki kirjaimet ovat erikoismerkkejä.

Venäjän alkeiden osaamisella on merkitystä erityisesti länsimaiden tiedustelupalveluille, kun he haluavat tietää kehitänkö salausjärjestelmiä venäläisille. Lisäksi käännöksen tekeminen vaatii töitä noiden aakkosten kanssa, jolloin käyttämäni menetelmät ja ajankäyttö kertovat jotain minusta. Sekin kertoo tartuinko syöttiin, ja se kerronko yrityksestä tässä tutkielmassa. Jos en yrittäisi etsiä käännöstä tai maakoodia, niin todennäköisesti osaisin venäjää tai tietäisin ainakin maakoodin. Jos tietäisin maakoodin, niin se lisäisi todennäköisyyttä sille, että olen yhteydessä venäläisiin. Mahdollisen tiedustelun suoritti tässä tapauksessa todennäköisesti CIA, NSA, SRV tai SIS/MI6. Lähettäjän tunnistamisessa numeron maakoodi tai viestin kieli ei merkitse mitään.

Lähipäivinä huomasin tietyssä paikassa pahvinpalan, johon oli kirjoitettu etunimeni. Kukaan muun niminen ei pysähtyisi katsomaan sitä. Kun tuolla paikalla on vieläpä hyvä näkyvyys useaan suuntaan, niin 600- tai 1200-millisellä objektiivilla ja jalustalla järjestelmäkameran crop factorista riippumatta saa kaukaa melkoisen hyviä kuvia. Tuollainen ”tykki” herättää kuitenkin huomiota, joten he ovat mieluiten piilossa vaikka pakettiautossa tai käyttävät pitkällä teleobjektiivilla varustettua pokkarikameraa. Minua ei olisi löytänyt

tietystä syystä matkapuhelimen sijainnin perusteella. Jos kyseessä olisi ollut Supo, niin he olisivat kyllä tunnistaneet minut ajokortin ja iän perusteella. Näiden kahden asian tapahtuminen melkein samoina päivinä on varsin epäilyttävää. Kuten kerroin johdannossa, tutkielmani on tehty vainoharhaisella ajatusmallilla.

### 2.2.2. Ohjelmistopohjaisten turvatuotteiden perusongelma

Nykyisiä käyttöjärjestelmiä ja ohjelmistoja voi pitää lähes poikkeuksetta enemmän tai vähemmän teoreettisesti turvattomina. Vaikka esimerkiksi virustorjuntaohjelmistoja ja web-selaimia on paikkailtu vuosia, niin silti niistä löytyy edelleen erilaisia tietoturva-aukkoja. Ohjelmiin kun lisätään jatkuvasti uusia ominaisuuksia eli koodia eli virheitä. Krakkerit pitävät omana salaisuutenaan osan suurelle yleisölle tuntemattomista tietoturva-aukoista, ja näin tekevät todennäköisesti myös suurimmat tiedustelupalvelut. Jos esimerkiksi web-selainta tai sähköpostiohjelmaa ajetaan rajoitetulla käyttäjätunnuksella, niin jostain käyttöjärjestelmän lukuisista komponenteista löytyy todennäköisesti tuntematon tietoturva-aukko, jolla on mahdollista suorittaa *käyttöoikeuksien ohittaminen* (privilege escalation), jolloin haittakoodilla on käytössään jopa täydet root-oikeudet.

Tämä rikkoo useimmissa mikroissa teoriassa kaikkien käyttöjärjestelmän päällä ajettavien kryptologiaan perustuvien sovellusten turvallisuuden. Tällaisia sovelluksia ovat muun muassa salasanatietokanta ja ohjelmistopohjainen kiintolevysalaus. Prosessoriin kehitetty virtualisointi saattaisi toimia, jolloin erityistä turvallisuutta vaativat sovellukset ajettaisiin toisessa käyttöjärjestelmässä. Tämä edellyttäisi kuitenkin sitä, että käyttöjärjestelmät olisi täysin eristetty toisistaan. Esimerkiksi yhteinen kiintolevy rikkoo periaatteellisen turvallisuuden. Jopa prosessorin mikrokoodista saattaisi löytyä tietoturva-aukkoja. Sandboxin käyttäminen on yksi vaihtoehto, mutta siinäkin voi olla tuntemattomia tietoturva-aukkoja. Täyden turvallisuuden takaaminen menee niin monimutkaiseksi, että kriittisimpiä sovelluksia varten kannattaa suosiolla käyttää täysin kaikilta tietoverkoilta eristettyä tietokonetta.

Ajatellaan että kehitettäisiin erillisellä palvelimella pyörivä web proxy, joka muuttaisi kaikki web-sivut kuvakartoiksi. Tekstikenttien sisältö ja muu syöte syötettäisiin proxyn luoman (X)HTML-sivun vastaaviin tekstikenttiin. Java, JavaScript, Flash-animaatiot ja muu dynaaminen sisältö karsittaisiin kokonaan pois. Organisaatiolla olisi palomuuri muun muassa Internetin ja yrityksen lähiverkon välissä. Nyt proxyssä itsessään saattaisi hyvinkin olla tietoturva-aukko, jonka kautta vaarantuu vähintään sen prosessi jos ei koko yrityksen lähiverkko. Sen kautta käyttäjän tavalliselle selaimelle voitaisiin syöttää jälleen perinteistä

haittakoodia ohi kaikkien palomuurien. Tavallinen käyttäjä tuskin epäilisi näin vainoharhaisen menettelyn turvallisuutta. Selaimiakaan ei tulisi päivitettyä. Monet nykyiset web-sivut eivät toimi ilman JavaScript- tai Flash-tukea. Niinpä tätä varten yrityksessä täytyisi olla käytännössä ylimääräisiä muusta verkosta eristettyjä koneita.

Toimistotyöntekijän täytyy yleensä käyttää myös sähköpostia, toimisto-ohjelmia sekä musiikintoisto-, pakkaus- ja muita ohjelmia. Niinpä kaikki sovellukset täytyisi eristää web-liikenteen tapaan, mikä ei ole käytännössä mahdollista.

# 3. Tietoturvan osa-alueista organisaatiossa

Käsittelen tässä luvussa useita tietoturvan osa-alueita, jotka täytyisi huomioida organisaatiossa ja ohjelmistoprojekteissa. Jos kiinnitetään huomiota vain kryptologiaan ja käyttäjätunnistukseen, niin aita saattaa jäädä turhan matalaksi muilla tietoturvan osa-alueilla.

## 3.1. Pelisäännöt

Organisaatioihin luodaan yleensä jonkinlainen tietoturvapolitiikka tai säännöstö. Tietoturvapolitiikka on joukko lakeja, sääntöjä ja käytäntöjä, joka säätelevät sen, kuinka organisaatio johtaa, turvaa ja jakaa arkaluonteista tietoa [Walker, 1985]. Muutaman hengen yrityksessä ne voidaan sopia suullisesti. Suurempaan organisaatioon on jo syytä laatia kirjalliset pelisäännöt. Usein on tarve luoda oma säännöstö jollekin tietylle tietojärjestelmälle. Valmista säännöstöä ei ole, koska ei ole olemassa tietoturvapolitiikkaa, joka sopii kaikkiin organisaatioihin tai ympäristöihin [Whitman *et al.*, 2001].

Tietoturvasäännöstö määrää tietyllä tarkkuudella henkilöstöryhmien pelisäännöt tietojärjestelmien käytölle. Se voidaan luoda vanhan pohjalta tai laatia alusta asti. Liian tarkat ohjeet ovat rasite käyttäjille ja liian ylimalkaiset eivät kerro riittävästi järjestelmien käytöstä. Kannattaa myös pitää mielessä, että sääntöjä täytyy todennäköisesti muuttaa ajan kuluessa. Ohjeistuksen käyttöarvo putoaa lopulta nollaan, jos sitä ei koskaan päivitetä ympäristön muuttuessa [Whitman *et al.*, 2001].

Aluksi organisaatiolle tai ympäristölle suoritetaan useita turvallisuuskatselmointeja, joilla kartoitetaan ympäristön nykyinen tilanne ja säännöt. Varsinainen säännöstödokumentti muodostetaan yleensä kolmella vaihtoehtoisella tavalla. Niiden soveltuvuus on kiinni organisaation ja sen tietojärjestelmien koosta ja hierarkiasta. Ensimmäinen vaihtoehto on luoda jokaiselle järjestelmälle ja tekniikalle oma dokumentti. Suuressa organisaatiossa tällaisen dokumenttijoukon ylläpito osoittautuu helposti erittäin suureksi taakaksi. Toinen vaihtoehto on tehdä vain yksi dokumentti, joka kattaa kaikki säännöt. Tällöin dokumentista tulee helposti liian ylimalkainen, jolloin siinä ei ole riittävästi yksityiskohtia. Kolmas vaihtoehto on luoda modulaarinen rakenne, jossa perusdokumentissa kerrotaan yleiset säännöt ja osajärjestelmille ja tekniikoille tehdään erilliset modulaariset liitedokumentit, joissa tarkennetaan perusdokumentin asioita, jos sille on tarvetta. Näin perusdokumentti pysyy melko samanlaisena ja moduuleita voidaan lisätä poistaa sekä päivittää itsenäisesti. Moduulit jäävät näin melko kompakteiksi. HTML-muotoinen ohjeistus on yksi

vaihtoehto, koska tällöin dokumenttien linkitys toimii luontevasti. Ohjeistus voidaan näin sijoittaa esimerkiksi intranetiin.

Ohjeistus sisältää useita lukuja, joista kukin kertoo jonkin asian kohdejärjestelmästä. Luvut sisältävät muun muassa järjestelmän yleiskuvauksen, ohjeiden kohteen sekä soveltuvuuden, käyttäjien roolit ja oikeudet, sallitut sekä kielletyt toimenpiteet, ylläpitoasioita, käyttäjien seurannan sekä väärinkäytön seuraamukset [Whitman *et al.*, 2001]. Dokumentissa voi olla käyttöohjeita esimerkiksi kirjautumiseen ja salasanojen tai salausavainten säilyttämiseen.

### 3.2. Laitteistoturvallisuus ja palomuurit

Varmuuskopiointi ja seuranta ovat tärkeitä asioita järjestelmien jatkuvan toiminnan takaamiseksi. Laitteistoturvallisuus alkaa tilojen lukituksesta ja UPS (Uninterruptible Power Supply)- tai muun varavirran järjestämisestä sähkökatkosten varalle. Palomuurin säännöt täytyy myös säätää kuntoon. Hyvänä perussääntönä on, että vain tietty liikenne sallitaan. Esimerkiksi kaikki Internetistä tulevat yhteydenotot kannattaa estää kaikilla protokollilla ja porteilla eli yleensä TCP:llä, UDP:llä ja ICMP:llä. Suositeltava ratkaisu on sellainen, missä jokaisella mikrolla on vielä lisäksi automaattisesti päivittyvä virustorjuntaohjelmisto ja ohjelmistopalomuuri [Northcutt *et al.*, 2001]. Ohjelmistopalomuuri estää mahdollisesti virusta tai vakoiluohjelmaa lähettämästä organisaation luottamuksellista tietoa ulkopuolisille. Lisäksi ylimääräiset palomuurit hankaloittavat krakkerin tehtävää.

Esimerkiksi web-palvelimet kannattaa eristää kokonaan muusta verkosta. Vaihtoehtoisesti ne voidaan sijoittaa Internetin ja lähiverkon väliin siten, että palvelimen molemmilla puolilla on palomuuri. Tällainen järjestely on nimeltään demilitarized zone (DMZ) [IBM, 2003]. Liikenne yleislähetysosoitteisiin kannattaa sulkea kokonaan [Northcutt *et al.*, 2001]. Web-palvelimien mukana tulee usein erilaisia testiskriptejä sijoitettuna esimerkiksi CGI-hakemistoon, jotka kannattaa yleensä poistaa tai siirtää pois käytöstä. Niistä on nimittäin löytynyt vuosien mittaan tietoturva-aukkoja [Ghosh, 1998]. Kaikkien HTML- ja muiden tiedostojen sekä hakemistojen käyttäjäoikeudet täytyy myös säätää kuntoon.

Jokaisen ylläpitäjän kannattaisi seurata vähintään Cert.fi-sivuston varoituksia ja haavoittuvuuksia sekä SANS Instituten tietoturva-aukkojen top 20 -listaa [SANS Institute, 2007]. Suurin osa hyökkäyksistä kohdistuu yleisiin tietoturva-aukkoihin, joten niiden korjaaminen on tärkeää. Muun muassa BugTraq-tietokannasta [SecurityFocus, 2008] voi etsiä, löytyisikö tietylle sovellukselle tietoturva-aukkoja.

### 3.3. Ohjelmointivirheet

Tietoturvan testauksessa voi käyttää matematiikassa esiintyvää vastaesimerkkimenetelmää. Matematiikassa vastaesimerkit romuttavat tehokkaasti vääriä väitteitä. Olen katselmoinut jonkin verran palvelinohjelmistojen lähdekoodia tavalla, jossa tutkin syöteparametrien ja puskureiden käyttöä. Ohjelmoijat tekevät toisinaan koodiin vaarallisia *puskurin ylivuotovirheitä* (buffer overflow) laiskuuden ja kiireellisen aikataulun vuoksi. Ylivuotovirheet ovat yleisin tietoturva-aukkojen syy. Eräs tapa testata ohjelmaa on, että yritetään keksiä ajatustasolla tilanteita, joissa toteutuksen tietoturva ei pidä. Nämä ajatukset voidaan kirjoittaa testitapauksiksi tai suorittaa ad hoc -testausta. Olen havainnut ad hoc -testit varsin tehokkaiksi, kun olen yrittänyt rikkoa ohjelmoijien toteutuksia black box -testauksessa. Tämän toimivuus on hyvin paljon kiinni testaajan asiantuntemuksesta ohjelmoijana, ajattelutavasta ja järjestelmällisyydestä. Siten perinteinen dokumentoitu testaus on keskimäärin turvallisempi valinta. Myös kryptologisen toteutuksen turvattomuus voidaan todistaa etsimällä tietty tilanne, joka toistuu tietyllä todennäköisyydellä ja rikkoo turvallisuuden. Koskaan ei saisi ajatella, että ongelmaa ei tarvitse korjata, koska krakkeri tai kryptoanalyytikko ei muka löydä kyseistä virhettä.

C ja C++ -kielillä toteutetut CGI-ohjelmat ovat erityisen alttiita ohjelmointivirheille, jos niissä ei tarkisteta kaikkia syötteitä ja niiden käsittelyä kunnolla. Javassa tällainen ylivuotovirhe aiheuttaa poikkeuksen, joka voidaan käsitellä. Jos poikkeusta ei käsitellä, niin ohjelman suoritus keskeytyy automaattisesti, jos kyseessä on perinteinen palvelinsovellus. Javassa perinteiset puskurin ylivuotovirheet eivät salli krakkerin suorittaa mielivaltaista konekielikoodia, jos virtuaalikoneen turvallisuus olisi täydellinen. Virtuaalikoneista on kuitenkin löytynyt toistuvasti uusia tietoturva-aukkoja.

Käyttöjärjestelmät rajoittavat virheiden hyväksikäyttöä, minkä vuoksi palvelinohjelmia kannattaa ajaa sellaisella käyttäjätunnuksella, jolla on hyvin rajoitetut oikeudet ajoympäristön resursseihin. Käyttäjien syöte täytyy tarkistaa huolella kaikissa ohjelmointikielissä. HTML-lomakkeiden parametrit sekä HTTP:n cookiet kannattaa tarkistaa kaikkien mahdollisten väärinkäytösten kannalta. Parametrien pituuden ja sallittujen merkkien tarkastaminen ei ole yleensä kiinni ohjelman tehokkuudesta tai toteutuksen vaikeudesta vaan ohjelmoijan laiskuudesta tai kiireellisestä aikataulusta. Erityisesti kannattaa kiinnittää huomiota tilanteeseen, jossa käyttäjän syötettä käytetään osana SQL-muotoista tietokantahakua seuraavan esimerkin tapaan:

```
sqlQuery = ”SELECT something FROM tablename ”+
  ”WHERE somevalue=’”+inputValue+”’;”
```

Jos esimerkin inputValue-muuttuja saa web-lomakkeen kautta esimerkiksi arvon ” ’; DROP table tablename; --”, niin kyseisen tietokantataulun tuhoava hyökkäys mahdollisesti onnistuu. Aiheesta löytyy varsin mielenkiintoinen kuvaus esimerkiksi sellaisia lomakkeita vastaan, joilla lähetetään järjestelmän unohtunut salasana käyttäjän sähköpostiosoitteeseen [Friedl, 2007]. Java-ohjelmissa tätä ongelmaa rajoittaa se, että käyttää PreparedStatement-luokan olioita, jolloin kaikki syöteparametrit ovat vahvasti tyypitettyjä [Crawford, 2002]. Omat tarkistusmetodit vaativat enemmän työtä ja voivat sisältää ohjelmointivirheitä. Jos niitä aikoo käyttää, niin on parempi hyväksyä vain tietyt merkit kuten heksadesimaaliarvot. On huomattavasti vaikeampaa luetella kaikki kielletyt erikoismerkit. Samanlaisia ongelmia esiintyy myös protokollissa ja web-palvelimien CGI-ohjelmissa. Esimerkiksi SMTP (Simple Mail Transfer Protocol)-protokollassa täytyy sähköpostiviestin viestiosassa korvata yhden pisteen sisältävät rivit kahden pisteen rivillä, koska pelkkä piste ennen rivinvaihtoa merkitsee viestiosan loppua. Viestiosan jälkeen krakkeri voisi syöttää muita komentoja ja lähettää ylimääräisiä viestejä.

Resurssien varauksessa täytyy ottaa huomioon mahdolliset palvelunesto-hyökkäykset, vaikka niitä ei voida nykyisissä tietoverkoissa täysin estää [Northcutt *et al.*, 2001]. Hyökkäyksen toteuttaminen voidaan kuitenkin tehdä vaikeammaksi. Esimerkiksi Linuxin TCP/IP-pinosta pudotetaan satunnaisesti yksi pending-tilassa oleva yhteys pois, mikä haittaa tiettyjä palvelunestohyökkäyksiä [Ghosh, 1998]. Tietokanta- ja muiden yhteyksien lukumäärä kannattaa rajoittaa. Esimerkiksi Java Servlet -toteutuksissa on yleensä muutenkin järkevää toteuttaa tietokantayhteyksiä varten database pool [Crawford, 2002], josta sovelluksen säikeet varaavat hetkellisesti tietokantayhteyksiä ja palauttavat ne takaisin käytön jälkeen. Näin tehdään yleensä tehokkuusvaatimusten vuoksi, koska tietokantayhteyden avaaminen on raskas toimenpide.

Jos poolissa ei ole riittävästi yhteyksiä, niin yhteyttä kysyvä säie joutuu odottamaan yhteyden vapautumista tai kysely epäonnistuu. Pooli voi kasvattaa ja laskea yhteyksien lukumäärää dynaamisesti, mutta niiden lukumäärälle täytyy määritellä maksimiarvo. Asiakasyhteyksien lukumäärä kannattaa myös rajoittaa. Jos resurssien käytölle asetetaan rajat, niin krakkerin palvelinesto-hyökkäys voi onnistua kyseistä sovellusta vastaan, mutta käyttöjärjestelmän kuorma pysyy kohtuullisena. Myös säikeille ja muille resursseille on mahdollista tehdä pooleja.

Java-ohjelmille voidaan suorittaa *takaisinmallinnus* (reverse engineering), koska tavukoodista saadaan helposti lähdekoodia ja samalla automaattisesti luokkakaaviot. Tätä voidaan hankaloittaa obfuscation-ohjelmistolla

[Low, 1998], joka muuttaa luokkien, metodien ja muuttujien nimet pseudosatunnaisiksi, jolloin koodin ymmärtäminen on huomattavasti vaikeampaa. Ulkoisia rajapintoja ei voi kuitenkaan ajaa obfuscation-sovelluksen läpi. Toimenpiteen jälkeen ohjelman rakenne on kuitenkin sama, joten kärsivällinen kilpailija voi korvata ensin kaikki satunnaiset nimet automaattisesti esimerkiksi nimillä method1, method2, var1, var2 ja niin edelleen. Tämän jälkeen hän voi nimetä muuttujat uudelleen generoidun luokkakaavion ja oman päättelyn perusteella. Obfuscation-ohjelmat muuttavat tarvittaessa myös ohjelmakoodin rakennetta, mikä tuo lisäturvaa. Merkkijonot voidaan myös tarvittaessa salata ja purkaa ajoaikana, mutta tämä vie mahdollisesti hyötyyn nähden suhteettomasti suorituskykyä.

### 3.4. Haittaohjelmat

Eräs merkittävä tietoturvariski ovat organisaation omat työntekijät, koska he tekevät inhimillisiä virheitä. Puhelin on edelleen yksi krakkerin työkaluista. Krakkeri voi udella käyttäjältä esimerkiksi organisaation puhelinnumeroita tai tietoa lähiverkon rakenteesta. Näin hän voi kohdistaa hyökkäyksensä täsmällisemmin. Hän voi esiintyä esimerkiksi toisena työntekijänä, asiakkaan edustajana tai poliisina.

Ihmiset lankeavat manipulointiin myös sähköpostin välityksellä. Tietoturva-aukkoja sisältävät yleiset sovellukset tarjoavat mahdollisesti helpon pääsyn organisaation lähiverkkoon. Sähköpostipalvelimen virustarkistimeen ei voi täysin luottaa, koska virusten uudet leviämismekanismit [Weaver, 2001] [Staniford *et al.*, 2001] pystyvät levittämään viruksen Internetin laajuudelta jopa alle minuutissa, mihin kukaan ei ehdi reagoida käytännössä mitenkään. Käyttäjiä on muutenkin syytä opastaa, sillä edistynyt krakkeri voi ohjelmoida troijalaisen itse, jolloin liitetiedosto pääsee todennäköisesti virustarkistuksen läpi.

Organisaatiot sallivat usein melko vapaasti tai täysin vapaasti tietoliikennettä sisäverkosta Internetiin. Eräs skenaario voisi olla sellainen, että krakkerin troijalainen luo järjestelmän saastumisen jälkeen pääteyhteyden TCP-yhteyden yli toiseen hakkeroituun järjestelmään. Nyt hän voi käyttää konetta käyttäjätunnuksen oikeuksien puitteissa. Käyttörajoituksetkin mahdollisesti vain hidastavat krakkeria, koska käyttöjärjestelmissä on ohjelmointivirheitä. Hän voi nyt käyttää konetta murtautumiseen lähiverkon muihin koneisiin ja hyvin usein myös muihin organisaatioihin suuntautuviin hyökkäyksiin mahdollisesti muiden avustuksella [Northcutt *et al.*, 2001].

On olemassa muitakin vastaavia tapoja saastuttaa koneita. Krakkeri saattaa asioida usein toimistossa, jossa on näkyvillä cd-levyjä. Hän voi korvata esi-

merkiksi jonkin tietyn ohjelman asennuslevyn saastuneella versiolla, tai hän voi lähettää vaikka kaupallisen ohjelman kokeiluversion perinteisessä postissa. Prässätty levy ja väärennetyt etiketit voivat saavuttaa kohdehenkilön luottamuksen. Hakkeroinnissa on mahdollista käyttää mielikuvitusta ja yhdistellä erilaisia hyökkäystapoja. Tällaiset hybridihyökkäykset ovat mahdollisesti hyvinkin tehokkaita. Klassinen tapa on tutkia organisaation paperijätettä, joten luottamukselliset asiakirjat on syytä syöttää paperintuhoojaan. Teoriassa paperisilppukin on mahdollista koota tietokoneella takaisin arkeiksi.

### 3.5. Tietoturva-arkkitehtuureista

Tietoturvaa voidaan parantaa ennaltaehkäisemällä, tunnistamalla, eristämällä, arvioimalla ja reagoimalla hyökkäyksiin [Ye *et al.*, 2001]. Ohjelmoijien kouluttaminen sekä huolellinen suunnittelu ja ohjelmointitapa sekä testaaminen ja koodin vertaisarviointi ehkäisevät tietoturva-aukkojen muodostumista. Onnistuneen hyökkäyksen vaikutukset voidaan jakaa luokkiin niiden vakavuuden kannalta. Hyökkäykseen reagoidaan perinteisesti katkaisemalla krakkerin tietoliikenneyhteydet, korjaamalla vauriot ja palauttamalla järjestelmä normaaliin tilaan. Tämä ei ole kyllä nykyään aina mahdollista suurissa järjestelmissä, kuten digital forensic investigation -asiantuntijat tietävät varsin hyvin [Adelstein, 2006]. Testauksella etsitään järjestelmästä tietoturvan kannalta heikkoja kohtia. Sellaisia ovat muun muassa heikot salasanat, ohjelmistojen vanhat versiot ja bugit omissa ohjelmistoissa.

Sallittujen operaatioiden ja tilojen määrittely järjestelmän suunnitteluvaiheessa sekä järjestelmän seuranta parantavat tietoturvaa. Ye ja muut [2001] käsittelevät järjestelmien tietoturvan suunnittelua prosessinhallinnan näkökulmasta, joka on yksi tapa suunnitella tietojärjestelmiä. Suunnittelumenetelmä muistuttaa jonkin verran yleisiä suunnittelumenetelmiä. Prosessikeskeisessä suunnittelumenetelmässä on objektiivinen, käsitteellinen, toiminnallinen ja fyysinen tarkastelukulma. Järjestelmän suunnittelu aloitetaan ylimmältä tasolta, joka on abstraktein. Käsitteellisessä vaiheessa ei oteta vielä kantaa järjestelmän käytännön toteutukseen kuten ohjelmointikieleen. Tosin joskus täytyy tehdä pieniä poikkeuksia. Muutos yhdessä tarkastelutasossa muuttaa mahdollisesti muitakin tasoja. Tasot helpottavat järjestelmän hahmottamista ja yksityiskohtien hallintaa. Järjestelmät voidaan usein jakaa alijärjestelmiin, jolloin suurten järjestelmien monimutkaisuutta voidaan pilkkoa pienempiin osiin.

Voidaan myös noudattaa tiettyjä arkkitehtuurisia rakenteita kuten *kehyksiä* (frameworks). Kehysrakenteet ja abstrakti suunnittelu tuottavat helposti lisäkustannuksia, mutta suurissa projekteissa ja tuoteperheissä se on usein jopa välttämätöntä. Jos tuotteen elinikä on pitkä, niin silloin tuotteen

rakenteelle asetetaan erityisiä vaatimuksia, koska sen toiminnallisuus ja rooli muuttuvat todennäköisesti ajan mittaan. Lyhytikäisissäkin tuotteissa kehysrakenteista ja ohjelmakirjastoista on ehkä hyötyä, jos tuotteet muistuttavat toisiaan. Suurissa sovelluksissa hyvin valittu kehysrakenne yksinkertaistaa sovellusta ja tuotteen hahmottaminen on helpompaa. Laadukkaat ja ajantasaiset dokumentit sekä sovelluksen selkeä rakenne helpottavat sovelluksen jatkokehitystä ja ylläpitoa. Esimerkiksi ylipitkiä luokkatiedostoja ja suoria viittauksia toisten luokkien muuttujiin täytyy välttää. *Suunnittelumalleja* (design patterns) [Gamma *et al.*, 1994] kannattaa käyttää, jos se on mahdollista ja järkevää.

Liiallista hajautusta kannattaa välttää, jos tekijätiimi ei ole aiemmin toteuttanut hajautettuja sovelluksia. Ylimääräinen hajautus lisää mahdollisesti myös tietoturva-aukkojen lukumäärää. Synkronointi on yksi suurimpia suurten järjestelmien ohjelmointivirheiden lähteitä. Jos erilaisia etäobjekteja on satoja ja tapahtumiin osallistuu kymmeniä tai jopa kaikki etäobjektit, niin toteuttajilla voi olla erittäin suuria vaikeuksia saada ohjelmisto eli tässä tapauksessa kaaos toimimaan oikein. Fregonese ja muut [1999] tarkoittavat ohjelmiston arkkitehtuurin suunnittelulla sitä, että määritellään *tukeva* (robust) kehysrakenne, jonka varaan voidaan rakentaa sovelluksia ja alijärjestelmiä. He myös mainitsevat uudelleenkäytön, jolloin sovelluksia ei tarvitsisi aina suunnitella kokonaan uudelleen. Toteutusten uudelleenkäyttö toteutuu turhan harvoin, mutta onnistuneessa tapauksessa siitä on hyötyä. Sovelluksessa on yleensä sitä suuremmalla todennäköisyydellä tietoturva-aukkoja, mitä enemmän siinä on muitakin ohjelmointivirheitä. Ongelmat arkkitehtuurissa tai suunnittelussa voivat aiheuttaa lisää ohjelmointivirheitä ja edelleen tietoturvaongelmia. Tällainen dominoteoria pätee ainakin osassa ohjelmistoprojekteja.

Hajautetun sovelluksen tietoturvan suunnittelua vaikeuttavat muun muassa käyttäjien suuri määrä, resurssien dynaaminen käyttö, säikeiden ja prosessien suuri määrä sekä useat käyttäjätunnistuksen menetelmät [Foster *et al.*, 1998]. Käyttäjätunnistuksessa käyttäjäystävällisin vaihtoehto on sellainen, että käyttäjää pyydetään todistamaan henkilöllisyytensä vain kerran. Tällaisia single sign-on -järjestelmiä on nykyään käytössä erityisesti webissä. Käyttäjän kannalta on raskasta, jos hän joutuu kirjautumaan useaan saman organisaation järjestelmään samoilla tunnuksilla. Verkossa voi olla myös palvelin, joka toimii osana hajautettua järjestelmää ja kirjautuu käyttäjän puolesta muihin järjestelmiin. Näin tapahtumat voivat kestää jopa useita viikkoja. Käyttäjän ei kuitenkaan tarvitse olla kirjautuneena järjestelmään näin kauan. Palvelun täytyisi säilyttää käyttäjätunnuksia vain niin kauan, kuin se on välttämätöntä.

Eräs vaihtoehto käyttäjätunnistukseen on sellainen, jossa jokaisesta käyttäjästä on palvelimella julkinen avain. Käyttäjä allekirjoittaa palvelun käyttötilanteessa salaisella avaimellaan määräaikaisen luvan, jonka hän lähettää palvelimelle. Lupa sisältää kellonajan, listan resurssien käyttöoikeuksista ja voimassaoloajan kullekin resurssille. Tällainen ratkaisu toimii parhaiten, jos salasanoja ei tarvita ulkoisiin järjestelmiin. Salasanoja voidaan toki välittää palvelimelle salattuna.

Kun selväkielisiä salasanoja käsitellään, niin myös salasanoja sisältävät muistialueet täytyisi ylikirjoittaa ennen muistialueen vapauttamista [McGraw and Viega, 2000], koska käyttöjärjestelmän muistivapautusfunktiot eivät sitä tee tehokkuussyystä. Ilman ylikirjoitusta vapautettu muistialue merkitään ainoastaan vapaaksi, joten krakkeri voi etsiä palvelimen *muistikaappauksesta* (memory dump) salasanoja. Javassa String-luokan instanssin merkkijonoa ei voi kuitenkaan muuttaa sen muodostamisen jälkeen, joten salasanojen käsittelyssä String-oliot on mahdollista korvata esimerkiksi merkkijonotaulukolla. Taulukon ympärille voi toki rakentaa oman luokan, jossa on muun muassa yksinkertainen wipe()-metodi merkkijonon ylikirjoitusta varten.

Salasanoja täytyisi säilyttää muistissa aina mahdollisimman lyhyen ajan [McGraw and Viega, 2000]. Jos salasanatiedosto tuhotaan tai kopioidaan, niin poistettava tiedosto täytyy aina ylikirjoittaa vähintään kerran. Levyjärjestelmissä lohko merkitään keskusmuistin tavoin vain vapaaksi eikä sitä automaattisesti ylikirjoiteta. Eräs ongelma on vielä tietokoneen virtuaalimuisti. Virtuaalimuistia käytetään erityisesti silloin, kun koneen kaikki ohjelmat eivät mahdu kerralla muistiin. Niinpä tällaisessa tilanteessa myös salasanat kopioidaan mahdollisesti kiintolevylle, jolloin krakkeri voi löytää myöhemmin virtuaalimuistista salasanoja. Kopiointia voidaan vähentää, jos palvelimeen asennetaan sen verran muistia, että sitä jää myös vapaaksi normaalikäytössä. Jos käyttöjärjestelmässä ja toteutuskielessä on mahdollista lukita muistia salasanoja varten, niin näin kannattaa tehdä. Lukitus estää tämän erityisen muistialueen kopiointia virtuaalimuistiin

### 3.6. Tietomurtojen tunnistaminen

*Tietoverkkomurtojen tunnistamismenetelmiä* (network intrusion detection) voidaan verrata kiinteistöjen hälytysjärjestelmiin [Stillerman *et al.*, 1999]. Hälytysjärjestelmän tavoin tietokoneohjelma reagoi tiettyyn tapahtumaan tai tapahtumasarjaan määritellyllä tavalla. Hakkerointiyritykset täytyisi mieluiten tunnistaa jo niiden alkaessa, jolloin niiden vaikutusta voidaan yrittää minimoida. Hyökkäyksiä tunnistetaan lähinnä neljällä tasolla, jotka ovat verkko-,

protokolla-, käyttöjärjestelmä- ja sovellustaso. Kerroksia voidaan tunnistaa tarvittaessa lisää.

Verkkotaso toteutetaan yleensä palomuurilla. TCP-yhteydessä krakkerin tai muun väärin käyttäytyvän henkilön yhteyttä ei sallita. UDP-pakettien tapauksessa yhteyttä ei ole, joten paketit jätetään vain käsittelemättä. Myös ICMP- ja muissa paketeissa käytetään sopivia rajoituksia. Protokollatason valvonta toteutetaan palomuurissa, sovelluksessa tai kirjastoissa, joka toteuttavat kyseisen protokollan. Sovellustason valvonta seuraa käyttäjän toimia sovelluslogiikan kannalta. Käyttöjärjestelmätaso toteutetaan käyttöjärjestelmässä tai virtuaalikoneessa. Sovelluksessa voi olla alimman tason kerros, joka rajoittaa ohjelman toimintaa konekielitasolla. Java-ohjelmissa virtuaalikone rajoittaa esimerkiksi appleteissa sallittujen toimintojen joukkoa.

Krakkerille voidaan uskotella, että häntä ei ole havaittu. Hänelle voidaan palauttaa esimerkiksi virheellistä tietoa. Se on kuitenkin harvoin mahdollista, ja valetoteutuksessa voi myös tietoturvaongelmia. Pilailu voi kostautua järjestelmän ylläpitäjälle. Kun krakkeri huomaa, että häntä on jymäytetty, niin järjestelmän hakkeroinnista voi tulla hänelle entistä suurempi haaste. Palomuuri voisi muokata asetuksiaan automaattisesti ja estää paketit tietyistä IP-osoitteista, mutta krakkeri voi väärentää helposti pakettien lähdeosoitteen, jolloin hän voisi käyttää lähdeosoitteiden kieltolistaa hyväksi palvelunestohyökkäyksessä. Hajautettu palvelunestohyökkäys yhdistettynä väärennettyihin satunnaisiin IP-osoitteisiin täyttäisi kieltolistan nopeasti. Ongelmaa voidaan lieventää siten, että rajoitetaan listan pituutta ja kielletään yhteydet vain tietyksi ajaksi. Hajautetun hyökkäyksen voi tunnistaa myös siitä, että useasta eri IP-osoitteesta tulee pienessä ajassa suuri joukko epätavallisia yhteydenottoja.

Hyökkäykset tunnistetaan yleensä joko malliin tai poikkeavuuteen perustuvalla menetelmällä [Stillerman *et al.*, 1999]. Tunnistusmenetelmä on tehokas, jos se tunnistaa mahdollisimman suuren osan hyökkäyksistä, mutta ei tuota turhia varoituksia. Snort on hyvä esimerkki tietomurtojen havaitsemistyökalusta, koska se on ilmainen, toimii suuressa osassa käyttöjärjestelmiä, sillä on aktiivinen käyttäjäkunta, ja se on teknisesti yksi parhaita [Northcutt *et al.*, 2001]. Tällä hetkellä malleja löytyy jo melkoisesti ja niitä kirjoitetaan jatkuvasti lisää. Malleihin perustuva menetelmä ei tunnista uusia tuntemattomia hyökkäyksiä, mutta erilaisilla heuristiikoilla niitäkin voidaan havaita. Ylläpitäjän kannalta oikeiden hälytysten ohella tärkeää on väärien hälytysten vähyys. Väärä hälytys muodostuu, jos käyttäjä tekee sallittuja toimintoja, mutta ne tulkitaan hyökkäykseksi. Malliperustaiset menetelmät tuottavat yleensä hyvin vähän vääriä hälytyksiä.

Poikkeavuuteen perustuvat menetelmät yrittävät muodostaa kuvan järjestelmän tai käyttäjän normaalista toiminnasta ja havaita tavallisesta poikkeavat tilanteet [Stillerman *et al.*, 1999]. Ensin määritellään, mikä on järjestelmän kannalta normaalia toimintaa. Menetelmää voidaan verrata elävän solun tai olion immuunijärjestelmän toimintaan, joka puolustautuu ulkopuolisia hyökkääjiä vastaan [Forrest *et al.*, 1997]. Ongelmia tulee silloin, jos järjestelmän kokoonpano vaihtuu. Ye ja muut [2001] ovat käyttäneet apunaan tiedonlouhintaa, kun he ovat muuttaneet järjestelmän kokoonpanoa.

Luonnon puolustusjärjestelmät ovat hyvin kehittyneitä. Ne sisältävät monikerroksisen ja hajautetun suojauksen, joka oppii miten hyökkääjä tunnistetaan ja miten sitä vastaan voidaan suojautua. Osa puolustusjärjestelmästä voi hajautuksen vuoksi tuhoutua, mutta silti puolustus pitää. Tosin ihmisessä ja samoin tietojärjestelmissä on osia, jotka ovat välttämättömiä toiminnan kannalta. Monikerroksisuus tulee siitä, että esimerkiksi iho, limakalvot ja solut suojaavat erilaisia ja osittain samoja hyökkäyksiä vastaan. Immuunijärjestelmissä on kuitenkin tunnetusti aukkoja.

Tietojärjestelmiä suojaavissa ohjelmissa on joitain biologisille immuunijärjestelmille ominaisia piirteitä, mutta ne ovat silti niihin verrattuna melko kehittymättömiä. Tuntematon hyökkääjä täytyy ensin erottaa ja tunnistaa ”itsestä”. Tietojärjestelmissä tapahtuu usein runsaasti sallittuja muutoksia ohjelman suorituksen aikana, mikä vaikeuttaa suojaavien järjestelmien rakentamista [Forrest *et al.*, 1997]. Hyökkäyksestä voidaan muuttaa hieman esimerkiksi sen toimenpiteiden tai komentojen järjestystä, jolloin tunnistus ei välttämättä toimi. Käyttäjän malliin voi kuulua esimerkiksi käskysarjoja ja muuta hänestä kerättyä tietoa. Esimerkiksi web-selaimesta voidaan kerätä tietoa sen versiosta, käyttöjärjestelmästä, *kekseistä* (cookie) ja näyttötilasta. Käyttäjä voi selata sivuja tietyllä nopeudella ja usein vain tiettyjä sivuja.

Liian herkkä järjestelmä kuormittaa ylläpitoa ja liian epäherkkä ei havaitse kunnolla hyökkäyksiä. Tässä kannattaa huomata analogia biometrisiin tunnistusmenetelmiin. On vain harvoin mahdollista toteuttaa erehtymätön heuristinen tunnistusmekanismi [Forrest *et al.*, 1997]. Herkkyys täytyy tietenkin päättää hajautetun järjestelmän käyttökohteen mukaan. Forrest ja muut [1997] pitävät yhtenä suurimmista eroista immuunijärjestelmien ja tietojärjestelmien välillä sitä, että immuunijärjestelmän ei tarvitse suojella salaisuuksia tai yksityisyyttä.

Valmiilla työkaluilla toteutetut hyökkäykset ovat yleisimpiä, koska niiden toteuttaminen onnistuu jopa henkilöltä, joka ei tiedä paljonkaan ohjelmoinnista tai hakkeroinnin teknisestä puolesta. Hakkerointi vaatii yleensä teknisen tietämyksen lisäksi myös tarkkuutta ja runsaasti kärsivällisyyttä. Internetissä on

kuitenkin paljon tietokoneita, joiden ohjelmistoja ei ole päivitetty. Tämän vuoksi yksinkertaisetkin hyökkäykset ovat suuressa joukossa IP-osoitteita yllättävän tehokkaita. On huomattavasti vaikeampi hakkeroida johonkin todennäköisesti hyvin suojattuun organisaatioon kuin satunnaiseen osoitteeseen. Montako sellaista Apache- tai Microsoft IIS web-palvelinta löytyy, joita ei ole päivitetty jopa vuosiin? Tietomurtojen tunnistusjärjestelmä estää mahdollisesti osan hyökkäyksistä, jotka kohdistuvat päivittämättömään palvelinsovellukseen. Se on kuitenkin epävarmaa, joten ohjelmistoja kannattaa päivittää usein. Palomuurit tai muut verkkotasoa valvovat ohjelmat eivät voi seurata salattua liikennettä. Palomuuri voisi tietysti toimia proxynä, mutta siitä aiheutuisi kryptologiaan liittyviä riskejä ja muita ongelmia.

Poikkeavuuteen perustuvissa suojausohjelmissa ongelmaksi muodostuu helposti toimintasarjojen kombinaatioiden lukumäärä. Jos sovelluksessa on runsaasti toimintoja, niin mahdollisia sallittuja toimintasarjoja on lukuisia. Järjestelmä luo seurattavasta toiminnasta tavallaan sormenjälkiä, jotka ovat jonkinlaisia tiivisteitä. Tiivisteet luodaan tunnistustavan opetusvaiheessa ja toiminnan aikana toimintasarjojen kuten järjestelmäkutsujen järjestyksen perusteella. Tiivisteet eivät ole kuitenkaan perinteisiä hash-arvoja vaan ne sisältävät enemmän rakennetta. Useimmiten on tarpeellista säilyttää toiminnasta yksityiskohtia, jolloin voidaan kehittää heuristiikkoja.

Jos tiiviste luodaan hyvin valitusta tiedosta, niin usein melko vähäinenkin määrä yksityiskohtia riittää hyvään tunnistustarkkuuteen [Forrest *et al.*, 1997]. Hyvin yksityiskohtaiset tiivisteet vievät enemmän resursseja. Pienemmällä muuttujajoukolla muodostuu opetusvaiheessa sellainen tiivistejoukko, joka kattaa suuren osan luvallisista toimintosarjoista, mikä vähentää väärien hälytysten lukumäärää [Stillerman *et al.*, 1999]. Jos tiivisteessä on runsaasti parametreja, niin kombinaatioiden määrä kasvaa nopeasti niin suureksi, että kaikkia luvallisia tilanteita ei voida kartoittaa. Tietokantahaut ovat raskaita toimenpiteitä, joten tiivisteet kannattaa säilyttää muistissa, jos se on mahdollista. Tietokantana voi toimia yksittäinen tiedostokin.

Tuhansien käyttäjien sovelluksissa tiivisteiden muodostaminen ja vertailu eivät saa viedä liikaa aikaa. Tiivisteet tallennetaan tietokantaan ja luetaan ohjelman käynnistyessä. Ohjelman suorituksen aikana muodostetaan ajonaikaisista toimintasarjoista tiivisteitä, joita verrataan tietokannan tiivisteisiin. Jos tiiviste löytyy tietokannasta, niin operaatio tulkitaan lailliseksi. Järjestelmä havaitsee ehkä myös hyökkäyksiä, joita kukaan ei ole edes vielä keksinyt. Voidaan käyttää myös useita rinnakkaisia menetelmiä ja laskea niille painokertoimet [Ye *et al.*, 2001]. Jos painokertoimien muodostama todennäköisyys ylit-

tää tietyn luvun, niin tapahtuma luokitellaan hyökkäykseksi. Roskapostisuodattimet käyttävät usein tätä periaatetta.

### 3.7. Tietoliikenteen kerääminen murtojen tunnistusjärjestelmälle

Järjestelmää voidaan seurata tietomurtojen tunnistusjärjestelmän heuristiikoilla lähinnä neljällä vaihtoehtoisella tavalla ja niiden muunnelmilla [Almgren and Lindqvist, 2001]. Järjestelmää voidaan seurata muun muassa verkon solmukohdissa kuten reitittimissä tai palomuureissa. Solmukohdissa täytyy olla tarkkana vasteaikojen ja suorituskyvyn suhteen. Monimutkainen logiikka hidastaa tiedonsiirtoa, jos kuorma on suuri ja solmukohdan laskentakapasiteetti ei riitä. Kuormantasaajan ja useamman tunnistuslaitteen avulla suurikin kuorma voidaan tarkistaa ilman suorituskyvyn romahtamista. Ilman kuormantasaajaakin kuorma voidaan tasata esimerkiksi siten, että jokaisesta tietoliikenneyhteydestä lähetetään kopio jollekin useasta tunnistusjärjestelmästä. Näin hyökkäykset ehditään tunnistaa melkein ilman viivettä.

Erityisesti osa vanhemmista lähiverkoista ei ole kytkeytyviä, jolloin *tietoliikenteen kerääminen* (packet sniffing) on helppoa. Jos krakkeri onnistuu murtautumaan lähiverkon jollekin koneelle, niin hän voi vakoilla koko lähiverkon liikennettä. Tätä verkon ominaisuutta voi käyttää myös lailliseen seurantaan. Tietomurtojen tunnistusjärjestelmä kerää tällöin kaikkien verkon koneiden tietoliikenteen. Kokonaisuuden suorituskyky pysyy täsmälleen samana, koska seuranta ei vaikuta järjestelmän toimintaan kuin hyökkäystilanteissa. Verkon rakenne on kuitenkin tietoliikenteen kannalta tehoton, jos palvelimia on runsaasti ja tietoliikennekuorma on suuri. Tietoturvan kannalta ratkaisu on kehno, jos verkossa käytetään selväkielisiä tietoliikenne- tai muita protokollia.

Kolmas vaihtoehto on sellainen, jossa kaikilla palvelinkoneilla on oma tietomurtojen tunnistusjärjestelmä. Sovellusten ylläpito voidaan keskittää, mikä vähentää ylläpitokustannuksia. Jokaisen koneen sovellus voi seurata eri asioita, jos koneilla on erilaisia palvelimia. Nämä kolme vaihtoehtoa ovat toiminnaltaan melko samanlaisia. Neljäs tapa on rakentaa toiminnallisuus suoraan palvelinsovellukseen, mikä voidaan toteuttaa rajapintojen avulla. Jos palvelinsovelluksen lähdekoodi on saatavilla, niin seurantatoiminnot voidaan yhdistää suoraan siihen.

Palvelinohjelmat tarvitsisivat yleisen ja mielellään standardin rajapinnan heuristiikkamoduuleille. Ongelmana on palvelinsovellusten monimuotoisuus, joten erilaiset kontekstit täytyisi ottaa huomioon. Osa moduuleista toimisi melkein kaikissa palvelimissa, jolloin ne toteutettaisiin esimerkiksi tiedonlouhinnalla tai poikkeavuuteen perustuvilla menetelmillä. Osa moduuleista toimisi

vain tietynlaisissa palvelimissa kuten web-palvelimissa. Jos tällainen rajapinta olisi olemassa, niin yritykset voisivat tuotteistaa moduuleita. Esimerkiksi virustorjuntamoduuli voitaisiin liittää sähköposti-, web-, SSH- ja proxy-palvelimiin. Toisaalta murtautuja voisi kohdistaa hyökkäyksensä tuota rajapintaa vastaan. Tietoliikenteen tarkistaminen täytyisi pystyä hajauttamaan useammalle tarkistuspalvelimelle, jos käyttäjiä on samanaikaisesti suuri määrä. Vikatilanteessa osa liikenteestä jätettäisiin esimerkiksi tarkistamatta tai ohjattaisiin toisille palvelimille tiettyjen sääntöjen mukaisesti.

### 3.8. Hajautetun järjestelmän lokitiedostot

Lokitiedostoja muodostuu eri sovelluksista yhteensä jopa kymmeniä, jolloin niitä täytyy mieluusti hallita keskitetysti. Ylläpitäjät kyllästyvät nopeasti lokitiedostojen tarkkailuun, jos tiedostot täytyy tutkia yksitellen ja ne sijaitsevat eri paikoissa. Lokitiedostoille voidaan tehdä ohjelma, joka lukee ne kaikki. Tiedostojen on hyvä olla samassa muodossa, jolloin niiden käsittely helpottuu. Erilaisille tiedostomuodoille voidaan tehdä *sovittimia* (adapter), joiden avulla lukuohjelmaan voidaan lisätä myöhemmin uusia tiedostotyyppejä. Näin esimerkiksi XML-muotoiset tiedostot voidaan muuntaa ohjelman ymmärtämään muotoon. Eräs vaihtoehto on käyttää syslog-palvelimia keskitettyyn lokitietojen keruuseen [Oxenhandler, 2003]. Syslog-toteutuksia löytyy lähes kaikille alustoille ja myös Javalle. Keskitetyssä lokivarastossa tiedostoja on helpompi hallita, ja lokipalvelin voidaan asentaa omalle koneelleen vaikka oman palomuurin taakse. Tarkinta seurantaa vaativien järjestelmien tietyt lokirivit voidaan jopa tulostaa paperille.

Lokirivit kannattaa luokitella. Luokittelu voidaan tehdä esimerkiksi lukualueena tai vakionimillä kuten ”AUDIT”, ”DEBUG”, ”INFO”, ”WARNING” ja ”ERROR”. Ohjelman asetusparametrit määrittelevät, miten yksityiskohtaista tietoa lokitiedostoihin tallennetaan koko ohjelman tai pienempien kokonaisuuksien osalta.

Lokitiedostojen yleinen ongelma on se, että tietoa on sopivassa tilanteessa joko liian vähän tai liian paljon. Tarkin lokikirjoitus vie useissa sovelluksissa erittäin paljon tallennustilaa, jolloin sen käyttö ei ole mahdollista tuotantoympäristössä. Virheilmoitukset ja varoitukset kannattaa kuitenkin yleensä aina kirjoittaa lokitiedostoon. Oleellista tietoa voi puuttua senkin vuoksi, että kaikki tieto ei ole tavoitettavissa [Almgren and Lindqvist, 2001]. Jos järjestelmää seurataan esimerkiksi sovellustasolla, niin tällöin käyttöjärjestelmän tietoliikenneprotokollapinoista ei yleensä saada tietoa. Tietyssä protokollassa tai ajoympäristössä voi olla virheitä, jotka mahdollistavat onnistuneen hyökkäyksen, jota ei voi siis edes havaita sovellustasolla. Jos liikenne on salatussa muodossa, niin

sitä täytyy seurata tasolla, jossa se on vielä selväkielisessä muodossa. Hyvin tarkoissa lokitiedostoissa on se ongelma, että niiden suodattaminen vaatii mahdollisesti paljon työtä ja tietämystä järjestelmästä.

Nykyiset kiintolevyt mahdollistavat yhä suurempia lokitiedostoja. Eräs vaihtoehto on karsia vanhoja lokitiedostoja suodattimien avulla, jolloin esimerkiksi yli kuukautta vanhemmista lokitiedostoista poistetaan suuri määrä sellaista tietoa, jota ei enää todennäköisesti tarvita. Jos tiedostot vielä pakataan suodatuksen jälkeen, niin alun perin gigatavujen kokoinen lokitiedosto vie lopulta tilaa ehkä vain muutamia mega- tai kilotavuja.

Järjestelmästä voidaan kerätä audit trail -lokia siitä, kuka, miten ja milloin on muuttanut jotain tiettyä järjestelmän asetusta [Almgren and Lindqvist, 2001]. Audit trail voidaan toteuttaa lokitiedostojen muodossa, mutta ne kannattaa pitää erillään tavallisista lokitiedostoista. Audit trail -lokia muodostuu tyypillisesti hyvin vähän, joten lokirivit hukkuisivat tavallisiin suuriin lokitiedostoihin. Vanhoja audit trail -lokeja ei tarvitse myöskään siivota palvelimelta tavallisten lokitiedostojen tavoin, koska järjestelmiä konfiguroidaan yleensä muuhun käyttöön nähden vähän.

### 3.9. Asiakkaiden tietoturva ja luottamus

Tietoturvamurto suuren yrityksen tietojärjestelmään aiheuttaa usein runsaasti negatiivista julkisuutta, mikä ei ainakaan lisää loppukäyttäjän luottamusta kyseiseen palveluun ja yritykseen. Luottokorttinumeron lähettämistä kauppiaalle pidetään perinteisesti turvattomana. Ennemmin tai myöhemmin tulee todennäköisesti yleiseen maailmanlaajuiseen käyttöön turvallisempia maksumenetelmiä, joista yksi voi olla esimerkiksi kertakäyttöiset luottokorttinumerot.

Yleensä palvelu generoi jonkinlaisen kuitin ostotapahtumasta. PGP (Pretty Good Privacy)-ohjelmisto on varsin yleisessä käytössä. Sitä tai muuta salausohjelmistoa voitaisiinkin käyttää aikaleimalla varustetun kuitin allekirjoittamiseen. Krakkerin ei pitäisi kuitenkaan päästä käsiksi palvelimen salaiseen avaimeen, mitä ei voida yleensä taata. Organisaation julkinen avain voi olla jonkin kolmannen osapuolen allekirjoittama. Kolmannen osapuolen avain täytyy kuitenkin olla luotettavasti tiedossa asiakkaalla. Palvelimen avain *mitätöidään* (revocate) ennenaikaisesti, jos salainen avain joutuu ulkopuolisen haltuun.

Jotkin ohjelmavalmistajat toimittavat ohjelmiensa mukana digitaalisen allekirjoituksen. Jos ohjelmavalmistajan tai henkilön julkinen avain on haettu aiemmin ja sen aitouteen voi luottaa, niin allekirjoituksen tarkistuksen jälkeen voidaan olla varmempia siitä, että kukaan ei ole muuttanut asennuspakettia. Allekirjoituksia käytetään kuitenkin turhan harvoin. Periaatteessa sovelluksien

jakelu voitaisiin toteuttaa varsin turvallisesti. Sovellus kehitetään ensin suljetussa verkossa, joka ei ole yhteydessä mihinkään muuhun verkkoon. Valmis ohjelmapaketti allekirjoitetaan tässä kehitysympäristössä esimerkiksi älykortilla, jota säilytetään esimerkiksi dataturvakaapissa, joka on myös mieluusti murtoturvallinen. Paketti siirretään allekirjoituksen kanssa web-palvelimelle esimerkiksi USB-muistin avulla.

Yleensä lähdekoodi ja salausavain eivät ole näin hyvin suojattuja. Todennäköisesti harva käyttäjä viitsii tai osaa tarkistaa allekirjoituksia, mutta ainakin valveutuneet käyttäjät voivat tuntea olonsa hieman turvallisemmaksi. Krakkerit voivat kuitenkin onnistua muuttamaan tunnetunkin ohjelmatoimittajan lähdekoodia, binäärikoodia ja tarkistussummia, minkä saivat kokea kantapään kautta GNU-projektin ohjelmistojen käyttäjät [Vincent, 2003]. Hash-arvot on helppo väärentää web-palvelimella. Ohjelmistojen asennuspakettien digitaalisia allekirjoituksia ei kannata luoda automaattisesti palvelimella.

Digitaaliset allekirjoitukset kannattaa olla käytössä myös silloin, kun asiakkaan ohjelmistoa on mahdollista päivittää automaattisesti. Esimerkiksi virustorjunta- ja tietomurtojen tunnistusohjelmistojen päivitystiedostot kannattaa allekirjoittaa. Näin voidaan estää se, että koneelle asennetaan automaattisesti väärennettyjä tiedostoja nimipalvelin-, reititys- ja useiden muiden hyökkäysten avulla. Uuden version päivitys tapahtuu esimerkiksi seuraavan protokollan mukaisesti:

1. Haetaan uuden version tiedostolista ja sen allekirjoitus. Tiedostolista sisältää jokaisen listan tiedoston tarkistussumman sekä aikaleiman.
2. Tarkistetaan listan allekirjoitus ohjelmatoimittajan julkisella avaimella. Tuo avain on tiedossa, koska ohjelma on asennettu koneelle.
3. Jos listan allekirjoitus täsmää ja sen aikaleima on uudempi kuin nykyisen version, niin kopioidaan kaikki listan tiedostot koneelle.
4. Lasketaan ja tarkistetaan jokaisen siirretyn tiedoston tarkistussumma.
5. Jos kaikki tarkistussummat täsmäävät, niin uusi versio voidaan asentaa.

Jos algoritmin jossain kohdassa ilmenee ongelma, niin päivitystapahtuma epäonnistuu kokonaisuudessaan ja siitä ilmoitetaan käyttäjälle. Ohjelman tiedostot korvataan uusilla vasta kohdan 4 jälkeen. Jos aikaleimaa ei tarkisteta, niin krakkeri voi asennuttaa koneelle jopa vanhoja ohjelmaversioita, joissa on tunnettuja tietoturva-aukkoja. Tiedostolistasta voidaan julkaista ajoittain lista, jossa on merkitty se, että ohjelmasta ei ole tullut tietyn aikaleiman jälkeen uudempaa versiota. Tiedostolistassa täytyy kuitenkin olla lista kaikista viimei-

simmän varsinaisen päivityksen tiedostoista, koska kaikki eivät ole kuitenkaan päivittäneet ohjelmaa tuohon varsinaiseen uusimpaan versioon.

Allekirjoituksilla varmistutaan siitä, että tiedonsiirrossa ei ole tapahtunut virheitä tai tahallisia muutoksia. Tarkistussummien täytyy olla kryptologian kannalta turvallisena pidettyjä. Esimerkiksi SHA (Secure Hash Algorithm)-algoritmit soveltuvat tähän tarkoitukseen hyvin. Digitaalinen allekirjoitus kannattaa myös suorittaa turvallisen kokoisella avaimella.

Toisinaan voi ihmetellä, miten erilaisista paikoista tietoturva-aukkoja löytyy. Felten ja Schneider [2000] kertovat hyökkäystavasta, joka perustuu ajoitukseen. Web-selaimet hakevat omasta välimuististaan sivuja, jotka on haettu jo aiemmin. Haku vie aikaa sen mukaan, onko sivu jo välimuistissa vai ei. Aikojen perusteella voidaan päätellä suurella todennäköisyydellä, onko käyttäjä käynyt tietyllä sivulla. Web-sivun ylläpitäjä voi kerätä käyttäjästä tietoa esimerkiksi tavalla, joka toimii nähtävästi kaikilla nykyisillä selaimilla ja ilman Javascriptiäkin. Helpoiten hyökkäys voidaan toteuttaa Javascriptillä. Sillä voidaan mitata aika, joka kuluu siihen, että haetaan web-sivu joltain palvelimelta. Arvio voidaan tehdä seuraavan algoritmin mukaisesti [Felten and Schneider, 2000]:

1. Haetaan jokin tiedosto omalta palvelimelta ja mitataan siihen kulunut aika.
2. Haetaan ja mitataan aika sellaiselle sivulle, josta halutaan ajoitustietoa.
3. Haetaan toinen tiedosto omalta palvelimelta ja mitataan siihen kulunut aika.
4. Jaetaan mittaukset osumiin ja huteihin aikojen perusteella ja päätellään, onko käyttäjä käynyt sivulla, joka on mainittu kohdassa 2.

Mittausta voidaan tarkentaa tilastollisella päättelyllä. Lisäksi varmoja huteja voidaan generoida siten, että yritetään hakea tiedosto, jota ei ole olemassa. Suurempi otos mittauksia tuottaa yleensä tarkempia tuloksia. Ilman Javascriptiä, Javaa tai muuta dynaamista toteutusta voidaan todennäköisesti käyttää hyväksi esimerkiksi selaimen tapaa hakea sivulle monta kuvatiedostoa. Kuvatiedostojen näyttö voidaan vielä estää palvelimen luomassa HTML-koodissa, jolloin kuvat haetaan, mutta niitä ei näytetä sivulla. Tällaisia hyökkäyksiä voidaan toteuttaa myös esimerkiksi nimipalveluhakuihin, jos käyttäjän koneella on DNS-välimuisti [Felten and Schneider, 2000]. Hyökkäystapaa voidaan hankaloittaa hidastamalla web-hakuja satunnaisesti esimerkiksi proxy-palvelimella tai rakentamalla selaimen välimuisti eri tavalla. Tällainen tietoturva-aukko on käyttäjien kannalta kiusallinen, koska he pitävät selainhistoriaansa yleensä melko henkilökohtaisena. Hyökkäystapa on kuitenkin todennä-

köisesti melko harvinainen. Suurempi riesa on se, että käyttäjät sallivat edelleen HTML-muotoisten sähköpostiviestien kuvien latautumisen. Tällä tavoin roskapostin lähettäjä saa tietää, että sähköpostiosoite on toiminnassa.

# 4. Satunnaisuuden ongelma

Tässä luvussa kerron pääosin satunnaisuudesta, johon ei kiinnitetä toteutuksissa useinkaan riittävästi huomiota. Ilman aitoa satunnaisuutta paraskin salausalgoritmi menettää turvallisuutensa. Kryptologian turvallisuus kun perustuu yleensä ennen kaikkea salaiseen avaimeen, jota kukaan ei onnistu arvaamaan tavalla tai toisella.

## 4.1. Satunnaisuuden kerääminen

Ajatellaan että ihmisen tavallisen arkipäivän aikana näkemästä kuvasta saataisiin videokuva yhtenä suurena tiedostona äänen kera. Se sisältää lukuisan joukon mikrotapahtumia, joita ei voi ennustaa eikä toistaa. Ehkä bussi on myöhässä, ja odotellessa puusta putoaa pari lehteä satunnaisesti pyörien. Kiinnität niihin huomiota tai sitten et. Et voi tietää ketä kadulla tulee vastaan ja miten pukeutuneena. Väkijoukko liikkuu koko ajan eri suuntiin ja eri nopeuksilla. Joku jarruttaa voimakkaasti liikennevalojen vaihtuessa punaiseksi. Jäät ehkä juttelemaan tutun kanssa, kun törmäät häneen kadulla. Lounaalla jonkun maitolasi kaatuu ja kuuluu kirosanoja. Kun luet sanomalehteä, niin yksittäinen sivu on hieman rypyssä, ja lehden painatuksessa on yhdellä sivulla värivirhe. Kun viet illalla roskia, niin rusakko tulee vastaan ja pakenee paikalta. Vaikka ihminen yrittää luoda koko ajan virallisen kaavamaisia järjestelmiä, niin silti kaaos vallitsee. Kun tuosta tiedostosta ottaisi hash-funktion arvon, niin se sisältäisi taatusti tuloksen pituuden verran satunnaisuutta, kunhan hash-funktio ei itsessään aiheuta epäsatunnaisuutta lopputulokseen.

Kryptologiassa tarvitaan satunnaisuutta muun muassa salausavaimia varten. Mikroprosessorit ovat deterministisiä Turingin koneen [Black, 2008] tapaan. Perusongelma on se, että satunnaista lukua ei voida muodostaa tunnetusta lähtöarvosta deterministisellä algoritmilla, koska tulos voidaan päätellä kullekin syötteelle. Satunnainen data täytyy siis kerätä sellaisesta lähteestä, joka sisältää ei-ennustettavaa satunnaisuutta.

Entropiaa eli kryptologian kannalta satunnaisuutta on kerätty perinteisesti arpakuutioista, ruletista ja muista vastaavista kasinomaailman vekottimista. Muita lähteitä ovat esimerkiksi kameran ottamat kuvat laavalampusta ja äänikortin linjaliitännän kohina [Eastlake *et al.*, 2005] [Wikipedia, 2008a]. Nämä ovat *laitteistolla toteutettuja satunnaislukugeneraattoreita* (hardware random number generators). Äänikortin analogisista linjaliittimistä saadaan mahdollisesti termistä kohinaa, vaikka siihen ei olisi liitetty edes laitteita. Kaikki analogiset äänentoistolaitteet tuottavat enemmän tai vähemmän kohinaa, mutta kohinan täytyy olla kryptologista sovellusta varten mitattavissa. Heikkolaatui-

simmat laitteet tuottavat usein eniten kohinaa ainakin äänentoistossa. Useissa käytännön sovelluksissa entropiaa kerätään käyttäjän näppäinpainalluksien ajoituksista ja hiiren liikkeistä. Erityisesti näppäimistön käytössä täytyy varmistua, että näppäinpainalluksia ei puskuroida esimerkiksi näppäimistöajurin toimesta. Käyttäjän syötteistä saadaan kerättyä varsin hitaasti satunnaisuutta. Lisäksi ihmisten toiminnassa on ennustettavuutta.

Äänentoistokäyttöön on olemassa laitepohjaisia kohinalähteitä muun muassa kaiuttimien taajuusvasteen laskemiseen. Digitaalisia kohinalähteitä tulee kuitenkin ehdottomasti välttää, koska ne tuottavat determinististä pseudosatunnaista kohinaa. Analogisten piirien terminen kohina perustuu elektronien melko satunnaiseen liikkeeseen johtimissa ja komponenteissa. Sähköä johtavat materiaalit sisältävät elektronien liikkeen kannalta epätäydellisyyttä ja epäsymmetrisyyttä jo atomitasolla.

Digitaalikamera tuottaa kohinaa, koska sen kenno joutuu mittaamaan eidigitaalista maailmaa analogisesti. Suurin kohina saavutetaan suurella herkkyysarvolla kuten järjestelmäkameroissa alueella ISO 1600 - 6400, mutta kohinanvaimennukseen käytetyt algoritmit saattavat aiheuttaa tilastollista vinoutumaa ja vähentää satunnaisuutta. Kamerasta voidaan ottaa kuvia linssisuojus päällä, jolloin mustasta kuvasta saadaan enemmän tai vähemmän kohinaa. Matkapuhelimissa on nykyään pääsääntöisesti kamera, jota voisi myös käyttää yhtenä entropialähteenä. Osa käyttäjistä ei käytä kameraa lainkaan, joten se on vain yksi lähde muiden joukossa. Toinen vaihtoehto on ottaa automaattisesti kuvia kysymättä käyttäjältä, mutta se vie virtaa akusta ja on ehkä poliittisesti epäkorrektia. On aina tärkeää tarkistaa ajoittain, että entropialähde toimii odotetusti. Satunnaisuuden lähde kun voi mennä rikki tai heikentyä. Olisi hyvä antaa käyttäjälle mahdollisuus tarkistaa ajoittain entropian lähteeltä saatavaa dataa. Lisäksi toteutuksen täytyisi tehdä lähteelle automaattisesti tilastollisia testejä, vaikka ne eivät varmistakaan aitoa satunnaisuutta.

Yksi havainnollinen satunnaisuuden lähde olisi *tuulessa huojuvaan lehtipuuhun perustuva satunnaislukugeneraattori* (Windy Broad-leaved Tree Random Number Generator). Keksin tämän idean, kun seurasin alkukesästä hetken lehtien havinaa. Lehtipuuta kuvattaisiin kameralla silloin, kun tuulee ja puussa on lehtiä. Jokainen lehti on epäsynkronissa keskenään. Lisäksi hieman vaihteleva tuuli vaikuttaa jokaiseen lehteen käytännössä ennalta arvaamattomalla tavalla. Yksittäistä lehteä voi rinnastaa perinteisen tai sumean logiikan bittiin. Lehti on karkeasti ottaen toisessa kahdesta yhtä todennäköisestä asennosta. Vaihtoehtoisesti sen asento voidaan tulkita tarkemmin. Puusta otetaan vaikka puolen minuutin välein tarkka valokuva lyhyellä valotusajalla. Siitä täytyy erottua yksittäiset lehdet. Kuvasta otetaan hash-arvo hash-funktiolla tai loh-

kosalaajaa käyttäen. Kun lopuksi yhdistetään joukko hash-arvoja XOR-operaatiolla, niin saadaan laadukasta satunnaisuutta ja tarvittaessa todella paljon verrattuna vaikkapa klassiseen lava-lampun kuvaamiseen. Kuvaan ja siten hash-arvoon tulee satunnaisuutta myös kameran kuvaan tuomasta kohinasta. Satunnaisuuden määrä tietenkin romahtaa puun suhteen, kun on tyyntä tai puu varistaa lehtensä.

Nykyisistä laadukkaista ratkaisuista ovat ehkä yleisimpiä termiseen tai kvanttimekaaniseen kohinaan sekä oskillaattoreihin perustuvat menetelmät [Eastlake *et al.*, 2005], joista löytyy valmiita ratkaisuja tietokoneen lisälaitteina. Kohinaa saadaan esimerkiksi tarkoitukseen sopivasta diodista. Myös radiokohinasta voidaan kerätä satunnaisuutta radiovastaanottimella. Jonkin säteilylähteen kuten palovaroittimen säteilyä voidaan mitata Geiger-mittarilla. USB-väylään voidaan rakentaa pienikokoisia satunnaisuutta tarjoavia laitteita. Yksi hyvä ratkaisu on lisätä tietokoneiden emolevylle tai mikroprosessoriin erillinen satunnaislukugeneraattori, joka tuottaa ei-determinististä satunnaisuutta.

Tällä hetkellä yksi mielenkiintoinen ja edullinen toteutus on VIA:n prosessoreihin sisältyvä PadLock Security Engine, joka sisältää muun muassa varsin nopeiden rautatason AES- ja RSA toteutusten lisäksi ainakin yhden perusteellisen arvion [Cryptography Research, 2003] mukaan varsin laadukkaan ja erittäin nopean satunnaislukugeneraattorin. Se perustuu neljään vapaasti eri nopeudella pyörivään oskillaattoriin. Generaattoreita on prosessorin uusissa malleissa ainakin kaksi samassa prosessorissa, joiden tuotos yhdistetään miksausfunktiolla. Niistä saa ulos satunnaisuutta niin paljon, että esimerkiksi salausavaimen muodostaminen jopa kymmenestätuhannesta avaimen pituisesta hyvin satunnaisesta lohkosta ja niiden keskinäisestä XOR-operaatiosta ei hidastaisi sovellusta käytännössä lainkaan. Tällaisesta voi vain haaveilla esimerkiksi käyttöjärjestelmien satunnaislukugeneraattoria käytettäessä, joihin muuten kannattaa suhtautua varauksella. Ne kun saavat suuremman deterministisyyden vuoksi kerättyä varsin hitaasti aitoa satunnaisuutta. Niitä kutsutaankin *näennäissatunnaislukugeneraattoriksi* (pseudorandom number generator) siltä osin, kuin ne ovat deterministisiä.

Satunnaisuutta voidaan testata esimerkiksi tilastollisilla menetelmillä. Pseudosatunnainen data voi läpäistä tarkatkin satunnaistarkistukset, mutta sitä ei voi silti käyttää käyttötarkoituksiin, joissa tarvitaan aidosti satunnaista dataa. Turvallisin ratkaisu on kerätä satunnaisuutta useasta lähteestä, jotka eivät korreloi keskenään. Lähteiden arvot yhdistetään miksausfunktiossa, joissa eri lähteiden satunnaisuus yhdistetään. Näin saadaan satunnaisuutta, vaikka vain yksikin lähteistä toimisi oikein ja muut olisivat epäkunnossa. Jos vaatimukset

ovat hyvin korkeat, niin silloin voidaan miksata jopa usean erikoislaitteella toteutetun entropialähteen arvot.

Linux-käyttöjärjestelmän kernel tarjoaa /dev/random- ja /dev/urandom-laitteet, jotka voidaan sen normaalin toiminnan lisäksi räätälöidä keräämään entropiaa esimerkiksi äänikortin termisestä kohinasta. Näitä loogisia laitteita pidetään yleisesti melko turvallisina, jos niitä käytetään oikein [Fenzi and Wreski, 2004]. Laitteet eroavat siten, että /dev/random-laitteen lukeminen blokkaa, kunnes entropiaa on tarpeeksi, joten se soveltuu esimerkiksi pitkäaikaisten salausavainten luontiin. /dev/urandom ei blokkaa, joten jos uutta entropiaa on vain vähän, niin se ei palauta laadukasta satunnaisuutta. Se palauttaa dataa, joka perustuu entropiavaraston senhetkiseen tilaan. Tilastollisesti data on laadukasta, mutta se korreloi vahvasti siihen dataan, jonka pooli on palauttanut aiemmin.

Sessioavaimiin ja vastaavaan käyttöön sen turvallisuus on useimmiten riittävä. Ongelmia esiintyy /dev/urandom-laitteen kanssa silloin, jos käyttöjärjestelmässä ei tapahdu oikein mitään ja samanaikaisesti satunnaista dataa tarvitaan hyvin paljon lyhyessä ajassa. Jos käytetään /dev/random-laitetta ja luodaan esimerkiksi epäsymmetristä avainparia, niin silloin sovellus voi pyytää käyttämään näppäimistöä, hiirtä ja kiintolevyä entropian luomiseksi. Rautatason satunnaislukugeneraattorit ovat kuitenkin parempi vaihtoehto. Windowsin satunnaislukugeneraattoriin täytyy suhtautua erityisellä varauksella sen tietyistä versiosta löytyneiden heikkouksien [Dorrendorf *et al.*, 2007] vuoksi.

### 4.2. Satunnaislukugeneraattorin miksausfunktio

Satunnaisdatan miksaus eli useamman entropialähteiden yhdistäminen yhdeksi ja yleensä tilastollisen vinouman poistaminen voidaan toteuttaa monella tavalla. Eastlake ja muut [2005] esittelevät niistä muutaman. Ensin täytyy tietysti mainita klassinen XOR-menetelmä, jossa syötearvojen biteille suoritetaan normaali bittikohtainen XOR-operaatio. Menetelmä toimii yksinkertaisuudestaan huolimatta, jos satunnaislähteet eivät korreloi keskenään. Turvallisemmat menetelmät perustuvat pääosin symmetrisiin salausalgoritmeihin ja hash-funktioihin. Lohkosalaajaa voidaan käyttää miksaukseen siten, että ensin muodostetaan satunnaisesta datasta salausavain. Salausavaimella salataan uutta satunnaista dataa. Näin saadaan lopullinen satunnainen data. Epäsymmetrisiä salausalgoritmeja ei voi suositella, koska ne ovat huomattavasti hitaampia.

Lohkosalaajissa täytyy huomioida lohkon ja salausavaimen koko. Esimerkiksi DES (Data Encryption Standard) [NIST, 1999]-algoritmissa sa-

lausavaimeen tarvitaan 56 bittiä ja salattavaan lohkoon 64 bittiä. 64-bittistä salakielitekstilohkoa varten tarvitaan siis 120 satunnaista bittiä. AES- eli Rijndael-algoritmin erikoistuksessa [NIST, 2001] [Daemen and Rijmen, 1999] lohkon koko on 128 bittiä ja salausavaimen koko 128, 192 tai 256 bittiä, joten 128 tulosbittiä kohti tarvitaan syötettä vähintään 256 ja enintään 384 syötebittiä. Lohkosalaajista voi käyttää mitä tahansa, jota pidetään yleisesti turvallisena. Esimerkiksi IDEA (International Data Encryption Algorithm) sopii tarkoitukseen. Jos tarvitaan vain lohkon koon verran satunnaista dataa, niin syötedatasta voidaan muodostaa esimerkiksi selväkielinen lohko ja lopusta datasta useita avaimia [Eastlake *et al.*, 2005]. Selväkieliteksti salataan ensimmäisellä avaimella, salauksen tulos seuraavalla avaimella ja niin edelleen.

Tärkeintä on aina muistaa se, että vaikka deterministinen sekoitusfunktio olisi miten turvallinen tahansa, niin satunnaisuutta ei saada siitä ulos yhtään enempää kuin mitä sen syötteenä annetaan. Satunnaisuutta ei voi myöskään saada ulos lohkon kokoa enempää. Tarkastelen tällaista tilannetta esimerkin voimin. Syötteen pituus on 256 bittiä ja se sisältää tilastollisen vinoutuman vuoksi vain 61 bittiä satunnaisuutta. Syötteen alusta otetaan 128 bittiä AES-lohkoa varten. Lopuista biteistä saadaan 128-bittinen salausavain. Salauksen tuloksena saadaan 128-bittinen salakieliteksti, joka sisältää kuitenkin vain tuon 61 bittiä satunnaisuutta. Tällä menettelyllä saavutetaan kuitenkin yksi merkittävä etu. Alkuperäisen syötteen satunnaisuus on jakautunut tasaisesti 128 bitin joukkoon.

Esimerkiksi osa lohkosalaajista perustuu *Feistelin verkkoon* (Feistel network), jossa epälineaarisilla S-box-muunnostaulukoilla sekoitetaan selväkieliteksti salausavaimen ja S-box-taulukon avulla. Olennaista tilastollisen vinouman poistamisessa on lohkosalaajien ja hash-funktioiden *diffuusio* (diffusion)-ominaisuus. Sillä selväkielitekstin yksittäisen bitin vaikutus levitetään noin puoleen tulosbiteistä [Schneier, 1996]. *Sekoitusta* (confusion) ja diffuusiota toistetaan useita kierroksia, jolloin päästään eroon selväkielitekstin tilastollisista suhteista. Jos äskeisen esimerkiksi salakielitekstistä otetaan puolikas, niin se sisältää 30,5-bittiä satunnaisuutta.

Yksisuuntaiset hash-funktiot soveltuvat myös hyvin miksaukseen, sillä niiden käyttö on salausalgoritmeja yksinkertaisempaa, ja niille voidaan syöttää helpommin mielivaltainen määrä entropiaa. Lopputuloksesta saadaan salausalgoritmien tapaan satunnaisia bittejä sen verran, kumpi hash-arvon bittimäärästä tai lähtöarvon satunnaisuudesta on pienempi. Hash-funktiossa lohkon syötettä käsitellään usein lohkosalaajien lohkoa montakin kertaa suuremmissa osissa. Esimerkiksi yleisimmissä MD5- ja SHA-1 (Secure Hash Algorithm)-algoritmeissa syötettä käsitellään 512-bitin lohkoissa [Rivest, 1992]

[NIST, 2002]. Kyseiset algoritmin lisäävät itse lohkosta puuttuvia *täytebittejä* (padding), joten niille voi syöttää mielivaltaisen määrän dataa.

Entropialähteiden tuottama data on yleensä tilastollisesti vinoutunutta, jolloin joidenkin arvojen frekvenssi on toisia suurempi. Vinoutumat vähentävät lähteen satunnaisuutta, mutta niiden suoristamiseen löytyy useita menetelmiä, joiden tehokkuus vaihtelee. Satunnaisdatasta voidaan ottaa esimerkiksi pariteetti, tai siihen voidaan soveltaa esimerkiksi *nopeaa Fourier'n-muunnosta* (Fast Fourier Transform) tai Neumannin muunnostaulukkomenetelmää [Eastlake *et al.*, 2005]. Myös salausalgoritmien *korvauslaatikoita* (S-box) tai tavallisia pakkausalgoritmeja kuten Deflate voidaan soveltaa. Laadukkainta eli turvallisinta jälkeä saadaan kuitenkin todennäköisesti salausalgoritmeilla ja hash-funktioilla, kuten edellä kerroin miksausfunktioiden yhteydessä. Tällöin ei tarvitse pohtia ongelmia ja rajoitteita, joita esiintyy esimerkiksi pakkausalgoritmien ja Neumannin menetelmän yhteydessä. Kryptologisia ratkaisuja kannattaa käyttää, jos prosessoriaika riittää siihen. Lohkosalaajat ja hash-funktiot ovat käytännössä niin nopeita, että ne eivät vie merkittävästi resursseja pienimuotoisessa käyttötarkoituksessa kuten avainten luonnissa.

Esimerkiksi sessioavaimia varten tarvitaan toisinaan entropiaa enemmän kuin sitä ehditään kerätä. Tällöin voidaan käyttää esimerkiksi lohkosalaajaa siten, että satunnaisella salausavaimella salataan edellisen kierroksen siemenarvo [Eastlake *et al.*, 2005]. Usein entropiaa kerätään erityiseen pooliin eli varastoon, josta sitä voidaan noutaa tarvittaessa. Jos poolia ei käytetä pitkään aikaan ja sinne lisätään jatkuvasti entropiaa, niin poolin entropia kasvaa, eli sieltä on mahdollista saada yhä laadukkaampaa satunnaisuutta. Tällaisia pooleja ovat muun muassa edellisessä kohdassa mainitsemani käyttöjärjestelmien satunnaislukugeneraattorit.

### 4.3. Arpakuutiolaatikko

Kryptoanalyysin kannalta ihmisten keksivät salasanat eivät sisällä riittävästi entropiaa [Halevi and Krawczyk, 1998]. Esimerkiksi konsonanttien ja vokaalien keskinäisissä frekvensseissä ja sijaintisuhteissa voi olla runsaasti toistuvia piirteitä äidinkielestä. Jos halutaan aidosti turvallinen salasana, niin perinteiset arpakuutiot pelastavat päivän. McGraw ja Viega esittävät [2000] menetelmän, jossa salasanan jokaista merkkiä varten heitetään kolmea arpakuutiota. Silmälukujen perusteella valitaan taulukosta kerralla yksi merkki. Salasanan muodostaminen onnistuu toki yhdelläkin nopalla, kun yrittää muistaa jokaisella kierroksella kaksi silmälukua. Jos omistat laatikollisen arpakuutioita, niin kolmellakymmenellä nopalla saisi tällä taulukolla kerralla satunnaisen salasanan, jossa on enintään kymmenen merkkiä. Satunnaisuus edellyttää tietysti sitä, että

kasinohuijari ei ole päässyt käsiksi arpakuutioihin. Tämän tutkielman satunnaiset kymmenlukujärjestelmän numerot olen luonut taulukolla 4.3 kahden tai neljän arpakuution avulla.

| **Ensimmäinen noppa** | **Toinen noppa** | **Numero** |
|---|---|---|
| 1, 2 tai 3 | 1 | 1 |
| 1, 2 tai 3 | 2 | 2 |
| 1, 2 tai 3 | 3 | 3 |
| 1, 2 tai 3 | 4 | 4 |
| 1, 2 tai 3 | 5 | 5 |
| 1, 2 tai 3 | 6 | 6 |
| 4, 5 tai 6 | 1 | 7 |
| 4, 5 tai 6 | 2 | 8 |
| 4, 5 tai 6 | 3 | 9 |
| 4, 5 tai 6 | 4 | 0 |
| 4, 5 tai 6 | 5 tai 6 | hylkää |

Taulukko 4.3. Kymmenlukujärjestelmän numerot arpakuutioilla.

Läpinäkyvästä laatikosta, joukosta arpakuutioita ja edullisesta digitaalikamerasta voisi tehdä kotitekoisen satunnaislukugeneraattorin. Arpakuutiot ovat tässä laatikossa, jossa on myös läpinäkyvä kansi. Kamera on kiinnitetty kanteen. Käyttäjä ravistelee laatikkoa kuin juomasekoitinta ja asettaa sen lopuksi takaisin pöydälle. Hän voi vielä vilkaista, että noppia ei ole päällekkäin. Pieni sivusuuntainen ravistelu pudottaa päällekkäiset nopat. Tämän jälkeen hän aktivoi sovelluksen toiminnon, jolloin ohjelma ottaa kuvan kameralla. Joissakin kameroissa voi kylläkin tulla ongelmia lyhyen kuvausetäisyyden kanssa. Arpakuutioiden silmäluvut tunnistetaan hahmontunnistusalgoritmeilla. Mitä enemmän arpakuutioita on, niin sitä enemmän saadaan myös satunnaisia merkkejä. Kolmesta nopasta saadaan 6 ^ 3 eli 216 eri vaihtoehtoa, jolloin saadaan yksi merkki salasanaan. Tosin kombinaatioista 12 kolmikkoa ovat sellaisia, jolle ei löydy merkkiä. Tällöin kaikki kolme noppaa hylätään, koska muuten joidenkin merkkien todennäköisyys kasvaa eli tilastollinen jakauma vinoutuu.

Kokonaiseen tavuun tarvitaan neljä noppaa. Koska 6 ^ 4 = 1296, niin neljällä nopalla voitaisiin esittää yli 10 bittinen luku. Jos arpakuutioiden arvot muutetaan sopivalla taulukolla 10-bittiseksi, niin tällöin kuitenkin noin joka viides noppanelikko jouduttaisiin hylkäämään, sillä (6 ^ 4 - 2 ^ 10) / 6 ^ 4 =~ 0,21. Koska 6 ^ 13 > 2 ^ 33, niin kolmellatoista nopalla saataisiin yli neljä tavua. Vain joka sadas noppajoukko joudutaan hylkäämään, sillä (6 ^ 13 - 3 * 2 ^ 32) /

6 ^ 13 =~ 0.013. Neljän nopan joukkoina samaan tarvittaisiin vähintään 16 noppaa.

Toteutussyistä kannattaisi ehkä käyttää tarkastelussa kahden nopan pareja, koska tällöin kustakin noppaparista saadaan viisi bittiä, sillä 2 ^ 5 < 6 ^ 2 < 2 ^ 6. Noin joka kymmenes noppapari jouduttaisiin hylkäämään. Jos noppia on pariton määrä, niin viimeisestä nopasta saadaan kaksi bittiä todennäköisyydellä 2/3. Noppia voidaan näin lisätä ja poistaa koska tahansa vaikka yksitellen. Noppien lisääminen kasvattaa tulosbittien lukumäärää hyvin lineaarisesti. Pieniä ja edullisia noppia voisi laittaa esimerkiksi makeislaatikkoon esimerkiksi 51 - 55 kappaletta, jolloin saadaan hyvin todennäköisesti 128 bittiä jokaisella sekoituskerralla. Toki kasinoluokan noppiakin voi käyttää, mutta ne myös maksavat moninkertaisesti ja niitä täytyy etsiä erikoisliikkeistä tai ulkomailta. Tuplaamalla noppien lukumäärä saadaan entropiaa jo 256 bittiä. Ohjelmassa voisi käyttää valmiita ja omatekoisia merkistöjä. Kuhunkin salasanan merkkiin tarvitaan merkistön koon kaksikantaisen logaritmin verran satunnaisia bittejä eli esimerkiksi kirjainten ja numeroiden tapauksessa lg (2 * 29 + 10) =~ 6,09 eli 7 bittiä. Tavut voitaisiin näyttää heksadesimaalimuodossa.

Jos kamera on jo valmiiksi käytettävissä, niin rakennelman kustannukset ovat laitteiston suhteen lähellä nollaa. Jonkun täytyisi tietysti toteuttaa ohjelma. Muun muassa erilaiset nopat, kamerat, kuvakulman ja objektiivin aiheuttamat vääristymät, laatikot, valaistus ja kuvausetäisyydet täytyisi ottaa huomioon. Tämä voidaan toteuttaa osittain siten, että käyttäjä kalibroi ohjelman ennen sen varsinaista käyttöä rajaamalla kuvasta silmälukuja. On myös mahdollista rakentaa tunnistus siten, että samanaikaisesti voi olla käytössä erilaisia noppia. Tällainen laite on mahdollista valmistaa teollisesti varsin pieneksi, ja se voisi toimia USB-väylässä. Edes erillistä ohjelmaa ei välttämättä tarvita ehkä asetusten muuttamista lukuun ottamatta, koska laite voi toimia USB-näppäimistönä. Se syöttäisi satunnaisen tiedon kuten näppäimistö, kun laitetta on pidetty hetken paikallaan tai asetettu tasaiselle pinnalle ravistelun jälkeen. Laitteen käyttöliittymästä voisi valita tietyn merkistön tai binäärimoodin päälle, jolloin laite kirjoittaa tavut heksadesimaaliarvoina.

Laitteen suurin ongelma on käyttäjälle aiheutuva vaiva, mutta juuri salasanojen ja erityisesti PGP-avainten generoinnissa valveutunut hallitusten signaaleilta suojautuva foliohattu olisi todennäköisesti innoissaan tällaisesta vekottimesta. Laite on suurellakin noppamäärällä hidas, mutta siinä ei tarvitse ottaa samalla tavalla huomioon ennustettavaa vinoutta satunnaisuudessa kuten monissa muissa menetelmissä. Kahden peräkkäisen heittokerran lukujono ei saa kuitenkaan korreloida keskenään eli olla liian samanlainen, mikä on merkki

käyttäjän laiskuudessa laatikon ravistelussa. Taas törmätään siihen tosiasiaan, että käyttäjä on usein itse suurin tietoturvariski.

### 4.4. Salausalgoritmin, hash-funktion sekä protokollien valinta

Jos johonkin kryptologiseen tehtävään on jo olemassa valmis turvalliseksi tunnettu algoritmi, funktio tai protokolla, niin sellaista kannattaa käyttää oman ratkaisun sijaan. Näin varsinkin silloin, jos ei tiedä tarkalleen mitä on tekemässä. Minulla on tullut muutaman kerran vastaan tilanne, jossa ohjelmoija on ehdottanut varsin turvatonta omatekoista protokollaa. Esimerkiksi erään protokollan ratkaisu meni tietylle joukolle ohjelmistoalan asiantuntijoita, mutta ilmeisesti kukaan muu ei huomannut, että ratkaisu oli altis *toistohyökkäykselle* (replay attack). Kehitin tehokkaan hyökkäystavan alle vartissa. Välillä törmään satunnaisissa web-palveluissa ratkaisuihin, jotka näyttävät erityisen turvattomilta. Havainnoista voi aina tiedottaa palvelun ylläpitäjää, mutta varsinaista tietomurtoa ei voi testata ilman erityistä lupaa, sillä se olisi rikos. Analysointitaitoa voi kehittää lukemalla artikkeleita hyökkäystavoista, etsimällä järjestelmällisesti arvioitavan toteutuksen heikkoja lenkkejä sekä käyttämällä omaa mielikuvitusta.

Kun luin pitkästä aikaa artikkelin [Borisov *et al.*, 2008] WEP-algoritmin heikkouksista aiemmin lukemani lisäksi, niin ihmettelin protokollan suunnittelussa tehtyjä tahattomia tai tahallisia virheitä. Erityisesti alustusvektorin (IV) riittämätön bittimäärä ja vieläpä sen vaihtuvuuden valinnaisuus yhdistettynä RC4-*jonosalaajan* (stream cipher) [Schneier, 1996] turvattomaan käyttöön ovat pahoja virheitä. Kyse on kuitenkin standardista eikä muutaman ohjelmoijan tietämättömyydestä. Pahin tilanne on silloin, jos IV:lle on asetettu staattinen tai usein toistuva arvo tietoliikennelaitteessa.

Hyökkääjä pitää kirjaa IV:n ja salakielitekstin pareista. Muun muassa verkkoliikenteen otsikkokentissä on runsaasti staattisia osia. Tilastollisella kryptoanalyysilla saattaa löytyä hyvinkin nopeasti yksi salakieliteksti ja sitä vastaava selväkieliteksti. Tällöin saadaan tuolle IV:n ja salausavaimen parille RC4-jonosalaajan pseudosatunnainen tuloste eli key stream yksinkertaisesti ottamalla XOR-operaatio salakielitekstistä ja selväkielitekstistä. Nyt kaikki tietyn IV:n ja salausavaimen parilla salatut salakielitekstit saadaan selville ottamalla XOR-operaatio key streamista ja salakielitekstistä. Luonnollisesti jos IV ei muutu, niin kaikki tietoliikenne voidaan murtaa vakioajassa kunnes salausavain muutetaan. Ja kun se vaatii yleensä käyttäjiltä vaivaa, niin sitä ei muuteta. Koska IV ja salakieliteksti ovat tiedossa, niin 40-bittinen heikko salaus voidaan murtaa tarvittaessa vaikka brute forcella.

Algoritmin heikkouden lisäksi tietoliikennelaitteessa on mahdollisesti yleisesti tunnettu oletusarvo salausavaimena, tai arvoksi on asetettu heikko salasana ja sisältää siten heikosti satunnaisuutta. Sanakirjahyökkäyksen kannalta ei ole käytännössä mitään hyötyä, vaikka olisi käytössä 128- tai tarkemmin 104-bittinen WEP. Tämä oli esimerkki siitä, mitä kaikkea täytyisi ottaa huomioon, kun käyttää salausalgoritmeja.

Aioin alun perin kertoa tässä kohdassa algoritmien valinnasta eri käyttötarkoituksiin. Tulin kuitenkin toisiin ajatuksiin, sillä eri tilanteissa voi rikkoa turvallisuuden aivan liian monella tavalla, jos ei ole tietoinen hyvin monesta eri asiasta. Esimerkiksi kiintolevysalauksessa ja SSH-protokollassa täytyy ottaa huomioon aivan erilaisia asioita. Wikipediassa on sinänsä nykyään melko hyviä yhteenvetoja siitä, onko tietty algoritmi tai protokolla todistettu turvattomaksi. Kryptologiaan hurahtaneet asiantuntijat korjaavat todennäköisesti melko nopeasti mahdolliset virheet.

Tällä hetkellä toimivia valintoja ovat lohkosalaajista AES, IDEA ja vanha 3DES sekä epäsymmetrisistä salaajista muun muassa RSA. Jonosalaajien käyttöä kannattaa pääsääntöisesti välttää ja käyttää niiden tilalla lohkosalaajaa cipher-feedback (CFB)- tai output-feedback (OFB)-moodissa, jos se vain on mahdollista. Hash-funktioista SHA-256, SHA-384, SHA-512 ja ehkä Whirlpool ovat tällä hetkellä hyviä valintoja. Turvallisia lohkosalaajia pidetään kuitenkin hash-funktioita laadukkaampina, kun useiden laadukkainakin pidettyjen hash-funktioiden turvallisuudesta on löytynyt huomautettavaa.

Joihinkin käyttötarkoituksiin voi käyttää heikompiakin algoritmeja. Jokaisella algoritmityypillä on joukko rajoituksia, jotka pätevät enemmän tai vähemmän kaikkiin algoritmeihin. *Selväkielitekstiin perustuva kryptoanalyysi* (known plain-text cryptanalysis) on erityisen vaarallinen ja on mahdollista kaikille algoritmityypeille. Siksi täytyy usein käyttää riittävästi satunnaisia täytebittejä (padding). Tietoliikenne- ja muissa protokollissa on usein ongelmana ennakoitavat otsikkotiedot.

Suurimpaan osaan salauksen käyttötarkoituksia on jo kehitetty toimiva protokolla, joita kannattaa ensisijaisesti käyttää. Omatekoisiin protokolliin jää hyvin todennäköisesti tietoturva-aukkoja. Huipputurvallinen salausalgoritmi kun jättää niin helposti väärän turvallisuuden tunteen. Muun muassa *toistohyökkäykset* (replay attack), salakielitekstin bittimuutokset, täytebittien ennakoitavuus, otsikkotietojen ennustettavuus selväkielitekstissä ja aikaan perustuvat hyökkäykset ovat vasta alkua mahdollisille virheille. Jos haluaa perehtyä aihepiiriin tarkemmin, niin kannattaa ehdottomasti lukea jokin kirja, josta saa hyvän kokonaiskuvan. Pidän itse Schneierin klassista punaista ufon kuvalla varustettua Applied Cryptography [1996] -kirjaa erinomaisena, mutta

se vaatisi vähitellen jo uuden painoksen. Olen enemmän tai vähemmän tietoinen sen jälkeen muuttuneista asioista, joten se haittaa minua. Kirja toimii edelleen mainiona hakuteoksena.

# 5. Käyttäjätunnistuksen menetelmiä

Tässä luvussa käsittelen erilaisia tapoja tunnistaa oikea käyttäjä. Viimeisen viiden vuoden aikana on muuttunut käytännössä vain muutama asia. Ohjelmistopohjaisia salasanatietokantoja on tullut yhä enemmän saataville, ja Windowsiin voi nykyään kirjautua kannettavan tietokoneen sormenjälkitunnistimen avulla. Lisäksi yritykset ovat aiempaa kiinnostuneempia kehittyneemmistä tavoista tunnistaa käyttäjä. Silti salasanat on edelleen se tärkein käyttäjätunnistuksen tapa.

## 5.1. Kolme toisiaan täydentävää tapaa

Käyttäjätunnistus voidaan toteuttaa kolmella eri tavalla ja niiden yhdistelmänä [Liu and Silverman, 2001]. Ensimmäinen tapa tunnistaa käyttäjä on kysyä häneltä jotain, mitä hän tietää. Tällaista tietoa ovat esimerkiksi salasana, PIN-koodi ja muu henkilökohtainen tieto. Toinen tapa on pyytää häneltä jotain, mitä hän omistaa. Perinteiset avaimet ja tiedonsalausmenetelmien salaiset avaimet ovat esimerkkejä toisesta tavasta. Kolmas tapa on tunnistaa käyttäjä hänen fyysisten ominaisuuksiensa perusteella, eli toisin sanoen sen perusteella, millainen käyttäjä on.

## 5.2. Etäyhteydet ja käyttäjätietojen tallentaminen

Palvelinta ei kannata yleensä etäohjata työmikrolta palvelimen käyttöjärjestelmän ylläpitotunnuksilla, koska tällöin koko palvelin on vaarassa. Käyttäjien koneita kun uhkaavat tietoturvaongelmat muun muassa web-selaimissa, sähköpostiohjelmissa ja vertaisverkkosovelluksissa. Turvallisempi vaihtoehto on muodostaa käyttöliittymä, jossa etähallinnan oikeuksia on rajoitettu.

Käyttäjätiedot voivat sijaita esimerkiksi yhdessä tai useammassa XML-tiedostossa, mutta yleensä käytetään tietokantaa tai täysin omaa tiedostomuotoa. XML-dokumenttien käyttöoikeuksista, käyttösessioista ja käyttäjien tunnistamisesta ovat kirjoittaneet muun muassa Michiharu Kudo ja Satoshi Hada [2000]. Jos käyttäjiä on hyvin paljon, niin tietokanta on usein hyvä vaihtoehto, koska suuren käyttäjämäärän sovellukset käyttävät usein jo ennestään tietokantaa, ja esimerkiksi yksittäinen salasanatiedosto skaalautuu heikosti suurelle käyttäjämäärälle. Tietokantaan voidaan rakentaa aina uusia tauluja ja sarakkeita käyttöoikeuksia ja muita tietoja varten. Yksittäisen tiedoston etu on se, että siitä voi ottaa helposti varmuuskopion ja tarkistussumman sekä verrata sitä esimerkiksi UNIX:in diff-komennolla vanhempaan versioon.

## 5.3. Salasanatunnistus

### 5.3.1. Salasanan laatu ja tallennus palvelimelle

Perinteisesti käyttäjä tunnistetaan salasanan perusteella. Yleisin huono yhdistelmä tietoturvan kannalta on se, että salasana on huonosti valittu ja se välitetään salaamattomana Internetissä. Heikkojen salasanojen tunnusmerkkejä ovat muun muassa lyhyys, yksinkertaisuus ja suorat kytkökset käyttäjään [McGraw and Viega, 2000]. Syntymäaika, tuttavien nimet, kielten sanat ja omien harrastusten pohjalta muodostetut sanat ovat esimerkkejä näistä kytköksistä. Pitkiä ja aidosti satunnaisia salasanoja on vaikea muistaa, joten käyttäjillä on valitettavasti kollektiivinen taipumus valita heikkoja salasanoja.

Esimerkiksi näppäinten fyysisen sijainnin perusteella voidaan tehdä hieman satunnaiselta vaikuttava salasana. Esimerkiksi salasanan ”vfrtgbnhy” merkit muodostavat näppäimistöllä kuvion, jonka voi havaita näppäilemällä kirjaimet järjestyksessä. Jos krakkeri on päässyt käsiksi salasanatiedostoon, niin tuhannet yleisimmät kuviot voidaan käydä läpi muutamassa sekunnissa. Salasanoja murtavat ohjelmat kokeilevat esimerkiksi sanojen leet-muunnelmia [McGraw and Viega, 2000] [Raymond, 2003], joten heikosta salasanasta ei kannata tehdä uutta vain muuttamalla kirjaimia numeroiksi ja symboleiksi. Esimerkiksi salasanan ”apina” eräitä leet-muunnelmia ovat ”4p1n4” ja ”@|21|\|4”. Näitä muunnelmia on toki hieman vaikeampi löytää kuin suoraan sanakirjan sanaa, koska muunnelmia voidaan muodostaa useilla tavoilla.

Palvelin tai sovellus voi asettaa ehtoja käyttäjän salasanan rakenteelle. Salasanan täytyy olla esimerkiksi vähintään kahdeksan merkin pituinen, ja merkkijonon osana ei saa olla sanoja joukosta sanakirjoja. Käyttäjätunnus tai henkilön nimi ei saa olla myöskään osana salasanaa. Salasanan täytyy sisältää tietyn määrän pieniä ja suuria kirjaimia, numeroita ja erikoismerkkejä. Ehtojen on tarkoitus estää triviaalit salasanat. Liian tarkat ehdot pienentävät salasanaavaruutta ja ovat rasite käyttäjälle, jonka salasanat eivät kelpaa ohjelmalle. Krakkerit voivat toisinaan käydä läpi suuren määrän salasanoja verkonkin yli. Tällöin kuitenkin voidaan estää laadukkaan salasanan murtuminen viiveillä [Hämäläinen ja Mäkikangas, 2002]. Viive voi kestää vakiona esimerkiksi muutaman sekunnin, ennen kuin seuraavaa salasanaa voidaan yrittää kyseiselle käyttäjätunnukselle. Lukituksen täytyy koskea mieluummin käyttäjätunnusta kuin yhteyttä, koska hyökkääjä voi luoda samanaikaisesti lukuisia yhteyksiä samasta tai eri IP-osoitteista, millä pystyisi kiertämään yksinkertaisimpia lukituksia. Viive voi myös kasvaa virheellisten yrityskertojen kasvaessa [Hämäläinen ja Mäkikangas, 2002]. Samaa periaatetta käytetään usein myös elektronisissa kassakaapeissa, joissa numerojonoa käytetään avaimena.

Salasanan käyttökohde määrittelee sen, täytyykö salasanan siirtotapahtuma salata, ja onko salasanan satunnaisuudella suuri merkitys. Salasanan paljastumisella ei ole ehkä suurta merkitystä, jos palvelinsovellus pitää esimerkiksi käyttäjän asetuksista ja mieltymyksistä tavallaan anonyymiä profiilia. Kun kuvioon tulee mukaan raha tai luottamuksellinen tieto, niin luottamuksellisuudella ja tietoturvalla alkaa olla suurempi merkitys. Online-hyökkäyksiä on huomattavasti vaikeampi toteuttaa, jos käytössä on lukitukset ja viiveet. Palvelimelle jää myös jälkiä lokitiedostoihin. Tällaiset hyökkäykset on varsin helppo tunnistaa. Salasanan kannalta täytyisi pyrkiä estämään offline-hyökkäykset eli sellaiset hyökkäykset, jotka voidaan toteuttaa hyökkääjän koneella ilman yhteyttä palvelimeen. Ne voidaan yrittää välttää tietoliikenteen tasolla usein käytännössä vain kryptologian avulla [Halevi and Krawczyk, 1998]. Yleensä kun voidaan olettaa, että tietoliikennettä on mahdollista salakuunnella.

Syksyllä 2007 murtauduttiin Suomessa useille palvelimille, joista saatiin noin 80000 käyttäjätunnusta [CERT-FI, 2007], niiden salasanat selväkielitekstinä tai hash-arvona sekä sähköpostiosoitteita. Murtautuja laittoi tuon tekstitiedoston yleisesti saataville Internetiin. Palvelimilla ei käytetty lainkaan salt-arvoja salasanatiedostoissa. Tämä mahdollistaa sen, että esilasketuilla Internetistä saatavilla rainbow table -tiedostoilla voidaan murtaa suuri osa niistäkin salasanoista, jotka ovat aidosti satunnaisia ja varsin pitkiä kuten 14-merkkisiä. Yksi tällainen toteutus on RainbowCrack. Internetistä löytyy myös web-palveluita, joihin voi syöttää hash-arvoja, ja palvelin laskee ja etsii rainbow tablesta vastaavan salasanan. Vaikka käytettäisiin pitkääkin salt-arvoa, niin turvattomat ja lyhyet salasanat voidaan murtaa silti sanakirjahyökkäyksellä ja brute forcella. Kun käyttäjätunnuksia on noin paljon, niin listasta voi jopa etsiä mielenkiinnosta hash-arvoja, jotka on muodostettu syöttämällä turvaton salasana kuten kirosana käsin MD5- tai SHA-1-ohjelmalle.

Nämä palvelimet eivät olleet yhteiskunnan kannalta kriittisiä. Suurin ongelma on se, että monet käyttävät samaa salasanaa useassa tai jopa kaikissa paikoissa. Ilmeisesti tuota tekstitiedostoa onkin käytetty murtautumaan myös muihin tärkeämpiin järjestelmiin. Mitä tästä opimme? Samaa salasanaa saa käyttää vain yhdessä palvelussa, ja salasanojen tallennuksessa täytyy käyttää erikseen kullekin salasanalle satunnaisesti valittua salt-arvoa. Salt-arvoksi voi ottaa vaikka 128-bittisen pseudosatunnaisen heksadesimaaliarvon, joka vie tilaa 32 merkkiä. Se ei ole suuri uhraus käyttäjää kohden nykyisillä mammuttimaisilla kiintolevyillä. Tämän jälkeen rainbow table -hyökkäys ei toimi käytännössä lainkaan, koska tuollainen taulu täytyisi muodostaa erikseen jokaiselle käyttäjälle.

### 5.3.2. Salasanan siirto palvelimelle

On monia tapoja välittää salasana turvallisesti palvelimelle. Tilanne on helpoin silloin, kun sekä käyttäjällä että palvelimella on epäsymmetrinen avainpari. Tällöin ei tarvita edes salasanaa. Usein kuitenkin vain palvelimella on tällainen avainpari, ja käyttäjä tunnistaa itsensä salasanalla. Jos salasana välitetään salattuna palvelimen julkisella avaimella, niin salatun selväkielitekstin mukana on oltava palvelimen luoma aikaleima *toistohyökkäyksen* (replay attack) estämiseksi. Sellaiseksi käy esimerkiksi palvelimen luoma satunnainen bittijono yhdistettynä tietokannassa säilytettävään juoksevaan lukuun. Tärkeintä on varmistua siitä, että aikaleima on vikatilanteissakin yksikäsitteinen. Sen vuoksi pelkkä tietokoneen virhealtis kellonaika ei ole kovin hyvä valinta. Aikaleima on voimassa vain hyvin lyhyen ajan, ja palvelin pitää kirjaa voimassa olevista aikaleimoista. Kun käyttäjä on tunnistettu, niin aikaleima vanhenee välittömästi. Aikaleimaa ei tarvitse välttämättä salata, kun se siirretään palvelimelta käyttäjän koneelle.

Aikaleima, käyttäjätunnus, salasana ja mahdollinen käyttäjän luoma sessioavain salataan yhdessä palvelimen julkisella avaimella, mutta ei ilman aidosti satunnaisia täytebittejä (padding). Tämän voisi tehdä myös siten, että salaa lohkosalaajalla ensin lohkon verran aidosti satunnaista dataa cipher block chaining (CBC)-moodissa käyttäen salausavaimena aikaleiman hash-arvoa. Muissa lohkoissa salataan varsinainen viesti. Lopputulos salataan palvelimen julkisella avaimella ja lähetetään palvelimelle. Palvelin purkaa viestin salaisella avaimellaan, purkaa lohkosalaajan salauksen ja varmistuu käyttäjän henkilöllisyydestä. Salasanan yrityskerroista pidetään tietenkin kirjaa kuten normaalisti erilaisissa palveluissa. Koska hyökkääjällä on tiedossaan palvelimen julkinen avain, niin ilman täytebittejä hän saisi nopeasti selville salasanan known-plaintext attack -hyökkäyksen avulla.

Omien padding-ratkaisujen sijaan voi suosiolla käyttää esimerkiksi RSA:n tapauksessa RFC 3447 -dokumentin [Jonsson and Kaliski, 2003] RSAES-OAEP-protokollaa. Kryptologiassa yksityiskohdat ratkaisevat. Padding-ratkaisusta riippumatta täytyisi aina varmistua, että salaukseen täytyisi tuoda vähintään 128 - 256 bittiä salaista aitoa satunnaisuutta. Jos satunnaisuutta on mukana vain esimerkiksi 40 bittiä, niin murtaminen vaatii vain 2 ^ (40 + salasanan sisältämä satunnaisuus bitteinä) verran salauskertoja julkisella avaimella.

Yksi vaihtoehto on luoda ensin yhteinen symmetrinen salainen avain siihen tarkoitetulla algoritmilla (key-exchange algorithm) kuten Diffie-Hellmanilla. Palvelin allekirjoittaa salaisella epäsymmetrisellä avaimellaan tuosta sessioavaimesta ja muista tiedoista otetun hash-arvon. Näin käyttäjän sovellus voi varmistua siitä, että toisessa päässä on kyseinen palvelin. Muutenhan

man-in-the-middle attack olisi mahdollinen. Tällaista ratkaisua käytetään muun muassa SSH-2:ssa [Ylönen *et al.*, 2006]. Tämän jälkeen luodaan samalla tavalla sessioavain. Jos käytettäisiin vain yhtä avainta ja sessioavain saataisiin murrettua, niin salasana selviäisi hyökkääjälle. Epäsymmetriset algoritmit ja protokollat on helppo toteuttaa turvattomasti, joten ohjelmoinnissa kannattaa käyttää mieluummin valmiita kirjastoja. Jos se ei ole mahdollista, niin nykyaikaisista tietoliikenneprotokollista kuten SSH-2:sta tai SSL:n ja TLS:n uusimmista versioista voi ottaa mallia salasanan välittämiseen. Erityistä tarkkuutta kuitenkin tarvitaan.

Myös uudentyyppisiä menetelmiä salasanan syöttämiseen tutkitaan. Salasana voidaan korvata esimerkiksi joukolla kuvia, joita täytyy valita ruudulta tietyssä järjestyksessä. Tällaisissa menetelmissä yritetään useimmiten käyttää hyväksi ihmisen visuaalista muistia. Kuvia voi olla esimerkiksi viidessä rivissä ja sarakkeessa. Ne voivat vaihtua jokaisen valinnan jälkeen. Esimerkiksi kymmenen valinnan salasana-avaruudessa olisi 25 ^ 10 eli noin 2^46 salasanaa. Tällaisen visuaalisen salasanan siirtäminen palvelimelle voidaan tietenkin toteuttaa samalla tavalla kuin tavallisen salasanan.

### 5.3.3. Kertakäyttöiset salasanat

Kertakäyttöiset salasanat on hyvä vaihtoehto tiettyihin sovelluksiin, joissa salasanalistojen jakelun aiheuttama vaiva ja kustannukset ovat perusteluja. Tällöin käyttäjälle lähetetään käyttötarkoituksen kannalta riittävän turvallisella tavalla salasanalista, jossa on erillinen aidosti satunnainen salasana esimerkiksi jokaiselle palvelun käyttökerralle. Lista voidaan toimittaa muun muassa perinteisellä postilla tai sähköpostilla käyttäjän PGP-avaimella salattuna. Lista voi joutua vääriin käsiin ja se on helppo kopioida [Halevi and Krawczyk, 1998]. Listassa ei kannata yleensä mainita käyttäjätunnusta. Lisäturvaa saadaan muun muassa pysyvällä mutta vaihdettavalla salasanalla, jolloin käyttäjätunnistukseen tarvitaan käyttäjätunnus, pysyvä salasana ja seuraava kertakäyttöinen salasana.

Yksi erilainen vaihtoehto on kuvattu muun muassa RFC 2289 -dokumentissa [Haller *et al.*, 1998]. Kyseisessä OTP (One-Time Password system)-protokollassa ei kuitenkaan käytetä salasanalistaa, mutta sen avulla voidaan periaatteessa luoda sellainen. OTP on hyvin samanlainen vanhemman S/KEY-protokollan kanssa, jonka pohjalta se on kehitettykin [Haller, 1995] [Haller *et al.*, 1998]. OTP tukee uudempia hash-funktioita kuten MD5:sta ja SHA-1:stä. Sen kanssa voi käyttää periaatteessa mitä tahansa hash-funktiota, kun taas S/KEY tukee ainoastaan MD4:sta.

Aluksi käyttäjä valitsee tai generoi satunnaisen salasanan. Lisäksi tarvitaan *siemenarvo* (seed) ja kierrosluku N. Siemenarvo ja salasana yhdistetään, ja tulos syötetään rekursiivisesti hash-funktiolle N kertaa. Jokaisella kierroksella muutetaan tulos tavukohtaisella XOR-operaatiolla 64-bittiseksi. Lopputulos lähetetään ja tallennetaan palvelimelle aiemmaksi kertakäyttöiseksi salasanaksi. Sen lisäksi palvelimelle tallennetaan käyttäjäkohtaisesti vain kierrosluku N ja siemenarvo, joka pysyy samana, kunnes käyttäjä vaihtaa salasanansa tai kaikki kertakäyttöiset salasanat on käytetty. Palvelin ei saa koskaan tietää käyttäjän varsinaista salasanaa.

Käyttäjätunnistuksessa palvelin lähettää siemenarvon ja kierrosluvun N, jotka voidaan lähettää julkisena verkon yli. Siemenarvo ja käyttäjän salasana yhdistetään asiakkaan koneella. Seuraavaksi yhdistelmä syötetään rekursiivisesti hash-funktiolle N - 1 kertaa. Jokaisella kierroksella muutetaan jälleen tulos tavukohtaisella XOR-operaatiolla 64-bittiseksi. Näin saadaan kyseisen kierroksen kertakäyttöinen 64-bittinen salasana. Jos esimerkiksi N on 100, niin hash-funktiota kutsutaan rekursiivisesti 99 kertaa.

Palvelin suorittaa käyttäjän lähettämälle salasanalle kerran hash-funktion ja muuttaa tuloksen samalla tavalla 64-bittiseksi. Jos tulos vastaa palvelimelle tallennettua aiempaa kertakäyttöistä salasanaa, niin käyttäjän salasana on oikea. Siinä tapauksessa kierrosluvun arvoksi asetetaan N - 1. Koska hash-funktiot ovat yksisuuntaisia, niin palvelin ei voi laskea seuraavaa salasanaa edellisestä, mutta se voi tarkistaa salasanan edellisen perusteella. Viimeinen avain saadaan suorittamalla vain yksi hash-operaatio. Tämän vuoksi N määrittelee, montako kertakäyttöistä salasanaa on mahdollista generoida. Salasanat välitetään esimerkiksi englanninkielisinä 1 - 4 merkkisinä sanoina. Sanat ovat 2048 sanan sanastosta, joten 64 bitin kertakäyttöinen salasana voidaan välittää kuudella sanalla. Jokainen sana kun sisältää 11 bittiä informaatiota. Kaksi ylimääräistä bittiä käytetään OTP:ssä tarkistussummana. Protokolla tukee myös salasanan heksadesimaaliesitystä.

Menetelmän suurin etu on se, että salasanat ovat kertakäyttöisiä, jos hash-funktiota tai alkuperäistä salasanaa ei pystytä murtamaan. Hyökkääjä ei saa hash-arvosta selville alkuperäistä salasanaa, paitsi jos salasana on heikko. Juuri heikko salasana on koko järjestelmän suurin riski. Turvallisinkin rekursiivisesti suoritettu hash-funktio saadaan murrettua nopeasti, jos salasana ei sisällä riittävästi satunnaisuutta. Lisäksi hash-arvon muuttaminen esimerkiksi 160-bittisestä 64-bittiseksi mahdollistaa hyökkäyksen, jossa lukuisalla joukolla vääriä salasanoja käyttäjätunnistus onnistuu. Hyökkääjä kun tietää kaiken mitä palvelinkin ja voi jopa käytännössä laskea brute forcella seuraavan kertakäyttöisen salasanan. Hyökkäys voidaan tehdä offline-tilassa ja vaatii keskimäärin 2

^ 63 yritystä. Tällöin salasanatietokanta toki korruptoituu käyttäjän kannalta. Siispä jotta protokolla olisi turvallinen, niin salasanan täytyisi sisältää satunnaisuutta mieluiten jopa hash-funktion tulosteen verran. Lisäksi protokollaa täytyy muuttaa tuon hash-arvon vähennysfunktion suhteen. Sen voisi poistaa kokonaan. Silloin protokolla olisi teoriassa varsin turvallinen. Suurin riski tulee tällöin pitkän salasanan muistamisesta tai tallentamisesta käyttäjän koneelle.

### 5.3.4. Haaste-vaste-menettely

Myöskään haaste-vaste-menettelyssä käyttäjän salasanaa ei siirretä selväkielisenä [Hämäläinen ja Mäkikangas, 2002]. Sekä palvelimella että asiakkaalla on tiedossa salasana. Palvelin lähettää satunnaista dataa, joka syötetään yksisuuntaiseen funktioon yhdessä salasanan kanssa. Tuloksena syntyy yleensä jonkinlainen hash-arvo. Tämä arvo lähetetään palvelimelle, joka vertaa tulosta arvoon, joka muodostetaan samoin kuin asiakkaan koneella. Jos arvot täsmäävät, niin asiakas on tunnistettu oikeaksi.

Menettelyn kriittisimmät kohdat ovat vastefunktio, haasteen pituus ja erityisesti salasana. Jos haaste on pitkä ja hyvin satunnainen sekä funktio yleisesti turvallisena pidetty, niin tunnistuksen luotettavuus on kiinni salasanasta. Hyökkääjällä on tiedossa vastefunktio, haaste sekä vaste. Turvallisesta vastefunktiosta ja haasteesta ei ole hyötyä, jos salasana on heikko [Halevi and Krawczyk, 1998]. Krakkeri voi offline-tilassa kokeilla helppoja salasanoja vastefunktioon yhdessä haasteen kanssa. Hän on löytänyt suurella todennäköisyydellä oikean salasanan, jos löydetty vaste on sama kuin vakoiltu vaste. Tällainen yksinkertainen hyökkäys voidaan estää salaamalla tietoliikenne tavalla tai toisella. Haaste-vaste-menettely toteutetaan kuten aiemminkin. Käyttäjän lähettämä vaste salataan esimerkiksi palvelimen julkisella avaimella tai symmetrisellä sessioavaimella. Myös tietoliikenne palvelimelta asiakkaalle voidaan salata. Haaste-vaste-menettely on helppo toteuttaa, jos asiakassovelluksessa voidaan suorittaa ohjelmakoodia. Vaste voidaan laskea vaikka Java-appletissa tai Javascriptillä.

Haaste-vaste-menettely voidaan toteuttaa myös laitteistopohjaisesti, jolloin käyttäjä syöttää haasteen pienikokoiseen laitteeseen, joka laskee vasteen. Tähän tarkoitukseen sopii esimerkiksi matkapuhelin tai kämmentietokone. RSA Security -nimisellä yrityksellä on ollut jo kauan SecurID-tuote, joka laskee kuusinumeroisen vasteen ajan ja salaisen seed-arvon perusteella, joten käyttäjän ei tarvitse syöttää laitteeseen haastetta. Vaste vaihtuu esimerkiksi minuutin välein. Varsinainen seed-arvo on sinänsä todennäköisesti aidosti satunnainen ja vastefunktio turvallinen. Sekä laite että palvelin tietävät tämän salaisuuden. Palvelin osaa laskea vasteen kellonajan ja seed-arvon perusteella. Tämäntyyp-

pinen tuote ei ole useinkaan järkevä suurelle käyttäjämäärälle, koska vastelaitteet ja lisenssit eivät ole todellakaan mitään ilmaisia. Jos hyökkääjä pääsee käsiksi palvelimella sijaitseviin seed-tiedostoihin, niin kaikkien käyttäjien laitteet ovat saman tien elektroniikkajätettä.

Hämäläinen ja Mäkikangas mainitsevat puhelimen käytön salasanan välittämiseen [Hämäläinen ja Mäkikangas, 2002]. Yksi vaihtoehto olisi sellainen, jossa käyttäjän täytyy lausua aina satunnainen lause, jonka tietokone lausuu ensin puhelimessa. Käyttäjä tunnistetaan käyttäjätunnistukseen viritetyillä puheentunnistusalgoritmeilla. Tietokone lausuu listasta seuraavan käyttäjän määrittelemän kertakäyttöisen salasanan sen jälkeen, kun käyttäjätunnistus on suoritettu onnistuneesti. Tällä tavoin yksinkertainen käyttäjän puheen tallentaminen ei riitä siihen, että krakkeri saisi selville uusia salasanoja. Samalla käyttäjä voi kantaa aina turvallisesti mukanaan kertakäyttöisiä salasanoja ympäri maailmaa. Huono puhelinyhteys ja flunssa ovat kuitenkin varsin ongelmallisia käyttäjätunnistuksen kannalta. SIM-kortti on myös kloonattavissa sekä puhelut kuunneltavissa. Lisäksi käyttäjän äänestä on ehkä mahdollista luoda profiili, jonka avulla voidaan hämätä käyttäjätunnistusta.

### 5.3.5. Sertifikaatit

Käyttäjän täytyisi usein varmistua palvelimen oikeellisuudesta. Entä jos hän on ulkomailla ja hänellä ei ole palvelimen julkista avainta? Hän voi ottaa ennen matkaa mukaansa hash-arvon palvelimen julkisesta avaimesta. Bitit voidaan koodata merkkijonoksi, jonka käyttäjä kirjoittaa esimerkiksi paperille. Matkalla hänen tarvitsee vain tarkistaa, että merkkijonot täsmäävät. Helppolukuinen koodausmuoto on mahdollista toteuttaa monella tavalla. Hash-arvo voidaan esittää esimerkiksi heksadesimaalimuodossa tai englannin kielen sanoina kuten OTP-protokollassa. Muun muassa SSH-ohjelmat näyttävät avaimen sormenjäljen sanamuodossa. Avain kannattaa tallentaa asiakasohjelmassa automaattisesti ensimmäisellä yhteyskerralla, kun käyttäjä on hyväksynyt sen. Tämän jälkeen ohjelma häiritsee käyttäjää vain silloin, kun sertifikaatti on vaihtunut. Vastaavasti myös käyttäjätunnistus voidaan toteuttaa käyttäjän luomalla avainparilla. Palvelimella on tällöin tiedossa käyttäjän julkinen avain. Esimerkiksi SSH-2-protokollassa on mahdollista korvata tällä tavoin salasanatunnistus.

CA:t (Certificate Authority) myöntävät sertifikaatteja eli digitaalisia allekirjoituksia, joilla jonkin palvelimen julkisen avaimen oikeellisuus voidaan todentaa. Sertifikaatit ovat vaivattomampi vaihtoehto kuin hash-arvot. CA allekirjoittaa omalla salaisella avaimellaan tässä tapauksessa kohdepalvelimen julkisen avaimen, jonka jälkeen asiakas voi varmistua kohdepalvelimen julki-

sen avaimen oikeellisuudesta. Web-selaimissa kohdepalvelimen domainin täytyy vastata sertifikaatin tietoja. Muun muassa sertifikaattien tarkistuksesta on löytynyt useita tietoturva-aukkoja. Usein suurin riski on kuitenkin käyttäjä, joka ei huomaa tai ymmärrä väärää domain-nimeä, sertifikaatin puuttumista tai selaimen varoituksia väärään domainiin liittyen. Tällöin man-in-the-middle attack on mahdollinen.

### 5.3.6. Vahva käyttäjätunnistus

Kyseessä on *vahva käyttäjätunnistus* (strong authentication, two-factor authentication), jos käyttäjän tunnistamiseen käytetään vähintään kahta menetelmää [Hämäläinen ja Mäkikangas, 2002]. Menetelmillä tarkoitetaan tässä sitä, että käyttäjä omistaa jotain, tietää jotain ja on jotain. Esimerkkinä käyvät sirukortti, salasana ja sormenjälki. Salasanan ja sormenjälkitunnistuksen yhdistelmä on täten vahvan käyttäjätunnistuksen esimerkki. Erilaisten menetelmien yhdistäminen ei tietenkään takaa turvallisuutta, mutta se vaikeuttaa hyökkäystä. Pelkän salasanan selvittäminen kun ei riitä.

Esimerkiksi sirukorttien käytössä on tietoturvaongelmia, koska krakkeri voisi allekirjoittaa jotain ylimääräistä tietoa, kun kortti on kiinni tietokoneessa. Yksi vaihtoehto on lisätä korttiin tai kortinlukijaan näyttö, jossa tiedon voi tarkistaa ennen allekirjoitusta. Allekirjoitus varmistetaan vielä esimerkiksi tunnusluvulla ja sormenjälkitunnistimella, joka on integroitu kortinlukijaan. Ratkaisu on jo melko turvallinen, jos mikrosiru olisi rakennettu fyysisesti turvalliseksi, ja kryptologisessa toteutuksessa ei olisi heikkouksia. Hieman kalliskin laite muuttuu yleistyessään markkinalakien vuoksi edullisemmaksi. Tietenkin itse lukijalaite on mahdollista väärentää, jolloin käyttäjä uskoo turvallisuuteen, kunnes näkee esimerkiksi pankin tiliotteen. Käyttäjän täytyisi siis kantaa suuri osa lukijalaitteen toiminnallisuudesta mukanaan, jotta saavutettaisiin parempi turvataso.

Useissa tapauksissa riittävä turvallisuus saavutetaan jo vahvoilla kryptologisilla ratkaisuilla. Vahva käyttäjätunnistus aiheuttaa käyttäjälle helposti enemmän vaivaa. Toisin sanoen tuotteen käytettävyys kärsii, mikä onkin usein vahvaan käyttäjätunnistukseen perustuvien ratkaisujen ongelma. Osassa palveluita käyttäjälle olisi hyvä sallia mahdollisuus valita menetelmät, joita käytetään käyttäjätunnistuksessa.

Käyttäjätunnistukseen liittyen ei ole rajapintaa, joka sopisi kaikkiin tilanteisiin ja kattaisi kaikki tunnistuslaitteet ja ohjelmistot. Kuitenkin esimerkiksi RFC 2865 [Rigney *et al.*, 2000] ja RFC 3748 [Aboba *et al.*, 2004] -dokumenteissa kuvatut RADIUS ja EAP (Extensible Authentication Protocol) sekä lisäksi PEAP (Protected EAP) soveltuvat monenlaiseen verkon yli tapahtuvaan käyttäjätun-

nistukseen. Nuo protokollat ovat melko avoimia, ja valmistajat voivatkin laajentaa niitä ja lisätä puuttuvat tunnistusmenetelmät. Tietenkin useissa sovelluksissa kuten pankkiholvissa, kannettavassa laitteessa tai paikallisesti tietokoneella suoritettavassa sovelluksessa ei tarvita erikoisia protokollia. Ne toisivat vain lisää ohjelmarivejä ja ylimääräistä monimutkaisuutta eli todennäköisesti enemmän tietoturva-aukkoja.

## 5.4. Biometrinen tunnistus

Biometristä käyttäjätunnistusta on käsitelty useaan kertaan melko yleisellä tasolla useimmissa ICT-alan lehdissä ja verkossa. Tämän vuoksi kertaan ensin vain lyhyesti perinteiset biometriset tunnistusmenetelmät. Sen jälkeen kerron syvällisemmin erilaisista siihen liittyvistä tekniikoista.

### 5.4.1. Yleistä biometrisistä tunnistusmenetelmistä

Biometriset ja muut uudet tunnistusmenetelmät on otettu laajempaan käyttöön erityisesti WTC-iskujen jälkeen. Yhdysvallat yrittää edelleen parantaa lentokenttien ja rajojen turvallisuutta muun muassa automaattisella seurannalla ja biometrisillä passeilla. Tämä on todennäköisesti nopeuttanut hieman erityisesti sormenjälkilukijoiden yleistymistä osaksi yritysmaailman ja kotikäyttäjien laitteita. Myös standardien kehittäminen on päässyt vauhtiin. Konsortiot ja työryhmät kehittävät standardeja erilaisiin tarkoituksiin [NIST, 2008].

Biometriset tunnistusmenetelmät mittaavat käyttäjän fyysisiä ominaisuuksia tai käyttäytymispiirteitä. Sormenjäljet, kasvonpiirteet, kämmenen muoto, verkkokalvo ja iiris ovat fyysisiä ominaisuuksia, joiden avulla käyttäjä on perinteisesti tunnistettu. Henkilö voidaan tunnistaa myös esimerkiksi jalkapohjan, hampaiden tai DNA:n perusteella. Myös elektroninen nenä on mahdollinen, eli käyttäjä tunnistetaan tuoksun tai hajun perusteella. Ihmisen käyttäytymispiirteitä sisältäviä ominaisuuksia ovat muun muassa kävelytyyli, ääni ja allekirjoitus.

Tietokone voisi tunnistaa käyttäjän myös esimerkiksi hänen tekemiensä askareiden perusteella. Kun käyttäjä tulee kotiin, niin hän todennäköisesti suorittaa tietyt rutiinit ja tarkentaa katseensa tietyllä tavalla. Käyttäjän ei tarvitse yleensä silmäillä asunnon esineitä ja huoneita tarkasti, koska hän on tutussa ympäristössä. Sen sijaan murtovaras tai muu vieras tutkii vierasta ympäristöä aivan eri tavalla. Jokainen myös kirjoittaa näppäimistöllä eri tavalla. Esimerkiksi painallusten viiveet ovat jokaisella erilaisia. Ehkä käyttäjä kirjautuu yliopiston UNIX-palvelimelle yleensä vain lukeakseen sähköpostia, joten hyvin poikkeava kirjoitustyyli, näppäinviiveet ja krakkerien käyttämät järjestelmä-

komennot ovat todennäköisesti merkki palvelimen luvattomasta käytöstä. Tällä tavoin olisi mahdollista havaita myös *yhteyden kaappaus* (session hijacking).

Biometrinen tunnistusmenetelmä mittaa ensin biometrisen tiedon, josta se luo mallin. Malli kuvaa käyttäjältä mitattua ominaisuutta. Se tallennetaan tiettyyn paikkaan, joka voi olla esimerkiksi palvelimen kiintolevy. Kun käyttäjä halutaan myöhemmin tunnistaa, niin häneltä mitataan uudelleen sama biometrinen tieto, josta luodaan uusi malli. Uutta mallia verrataan alkuperäiseen. Jos ne täsmäävät, niin käyttäjä on tunnistettu [Liu and Silverman, 2001]. Usein teknisesti vaikeinta on toteuttaa mallin luonti eli biometrisen tiedon tulkinta.

Biometriset tunnistusmenetelmät ovat periaatteessa turvallisia tietyissä käyttötarkoituksissa, koska kehon ominaisuuksia ei voi yleensä varastaa, unohtaa tai väärentää [Liu and Silverman, 2001]. Harvemmin unohdamme käsiämme kaupan kassalle. Voidaan kuitenkin olettaa, että krakkeri pääsee ennemmin tai myöhemmin käsiksi tietokantaan. Jos sinne on tallennettu käyttäjien biometriset mallit sellaisenaan, niin hän voi niiden avulla murtautua myös kaikkiin muihin järjestelmiin, joihin käyttäjä on kirjautunut samalla biometrisellä mallilla. Ominaisuuksia ei voi siten pitää salaisuuksina, joten menetelmä ei yleensä sovellu verkon yli tapahtuvaan käyttäjätunnistukseen. Se sopii muun muassa käyttöjärjestelmään kirjautumiseen kannettavassa tietokoneessa, passintarkastukseen ja vastaavaan, jossa tunnistustapahtuma on mahdollista suorittaa enemmän tai vähemmän luotettavasti. Käyttäjän ominaisuuksia voidaan verrata salasanoihin, joita on vain muutamia. Olisi melkoisen turvatonta käyttää koko elämänsä aikana vain esimerkiksi kymmentä salasanaa. Tunnisteiden kopiointi ja väärentäminen onnistuvat todennäköisesti suurilta tiedustelupalveluilta. Esimerkiksi sormenjälkiä löytyy sieltä täältä, ja niiden kerääminen on varsin helppoa.

Hyökkäyksen kohteena olevat käyttäjät eivät voi vaihtaa biometrisiä ominaisuuksiaan salasanan tapaan. Täten toteutus olisi parhaimmillaan silloin, jos biometrinen malli olisi täysin erilainen jokaisessa palvelussa. Viranomaisten käyttämissä tietokannoissa täytyy säilyttää käyttäjien todelliset mallit, mutta muissa järjestelmissä tälle ei ole yleensä tarvetta. Yritysten ja ylläpitäjien motiiveista ei ole aina varmuutta, joten mallia ei voi pitää siksikään suurena salaisuutena. Tämän enempää en kertaa tässä biometrisiä tunnistusmenetelmiä yleisellä tasolla. Käsittelin aihetta seminaarityössäni [Santala, 2001]. Webistä löytyy runsaasti aiheeseen liittyviä uudempia yleiskatsauksia.

### 5.4.2. Tiedon salaus biometrisellä mallilla

Useita monimutkaisia virheenkorjaus- ja muita menetelmiä on ehdotettu käytettäväksi osana avainten tallentamista [Uludag *et al.*, 2004]. Tällöin peräkkäin mitatuista malleista yritetään karsia tavalla tai toisella pienet poikkeavuudet, jolloin ominaisuus saadaan parhaassa tapauksessa mitattua aina samanlaisena. Tällöin mallia voi käyttää salausavaimena. Esitän seuraavassa niistä muutamia.

Esimerkiksi Soutar ja muut [1998] ovat esittäneet ratkaisun sormenjälkiä varten, jossa salausavain k0 yhdistetään monimutkaisten funktioiden avulla sormesta otettuihin kuviin. Kirjautumisvaiheessa otetaan useita kuvia, joista muodostetaan suodatusfunktio H(u). Funktion virheensiedon ja käyttäjien erottelun tasapainoon voi vaikuttaa parametrin avulla. Korrelaatiofunktio c0(x) muodostetaan H(u):n ja kuvien avulla. Sen avulla saadaan tietoa sormenjälkikuvien samankaltaisuudesta. Salausavaimen k0 ja korrelaatiofunktion c0(x) avulla muodostetaan hakutaulu LT. Hstored(u) saadaan poistamalla tietyllä satunnaisuuteen perustuvalla tavalla informaatiota funktiosta H(u). Salaisella avaimella k0 salataan osa funktiosta Hstored(u). Muodostetusta salakielitekstistä otetaan hash-arvo, joka toimii tarkistussummana id0. Hakutaulu LT, muokattu suodatusfunktio Hstored(u) sekä tarkistussumma id0 tallennetaan kantaan.

Tarkistusvaiheessa luodaan uusien kuvien ja kannan Hstored(u)-arvon avulla uudelleen c1(x) eli korrelaatiofunktio. Sen ja hakutaulun LT avulla saadaan selville salainen avain k1. Tämän jälkeen osa funktiosta Hstored(u) salataan kuten aiemmin. Siitä saadaan hash-funktion kautta uusi tarkistussumma id1. Jos kannan tarkistussumma id0 ja muodostettu id1 täsmäävät, niin salausavain on oikea, jolloin myös käyttäjä on algoritmin kannalta oikea. Algoritmi ei kestä ongelmia kuvien keskinäisessä *kohdistuksessa* (alignment), joita biometrisissä kuvissa esiintyy. Keksijät eivät myöskään selvitä sitä, paljonko satunnaisuutta katoaa algoritmin funktioissa. Satunnaisuuden väheneminen helpottaa kryptoanalyysiä.

Davida ja muut [1998] ovat esittäneet ratkaisun, jossa käytetään tunnisteina iiristä IrisCode-muodossa. Kun malli luodaan tietokantaan, niin useasta kuvasta luodaan IrisCode-muotoinen malli enemmistöperiaatteella. Siinä mallin jokainen bitti saa arvon sen mukaan, mikä on malleissa yleisin bitti kyseisessä kohdassa. Virheiden määrä vähenee, mitä enemmän lukukertoja käytetään referenssimallin luomiseen. Tämä edellyttää tietenkin sitä, että lukijalaite toimii oikein ja ympäristö ei muutu.

Erityisesti allekirjoitusvaiheessa lukukertoja voi olla huomattavasti enemmän kuin käyttäjätunnistuksessa. Tosin kun kyse on iiriksestä, niin kamera voi

ottaa kuvia sarjassa. Mallista luodaan sen verran virheenkorjausbittejä, että siitä voidaan korjata kymmenen prosenttia virheistä. Käyttäjän nimen, lisätietojen, mallin ja virheenkorjauksen yhdistelmästä otetaan hash-arvo, joka allekirjoitetaan. Tietokantaan tallennetaan tämä arvo yhdessä virheenkorjausbittien sekä käyttäjän nimen sekä lisätietojen kanssa. Lisätiedoissa on esimerkiksi käyttöoikeuksiin liittyvää tietoa ja sertifikaatin myöntäneen tahon nimi.

Allekirjoitusta käytetään alkuperäisessä ratkaisussa siksi, että se on kehitetty älykortteja ja esimerkiksi magneettinauhoja varten. Esimerkiksi viranomainen suorittaa allekirjoituksen, jolloin käyttäjän henkilöllisyys voidaan myöhemmin varmentaa muualla. Käyttäjätunnistuksen aikana käyttäjän IrisCode-malleista luodaan yksi IrisCode-malli, joka korjataan tietokannan virheenkorjausbittien avulla. Käyttäjän nimi, lisätiedot, korjattu malli ja virheenkorjausbitit yhdistetään. Tuloksesta otetaan hash-arvo, joka mahdollisesti allekirjoitetaan. Jos lopputulos on sama kuin tietokantaan tallennettu, niin käyttäjä on tunnistettu onnistuneesti. Ratkaisua on kritisoitu siitä, että virheitä voi olla IrisCodessa enemmän kuin kymmenen prosenttia [Uludag *et al.*, 2004]. Lisäksi virheenkorjausbitit paljastavat mallista tietoa. Tämäkään ratkaisu ei kestä ongelmia kuvien keskinäisessä kohdistuksessa.

Monrose ja muut [1999] ovat kehittäneet ratkaisun, johon he ovat saaneet ideansa salasanojen salt-arvoista. Siinä perinteisten salasanojen turvallisuutta parannetaan sen perusteella, kuinka kukin käyttäjä käyttää näppäimistöä. Kullakin käyttäjällä on erilaiset viiveet näppäinpainallusten välillä ja painalluksien kestossa. Ratkaisu tuo salasanaan kuitenkin vain noin 15 bittiä satunnaisuutta [Uludag *et al.*, 2004]. Lisäksi jos hyökkääjän on mahdollista tavalla tai toisella mitata näppäinviiveitä, niin hänen on mahdollista muodostaa käyttäjän malli.

Kaikissa näissä ja muissa esitetyissä tavoissa on toistaiseksi ongelmia. Ongelmat johtuvat siitä, että saman käyttäjän mallissa on eroavuuksia eri lukukerroilla. Monissa tunnistusmenetelmissä tarvitaan monta lukukertaa, jotta saadaan keskimääräinen eli todennäköinen malli. Kuvien erilainen kohdistus eri lukukerroilla näyttää olevan yleinen ongelma.

Kuville voisi kehittää jonkinlaisen esisuodattimen, jolla kuvat asetettaisiin jollain standardilla tavalla koordinaatistoon, ja kohdistusvirheiden vaikutus minimoitaisiin. Tällaisen suodattimen ei ole kuitenkaan tarkoitus poistaa kohinaa tai muitakaan yksityiskohtia. Kuvan pyöritys oikeaan asentoon täytyisi tehdä ennen sitä toisella suodattimella. Suodatin täytyisi luoda erikseen jokaiselle biometriselle tunnistusmenetelmälle, jos siinä otetaan kantaa mallien yksityiskohtiin. Ongelmaa voisi miettiä pisteinä tai vektoreina kaksiulotteisessa koordinaatistossa. Säännöt voivat olla periaatteessa monimutkaisia ja jonkin

verran laskentaakin vaativia. Periaate olisi kuitenkin se, että funktiolle riittää syötteenä vain biometrisen mallin kuva. Tulosteena tulee kuva, joka on kohdistettu deterministisesti. Jos tällainen funktio on mahdollista toteuttaa, niin se korjaisi ongelmia monessa ehdotetussa ratkaisussa.

Tunnistinlaitteella saadaan esimerkiksi sormenjälki yleensä samankokoisena. Kun vertailtava biometrinen malli luodaan, niin kuvasta voitaisiin valita jonkin kriteerin avulla pienempi esimerkiksi neliskulmainen alue. Sormenjäljen reuna-aluetta ei kannata valita, koska se mahdollisesti puuttuu myöhempien lukukertojen kuvista. Seuraavaksi lasketaan esimerkiksi yhteen kaikkien rajatulla alueella sijaitsevien yksityiskohtapisteiden kyseisen alueen koordinaatiston x- ja y-koordinaatit tietyllä tavalla. Tämä tieto tallennetaan alueen koon kanssa kantaan. Kun käyttäjätunnistuksessa halutaan korjata kuvan kohdistus, niin etsitään hakualgoritmilla kohta, jossa summa on mahdollisimman lähellä alkuperäistä.

Hakuongelmaa voi ajatella siten, että kombinaatioita on enintään yhtä paljon kuin missä pienemmän koordinaatiston kulmapiste voi käydä. Näitä pisteitä voi olla enintään niin paljon kuin kuvapisteitä vähennettynä kaikki alueet, joissa osa-alue ei mahdu kokonaisuudessaan alkuperäiseen bittikarttakuvaan. Jotta mallista paljastuisi mahdollisimman vähän tietoa, niin summien tilalla voidaan käyttää jakojäännöstä. Lisäksi mallin satunnaisuutta kätkee se, että vain osa mallista käytetään summan luomiseen. Löydettyjä kohtia voi testata salauksen avaamisessa useilla vaihtoehdoilla paremmuusjärjestyksessä.

Summa voidaan muodostaa painottamalla kuvan sisältämää tietoa monin tavoin. Sormenjäljessä on erilaisia yksityiskohtia, joten pisteitä voidaan painottaa niiden tyypin mukaan. Vastaavasti sormenjäljen harjanteita voi ajatella vektoreina, jotka päättyvät viimeistään rajatun alueen reunoille. Harjanteet voisi periaatteessa muodostaa Bézier-käyristä vektorigrafiikkaohjelmien tapaan, jolloin kuva vektoroidaan.

Tämän luvun viimeistelemättömän version kirjoitin jo muutamia vuosia sitten. Sen jälkeen biometristen tunnisteiden käyttöä salausavaimena ovat monet miettineet esimerkiksi BioHashing-termin alla. Tätä ongelmaa ei ole vieläkään ratkaistu, vaikka kieltämättä yritystä on ollut. Joka tapauksessa käyttäjän ominaisuudet voidaan kopioida, joten sormenjäljet, verkkokalvo tai muut ominaisuudet eivät ole hyviä salausavaimia, vaikka niitä prosessoitaisiin miten hienosti tahansa ilman ylimääräistä salaista tietoa. Deterministisen funktion syöte kun pysyy koko ajan samana. Yhdistettynä turvalliseen ajoympäristöön kuten luvussa kuusi kuvaamaani laitteistotason salasanatietokantaan tämä voisi olla järkevä vaihtoehto, mutta silloinkin aidosti satunnainen perinteinen

salausavain on kryptologian kannalta satunnaisempi, yksinkertaisempi ja vikasietoisempi vaihtoehto kuin biometrinen malli.

Kryptoanalyysin kannalta biometrinen malli sisältää todennäköisesti turhan paljon ennustettavuutta verrattuna perinteiseen salausavaimeen. Esimerkiksi yli 99 % ihmisperimästä on hyvin pitkälle samanlaista kaikilla ihmisillä [Gleni and Petratos, 2004]. Lopusta DNA:stakin löytyy kuitenkin osia, jotka eroavat eri ihmisillä niin paljon, että niitä voidaan käyttää luotettavasti esimerkiksi rikosoikeudellisissa käyttötarkoituksissa. Biometrinen käyttäjätunnistus on vielä varsin uusi ja suorastaan kehittymätön tietojenkäsittelyn osa-alue verrattuna kryptologiaan, joten suhtaudun siihen nykyään hieman skeptisesti. Varsinkin tunnisteiden käyttö salausavaimina tai salasanoina ei vakuuta edellä mainituista syistä.

### 5.4.3. Palvelinkohtaiset mallit

Perinteisesti biometristen ominaisuuksien vertailu suoritetaan palvelimella biometrisen raakadatan tai tunnistuslaitteen luoman mallin avulla. Laite tuottaa käyttäjäkohtaisesti aina enemmän tai vähemmän saman mallin. Krakkeri voi näin päästä käsiksi malliin tai raakadataan joko asiakkaan omalla koneella, verkossa tai palvelimella. Ongelma voitaisiin ratkaista osittain suorittamalla jokin yksisuuntainen funktio jo tunnistuslaitteessa kullekin mallin arvolle tai koko mallille.

Funktion tuloksen täytyisi määräytyä kohdejärjestelmän mukaan siten, että kahden kohdejärjestelmän mallit korreloivat mahdollisimman vähän keskenään. Jos se valitaan kryptologian kannalta oikein, niin käyttäjän oikean mallin paljastuminen ei ole vakavaa muiden järjestelmien kannalta. Tällöin palvelinkohtainen malli voidaan aina vaihtaa. Biometriset mallit ovat käyttäjän tunnisteita eikä salaisuuksia [Schneier, 1999], koska niitä ei voi käytännössä vaihtaa ja niitä on helppo vakoilla. Biometristä mallia ei voi *mitätöidä* (revocate) kuten tavallisia epäsymmetrisiä salausavaimia tai salasanaa. Jos oletetaan, että tietyn käyttäjän malli saataisiin tunnistuslaitteesta aina täsmälleen samanlaisena, niin tällöin saataisiin palvelinkohtainen malli esimerkiksi seuraavasti hash-fuktiolla tai lohkosalaajalla:

hash (identifier + key)
tai
blockCipherEncrypt (identifier, key).

Plusmerkki tarkoittaa merkkijonojen *yhdistämistä* (concatenation) ja key aidosti satunnaista salausavainta, jonka pituus kannattaa suhteuttaa hash-

funktion tai lohkosalaajan turvatason mukaan. Esimerkiksi SHA-1-funktion lopputulos on 160-bittinen, joten turvallisuus paranee lisäämällä satunnaisuutta vähintään samaan pituuteen asti. Funktion arvon täytyy olla uusien tunnistuskertojen vuoksi deterministinen, joten avaimen lisäksi funktioon ei voi lisätä muuta satunnaisuutta. Salausavaimen satunnaisuus on siis erityisen tärkeää.

Tämä on kuitenkin epärealistinen tilanne, kuten kerroin edellisessä alakohdassa. Todellisuudessa käyttäjän malli eroaa hieman eri lukukertojen välillä [Uludag *et al.*, 2004]. Yksittäiset tavut tai arvot saattavat vaihdella eri lukukertojen välillä lukulaitteen epätarkkuuden, lian ja muiden ongelmien vuoksi. Kohdistusongelmien vuoksi hash-funktiota tai lohkosalaajia ei voi käyttää niiden toimintaperiaatteen vuoksi. Koska laadukkaat hash-funktiot ja lohkosalaajat noudattavat enemmän tai vähemmän Strict Avalanche Criteriaa (SAC) [Schneier, 1996], niin yhden bitin muutos muuttaa merkittävästi koko lopputulosta. Lukijalaitteita ja niiden algoritmeja voidaan yrittää kehittää siten, että malli olisi mahdollisimman usein täsmälleen samanlainen. Jos kohdistusongelmia ei olisi, niin funktioksi voisi valita esimerkiksi one-time padin [Schneier, 1996] seuraavasti:

```
oneTimePad(identifier, key)
eli
for(i=0; i<identifier.size; i++){
        ciphertext[i] = identifier[i] ^ key[i];
}
```

Kryptoanalyytikko voi ehkä päätellä jotain alkuperäisestä mallista eri lukukertojen tavujen eroavuuksista tehtyjen tilastojen perusteella, mutta koko mallin selvittäminen ei todennäköisesti onnistu, ja silloinkin hyöty olisi kyseenalaista, koska malli salataan eri avaimella kullekin palvelimelle. Vaikka salaus on tavukohtaista, niin se lisää silti merkittävästi virhealttiutta. Koska lukijalaitteen tai sovelluksen täytyy säilyttää salainen avain, niin se voisi myös säilyttää sen verran tietoa mallista, että erilaiset virheet voitaisiin havaita. Tarkistustieto ei saa paljastaa liikaa tietoa mallista.

Yksi vaihtoehto olisi käyttää perinteistä taulukkoon perustuvaa korvaussalaajaa. Se on turvaton verrattuna nykyisiin salausalgoritmeihin, mutta se sietää paremmin osaa virheistä. Tämä johtuu siitä, että kukin tavu ja arvo on korvattu erikseen eikä osana suurempaa lohkoa. Palvelinkohtaiset mallit ovat käytännössä mahdollisia sitten, kun on ratkaistu edellisessä alakohdassa mainitut ongelmat. Sen jälkeen voidaan käyttää tavallista lohkosalaajaa.

### 5.4.4. Mallin vertailu lukulaitteessa tai palvelimella

Malli voidaan periaatteessa tallentaa myös sellaisenaan *peukalointia kestävälle* (tamper resistant) älykortille. Tällöin niiden vertailu tapahtuu perinteisesti vertailemalla ja siten varsin luotettavasti. Kortin yksikäsitteistä mallia kun voidaan tällöin käyttää palvelinkohtaisen mallin luomiseen. Sen sijaan on kuitenkin järkevämpää käyttää tunnisteena kortin rautatason satunnaislukugeneraattorin generoimaa pitkää salasanaa tai salausavainta. Yleensä ei ole lopulta mitään syytä, miksi käyttäjän biometrinen malli täytyisi lähettää asiakaskoneen ulkopuolelle. Toisaalta koska mallit eivät ole salaisuuksia, niin ne voidaan toimittaa palvelimelle jopa alkuperäisessä muodossa, jos tietoliikenneyhteys salataan. Käyttäjä toimittaa palvelimelle lisäksi salausavaimen. Avaimella palvelin avaa tiedoston, joka sisältää käyttäjän biometrisen mallin. Palvelin vertaa mallia käyttäjän toimittamaan malliin. Tässä ratkaisussa mallien vertailu on yhtä yksinkertaista kuin silloin, kun ei käytetä lainkaan kryptologiaa.

Lukijalaitteeseen voidaan lisätä pieni näppäimistö sekä viivakoodilukija palvelimen julkisen avaimen tunnisteen lukua varten. Kaksiulotteinen viivakoodi voidaan lukea laitteelle paperilta tai näytöltä. Vaihtoehtoisesti voidaan käyttää syöttämiseen esimerkiksi laitteen näppäimistöä. Yksi käytännöllinen tapa on käyttää luotetun tahon eli CA:ta (Certification Authority) luomaa sertifikaattia palvelimen tunnistamiseen. Kerralla voidaan kirjautua vain yhteen palvelimeen, ja tunnistuslaitteessa on näyttö, josta näkee palvelimen osoitteen ja nimen. Kun yhteys on muodostettu palvelimen ja kortin välille, niin kortti luovuttaa tietokoneelle erillisen satunnaisen sessioavaimen, jonka avulla luodaan varsinainen yhteys tietokoneen ja palvelimen välille. Näin palvelimen ja kortin välistä liikennettä ei voida tutkia jälkikäteen, vaikka käyttäjän kone olisi krakkerin hallussa.

Suurin osa prosessoinnista kannattaa suorittaa älykortissa. Näin korttia voidaan käyttää useassa paikassa. Kyseessä on vahva käyttäjätunnistus, koska käyttäjällä täytyy olla kortti sekä biometrinen tunniste. Lisäturvaa saadaan vielä PIN-koodilla. Tämän tyyppinen ratkaisu voisi toimia sähköisenä henkilökorttina. Käyttöjärjestelmää ja esimerkiksi tiedostojen salausta varten voisi tehdä erillisiä protokollia. Monipuolisuus kylläkin lisää todennäköisyyksiä tietoturva-aukoille. Kunkin sovelluksen salausavain tai salasana annetaan, jos ohjelman sertifikaatti on kunnossa.

Lukijalaitteen näytöstä pitäisi käydä ilmi, mistä sovelluksesta on kyse, kun käyttäjää tunnistetaan. Käyttäjähän on yleensä tällöin muutenkin tietoinen tunnistustapahtumasta, joten satunnaiseen aikaan tulevat tunnistuspyynnöt herättävät ainakin osassa käyttäjiä epäilyksiä. Psykologisia ongelmia kuitenkin

löytyy. Jos laitetta käytetään käyttöjärjestelmään kirjautumiseen, niin käyttäjä ei enää jaksa katsoa joka kerta sertifikaatin tietoja. Tämän voikin korjata siten, että pelkkä älykortin läsnäolo lukijalaitteessa riittää kirjautumaan järjestelmään. Kortille voidaankin luoda myös sellaisia salausavaimia, jotka luovutetaan digitaalista allekirjoitusta vastaan.

Jos laitteen fyysisiä suojauksia yritetään murtaa, niin luonnollisesti muistin sisältö ylikirjoitetaan akkuvarmennuksella. Sen sijaan jos yrityskertoja kertyy liikaa joko biometrisen mallin tai PIN-koodin kanssa, niin laite voidaan lukita PUK-koodin tyyppisellä koodilla matkapuhelimen tapaan. Vaihtoehtoisesti avaimet sekä biometrinen malli voidaan salata viranomaisen tai muun luotettavan tahon julkisella korttikohtaisella avaimella. Korttiin on luotu logiikka tällaista palautustilannetta varten. Kun kortti halutaan palauttaa entiseen tilaansa, niin ensiksi luotettu taho tarkistaa käyttäjän henkilöllisyyden rekisterin valokuvan ja henkilötietojen perusteella. Kortti tutkitaan myös fyysisten muutosten varalta. Käyttäjältä tarkistetaan myös kortin käyttämä biometrinen malli.

Tämän jälkeen kortti purkaa salauksen luotettavan tahon salaisella avaimella, ja kortille syötetään viranomaisen uusi julkinen avain ja luodaan uusi biometrinen malli. Esimerkiksi Smith ja Weingart ovat käsitelleet kirjoituksessaan [1998] tamper resistant -tekniikoita ja avainparien luontia kortilla. Monilla organisaatioilla on resursseja murtaa kortit moninaisilla tavoilla [Anderson and Kuhn, 1996].

### 5.4.5. Biometrinen tunnistus ja salasanatietokanta

Muun muassa Microsoft on tuonut markkinoille näppäimistön, jossa on sormenjälkitunnistin. Sillä voi kirjautua Windowsiin ja web-palveluihin. Web-palveluiden salasanat tallennetaan kiintolevylle. Verkkopalveluihin kirjaudutaan ensimmäisen käyttökerran jälkeen asettamalla sormi lukijalle. Kiintolevylle tallentamisessa on se ongelma, että todennäköisesti joku krakkeriryhmä murtaa tällaisen ratkaisun suojaukset melko nopeasti. Jos tällaisesta yksittäisestä ratkaisusta tulisi yleinen, niin todennäköisesti joku kirjoittaa esimerkiksi verkkomadon, joka kopioi käyttäjän salasanat.

Vaikka salasanatietokanta olisi salattu, niin salausavain sijaitsee todennäköisesti tietokoneella. Koska valmistaja ei suosittele ratkaisun käyttöä kriittisiin salasanoihin, niin kiintolevylle tallennuksessa tai muussa toteutuksessa on varmasti selviä kompromisseja tietoturvan kannalta. On toki mahdollista, että lukijalaite on tavallaan kaksisuuntainen. Tällöin salausavain ja biometrinen malli voidaan pitää lukijalaitteen omassa muistissa. Kun käyttäjä haluaa käyttää laitetta uudessa web-palvelussa, niin salasana lähetetään laitteelle, joka

salaa sen. Salattu tieto lähetetään takaisin tietokoneelle, joka tallentaa sen kiintolevylleen.

Ehkä Microsoft on toteuttanut ratkaisunsa juuri näin. ROM-piirille tallennetulla 128-bittisellä salausavaimella pärjää pitkälle, jos avain on aidosti satunnainen. Koska salausavain on eri jokaisessa lukijalaitteessa, niin salasanat olisivat todennäköisesti suojassa hakkereilta ja madoilta salatussa muodossa. Kriittinen kohta on se, kun käyttäjä kirjautuu johonkin palveluun. Tällöinhän salasana tulee selväkielisenä lukijalaitteelta. Salasanaa käsittelevä ohjelmakoodi kuten käyttöjärjestelmän näppäimistöajuri sekä web-selain ovat ongelmalliset kohdat tietoturvan kannalta.

On toki mahdollista toteuttaa ratkaisu, jossa luodaan salattu yhteys lukijalaitteen ja palvelimen välille. Lukijalaite palauttaisi käyttäjätunnistuksen päätteeksi erillisen sessioavaimen tai muun tiedon web-selaimelle, jotta palvelua voidaan käyttää käyttäjätunnistuksen onnistuessa. Koska ratkaisu on hyvin samanlainen kuin edellisessä alakohdassa, niin voitaisiin muodostaa yksi standardi rajapinta, jota on mahdollista käyttää hyvin erilaisissa käyttötarkoituksissa. Kun salasanat sijaitsevat kiintolevyllä, niin haittaohjelmien on mahdollista tuhota ne. Salasanat olisivat paremmin suojassa, jos ne tallennetaan esimerkiksi lukijalaitteelle. Laitteen rikkoutuminen on tietenkin aina yksi riskeistä.

Myös man in the middle -hyökkäykset ovat mahdollisia, jos palvelimella ei käytetä sertifikaatteja tai julkisia avaimia. Nekään eivät tietenkään takaa turvallisuutta. Hyvänä esimerkkinä palvelimen avainparin ennenaikaisesta vanhenemisesta toimii tietomurto UNIX-palvelimelle, jonka jälkeen esimerkiksi palvelimen SSH-avaimen joutuu vaihtamaan uuteen. Taitava krakkeri kätkee jälkiään tai jopa luo valejälkiä, jotta ylläpito ei huomaisi tietomurron vakavuutta. Hän on voinut kopioida muun muassa palvelimen salaisen avaimen, jonka avulla hän voi luoda aidolta näyttäviä valepalvelimia.

Jos biometristä mallia halutaan välttämättä käyttää esimerkiksi salasanatietokannan salauksessa, niin salattavassa selväkielitekstissä olisi hyvä olla vain mahdollisimman satunnaista binääridataa. Näin kryptoanalyytikko ei tiedä, koska hän on löytänyt oikean mallin eli salausavaimen. Samaa periaatetta voidaan käyttää silloin, kun käytetään salasanaa. Tämä täytyy kuitenkin ottaa huomioon ohjelman käytettävyydessä. Ohjelma ei voi tällöin välttämättä tietää, onko salasana oikea.

UBS-muistia voidaan käyttää edullisena salasanatietokannan säilytyspaikkana. Se voi olla suurimman osan ajasta irti koneesta, mikä vaikeuttaa krakkerien työtä. Vielä turvallisempi vaihtoehto on lisätä USB-muistiin biometrinen lukijalaite siten, että tietokoneesta ei ole pääsyä laitteen toteuttamaan

salasanatietokantaan. Tällöin avaimet sekä käyttäjän malli ovat ehkä osittain suojassa krakkereilta. Vaihtoehtoisesti esimerkiksi SIM-kortti ja erilaiset älykortit soveltuvat myös tähän käyttötarkoitukseen.

Kun sormi painetaan kovaa alustaa vastaan, niin sen pinta mahdollisesti venyy suuntaan tai toiseen. Tästä tuli mieleeni sellainen lukijalaite, jonka pinta venyisi painettaessa. Pinnan täytyisi olla juuri sopivan taipuisa, jotta sormen pinta vääristyisi mahdollisimman vähän. Tätä voi havainnoida tavallisella tyynyllä. Ehkä lukijalaitteen tarkkuus kohenisi tällä tavoin. Laitteiden ja biometristä raakadataa käsittelevien algoritmien kehittäminen on tärkeää, jotta laitteen muodostama malli olisi aina mahdollisimman samanlainen.

### 5.4.6. Anonyymi biometrinen käyttäjäseuranta

Kun keskustelin jokin aika sitten kirjoitelman ohjaajan professori Jyrki Nummenmaan kanssa palvelinkohtaisista biometrisistä malleista, niin hän esitti idean anonyymistä käyttäjätunnistuksesta. Mietin tuolloin hetken, että mitä hyötyä siitä voisi olla, jos käyttäjät täytyy kuitenkin tunnistaa. Jaoin tämän ongelman kahteen tilanteeseen. Käsittelen tässä alakohdassa käyttäjien anonyymiä seurantaa ja seuraavassa anonyymiä tunnistusta.

Ihmiset eivät useinkaan pidä siitä, että heidän liikkumistaan seurataan tarkasti. Joillakin työpaikoilla tämä menettely on työnantajan asettama vaatimus, ja kamerat valvovat katuja ja kauppoja. Kamerat kuvaavat ideaalimaailmassa videota vain rikosten selvittämistä varten. Vapaa-aikana ihmiset eivät yleensä halua, että heidän sijaintiaan voidaan seurata, vaikka se onkin teknisesti todella helppoa viranomaisten ja tiedustelupalveluiden puolesta, jos kantaa mukanaan matkapuhelinta.

On teoriassa mahdollista rakentaa anonyymi järjestelmä esimerkiksi virastoihin, tulliin, museoihin, taidenäyttelyihin. Siinä viranomaisilla on mahdollisuus selvittää rikostilanteessa ainakin jossain määrin, ketä kohteessa on kulloinkin käynyt. Käyttäjän malli tai laitteelta saatava raakadata salataan viranomaisen julkisella avaimella täytebittien (padding) kera ja tallennetaan salattuna tietokantaan. Kohteen ylläpito ei saa näin selville alkuperäisiä tunnisteita tai malleja. Laitteen täytyy olla mieluiten viranomaisten ylläpitämä laite eli perinteinen *musta laatikko* (black box).

Toteutuksen turvallisuutta on tietenkin mahdoton taata. Laitteistotason suojaus on oma tieteenalansa, joka kehittyy yhdessä kryptologian, ohjelmistojen ja erityisesti laitteistotason hyökkäysten kehittyessä [Skorobogatov, 2005] [Anderson and Kuhn, 1997] [Anderson and Kuhn, 1996]. Esimerkiksi älykorttien suojaamiseen kehitellään uusia suojakerroksia ja menetelmiä, jotta pääsy esimerkiksi salausavaimiin olisi yhä vaikeampaa.

Ongelman vaikeutta kuvaa se, että edes ydinaseet eivät ole täydellisesti suojassa manipuloinnilta, vaikka niissä lienee kehittyneimmät suojaukset [Anderson and Kuhn, 1996]. Suojausten tarkka kuvaus on luonnollisesti erittäin salaista tietoa. Jos asetta yritetään peukaloida, niin se tuhoaa itsensä sen mukaan, mikä malli on kyseessä. Yksi tapa on muuttaa fissiopolttoaineen kuten plutoniumpallon (pit) muoto epäsuotuisaksi, jolloin kriittistä massaa ei pääse enää käytännössä muodostumaan [Anderson and Kuhn, 1996] [Wikipedia, 2008b]. Toinen tapa on räjäyttää ase epäsymmetrisesti siten, että kriittistä massaa ei muodostu, jolloin enintään välitön lähiympäristö tuhoutuu ja saastuu polttoaineesta. Käyttökoodit eli salasanat tuhotaan. Myös kaikki kriittiset sähköiset osat kuten fissioreaktion käynnistävät neutronigeneraattorit tuhotaan esimerkiksi suunnatulla panoksella [Anderson and Kuhn, 1996].

*Peukalointia kestävät* (tamper resistant) älykortit ja prosessorit yrittävät hieman samoilla periaatteilla estää kortin avaamisen pienessä mittakaavassa. Jos murtoyritys havaitaan, niin esimerkiksi salausavain ylikirjoitetaan useita kertoja, jonka jälkeen muistipiiristä katkaistaan virta. Kaupallisissa käyttökohteissa keinot tiedon ja piirilogiikan tuhoamiseen ovat tietenkin sotilaskäyttöä rajatummat. Piiri ei pysy kauan markkinoilla, jos se räjähtää tai syttyy palamaan ongelmatilanteissa. Kehittyneitä kaupallisia yleiskäyttöön sopivia prosessoreja kuitenkin löytyy. Eräässä toteutuksessa prosessori mittaa muun muassa ionisoivan säteilyn määrää, käyttöjännitettä, kellosignaalia ja lämpötilaa [Smith and Weingart, 1998]. Ulkokuori on taipuva monikerroksinen johdinviidakko, jossa sähköiset muutokset havaitaan välittömästi. Tuotteen kuljetus ja varastointi vaativat erityisjärjestelyjä. Jos lämpötila nousee liian suureksi tai laskee liian pieneksi, niin prosessori ylikirjoittaa muistinsa sisällön. Toteutukset muuttuvat yhä monimutkaisemmiksi hyökkäystapojen kehittyessä.

Palataan takaisin anonyymiin käyttäjäseurantaan. Tunnistustapahtuman kellonaika voidaan liittää tunnisteen yhteyteen, jolloin tiedetään, mitä salattuja tunnisteita täytyy toimittaa poliisille rikoksen sattuessa. Jos tarkka kellonaika tallennetaan vain salattuna ja pelkkä päivämäärä selväkielisenä, niin myös kellonajoista on mahdollista tehdä osittain ”anonyymejä”, jos tapahtumia ei tallenneta yksittäisiin tiedostoihin. Yksittäisiäkin tiedostoja voidaan käyttää, jos tiedostojen kellonajat vaihdetaan esimerkiksi päiväkohtaiseksi vakioajaksi tallennuksen jälkeen. Tunnisteen luku ja salaus suoritetaan laitteessa, joka tarkistaa, että salaukseen käytettävä avain on viranomaisen allekirjoittama. Näin kukaan ei voi vaihtaa avainta tarkoituksella toiseksi. Tapahtumat voidaan tallentaa laitteen kiintolevylle. Vanhat tunnisteet tuhotaan käyttökohteen mukaan tietyn ajan kuluttua.

Jokaisessa laitteessa voi olla käytössä erillinen viranomaisen luoma julkinen avain. Laitearkkitehtuuriin olisi mahdollista rakentaa hierarkkinen allekirjoitusmenettely. Tällä avaimella allekirjoitetaan kaikki viranomaisen luomat julkiset avaimet. Esimerkiksi valaistusta on vaikea kalibroida täsmälleen samanlaiseksi kaikissa kohteissa, joten tunnisteissa voi olla eroja. Valaistus vaikuttaa esimerkiksi iiriksen tunnistukseen [Dunker, 2003]. Mallin sijaan kannattaa salata esimerkiksi kuva sormenjäljestä, iiriksestä tai muusta ominaisuudesta. Tällöin laitteiden ei tarvitse olla identtisiä ja kalibroituja keskenään, ja eri kohteissa voitaisiin käyttää eri tunnistusmenetelmiä.

Sormenjäljet ovat kuitenkin hygienian kannalta huono vaihtoehto esimerkiksi silloin, kun influenssa tai muu tauti on aktiivisena. Esimerkiksi ovenrivat ja pankkiautomaattien näppäimistöt eivät ole kovin hygieenisiä. Harva käyttää hansikkaita pankkiautomaatilla. Iiris voidaan tunnistaa tällä hetkellä silmälasienkin läpi yli puolen metrin päästä. Iiristunnistus olisi luotettava, vaikeasti väärennettävä ja hygieeninen [Dunker, 2003]. Myös kasvojen lämpökuva on hygieeninen vaihtoehto. Tällöin täytyy tallentaa lisäksi tavallinen kuva kasvoista, jotta ihmisen on helpompi vertailla kuvia. Näin perinteinen naamiointi ei riittäisi salaamaan rikollisen henkilöllisyyttä.

Tunnisteet täytyy salata sellaisessa muodossa, että niitä voidaan käyttää rikosoikeudellisissa käyttötarkoituksissa. Sellaisia vaatimuksia on määritelty esimerkiksi sormenjälkiä varten [Interpol, 2004] [Interpol, 2000]. Dokumenteissa otetaan kantaa muun muassa kuvanlaatuun, joten kuvamuotoiset sormenjälki- tai muut tunnisteet kannattaa tallentaa sellaisena, että häviöllinen pakkaus ei heikennä merkittävästi niiden laatua rikosoikeudelliselta kannalta. Erilaiset standardit ovat tässä hyödyllisiä. Täysin erilaisetkin rikokset eri kohteissa voidaan näin yhdistää samaan henkilöön, vaikka henkilö ei olisikaan rikosrekisterissä. Ehkä rikollinen jää joskus kiinni muusta rikoksesta tai teettää biometrisen passin, jolloin hänet voidaan yhdistää aiempiinkin rikoksiin. Ratkaisu on kuitenkin yksityisyyden suojan kannalta epäilyttävä ja kiusallinen. Kameravalvonnan pitäisi pääsääntöisesti riittää ja sitäkin käytetään liikaa. Se on muuten yksi anonyymin biometrisen käyttäjätunnistuksen muoto, vaikka kuvaa ei yleensä salatakaan.

### 5.4.7. Anonyymi biometrinen henkilöllisyyden tarkistus

Jos käyttäjän täytyy kuulua tiettyyn joukkoon, mutta henkilöllisyydellä ei ole muuten merkitystä, niin tarvitaan anonyymiä käyttäjätunnistusta. Tällaisia käyttökohteita voisivat olla esimerkiksi paikallisliikenteen bussit, junat, huvipuistot ja vastaavat. Käyttäjästä tehdään ennen palvelun käyttöä biometrinen malli, jolle luodaan määräajaksi käyttöoikeus. Käyttöoikeudessa on määritelty

sen voimassaolo sekä biometrinen malli. Tunnistustilanteessa mallien joukosta etsitään se malli, joka vastaa kyseistä käyttäjää. Jos mallia ei löydy, niin palvelun käyttöä ei sallita. Useimmissa yrityksissä ja organisaatioissa täytyy kuitenkin tietää ihmisten henkilöllisyys väärinkäytösten jälkeen.

Esimerkiksi huvipuistossa ja junissa riittää, että asiakkaalla on lupa käyttää palvelua määräajan. Käyttäjiä on kerralla vain sen verran, että suurempia ongelmia ei pitäisi tulla. Julkisessa liikenteessä täytyisi kuitenkin varautua tunnistusongelmiin, jotta kenenkään ei tarvitse jäädä järjestelmän teknisten ongelmien vuoksi laiturille. Lisäksi käyttäjän tunnistaminen oikeaksi ei ole varmaa, joten toisen kustannuksella matkustaminen on mahdollista. Tämän vuoksi ei voida toistaiseksi luopua bussien matkakorteista tai muusta vastaavasta tekniikasta.

Biometrisen ratkaisun voisi yhdistää kioskista ostettaviin kortteihin. Kortin tarkoitus on säilyttää esimerkiksi salausavaimia, joiden avulla voidaan kirjautua tietojärjestelmiin. Riskinä on tietysti se, että kortti kopioidaan, varastetaan tai vaihdetaan käyttäjän tietämättä. Jos jollakin taholla on taitoa kiertää kortin suojaukset, niin sillä on todennäköisesti myös taitoa luoda käyttäjän malli vakoilemalla häntä tai keräämällä sormenjälkiä. Huomautan tässä vielä kerran, että biometrisiä malleja ei saa pitää tärkeinä salaisuuksina.

### 5.4.8. CAPTCHA

Usein riittää, että käyttäjä tunnistetaan ihmiseksi. Tietokoneella voidaan generoida ongelmia, joiden ratkaiseminen vaatii enemmän tai vähemmän ihmisen älykkyyttä. Ongelmissa voidaan käyttää hyväksi muun muassa nykyisten hahmontunnistusalgoritmien heikkouksia. Esimerkiksi ilmaisia sähköpostiosoitteita tarjoavat palvelut käyttävät näitä CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart)-testejä [Ahn *et al.*, 2004]. Toinen nimitys näille toteutuksille on *käänteiset Turingin testit* (Reverse Turing Tests, RTTs) [Naor, 1996]. Hahmontunnistusalgoritmeilla on esimerkiksi vaikeuksia tunnistaa kirjaimia kuvasta, jos kirjainten muotoa vääristetään, asetetaan päällekkäin ja lisätään kohinaa. Testien käyttöönotossa täytyisi ottaa huomioon myös käyttäjät, joilla on ongelmia esimerkiksi näön tai hahmottamiskyvyn kanssa. Siksi ääni- ja tekstimuotoisia ongelmia kannattaa käyttää vaihtoehtoina. Tällaisia hahmotustehtäviä voisi periaatteessa käyttää sähköpostin yhteydessä esimerkiksi siten, että käyttäjälle lähetetään kuvamuodossa ongelma, jonka hänen täytyy ratkaista ennen kuin vastaanottajan palvelin hyväksyy viestin.

Pinkas ja Sander [2002] ovat esittäneet parannuksen myös perinteiseen salasanatunnistukseen, jossa käyttäjien heikot salasanat ovat suuri riesa. Joissain

sovelluksissa kuten verkkohuutokaupassa tunnusten lukitseminen tai viiveen lisääminen epäonnistuneen käyttäjätunnistuksen jälkeen mahdollistaa esimerkiksi sellaisen hyökkäyksen, jossa huutaja tekee edullisen tarjouksen ja lukitsee kilpailijoiden tunnuksia. Lisäksi sanakirjahyökkäys suureen joukkoon käyttäjiä todennäköisesti onnistuu jollekin käyttäjälle. Pinkasin ja Sanderin [2002] ratkaisu toimii siten, että palvelin tunnistaa asiakkaan määräaikaisen *keksin* (cookie), IP-osoitteen tai muun tunnisteen avulla.

Jos cookie sisältää palvelimen luoman allekirjoituksen tai muun tunnisteen kyseiselle käyttäjätunnukselle, niin käyttäjä ei joudu suorittamaan testiä. Muuten testi täytyy suorittaa, jos salasana on oikein. Jos salasana on väärin, niin se täytyy suorittaa tietyllä todennäköisyydellä. Artikkelissa ehdotetaan yhdeksi todennäköisyydeksi 0.05 eli viisi prosenttia. Jos krakkeri syöttää esimerkiksi tuhat väärää salasanaa, niin hän joutuu suorittamaan viisikymmentä testiä. Turvallisuutta voidaan parantaa suurentamalla prosenttilukua. Cookien allekirjoituksen täytyy olla kryptologian kannalta riittävän turvallinen. Jotta keksin tarkistaminen ei vie liikaa resursseja, niin se kannattaa tallentaa tietokantaan. Keksi voi olla tällöin esimerkiksi satunnainen merkkijono. Lisäksi sillä voi olla voimassaoloaika.

Koska käyttäjät käyttävät usein melko rajallista määrää tietokoneita, niin tämä ratkaisu heikentää palvelun käytettävyyttä usein melko vähän. Käytettävyys heikkenee esimerkiksi silloin, kun selaimen cookiet poistetaan aina automaattisesti, kun selainohjelma suljetaan. Cookie asetetaan vain silloin, kun käyttäjä on kirjautunut järjestelmään onnistuneesti. Näin tapahtuu yleensä silloin, kun käyttäjä kirjautuu tietyllä tietokoneella ensimmäisen kerran tai cookie on vanhentunut. Pinkas ja Sander esittävät myös muunnelman, missä testiin täytyy vastata samalla sivulla kuin missä syötetään käyttäjätunnus ja salasana.

Palvelinestohyökkäyksetkin voitaisiin ottaa huomioon siten, että varastoidaan suuri määrä uusia testejä. Testejä luodaan erityiseen välimuistiin silloin, kun palvelimella ei ole merkittävästi muuta kuormaa. Kun välimuistista otetaan käyttäjälle uusi testi, niin se poistetaan samalla välimuistista. Välimuistille voidaan asettaa raja-arvo, jonka alittuessa välimuistista ei enää poisteta testejä, vaan valitaan testi satunnaisesti. Ala-arvon alittuessa lähetetään sähköposti- tai muun tyyppinen hälytysviesti ylläpidolle. Erityisesti sähköpostiviestejä voidaan lähettää vaikka tunnin välein niin kauan, että tilanne on normaali. Satunnaisvalintaa käytetään niin kauan, että testejä on tietyn lukumäärän verran raja-arvoa enemmän. Tällöin lähetetään jälleen ylläpidolle viesti, jossa kerrotaan, että tilanne on palannut normaaliksi. Näin testit eivät lopu palvelunestohyökkäyksen aikana, vaikka hyökkäys kestäisi päiviä. Esimerkiksi satojentu-

hansien testien ratkaiseminen pienen ihmisjoukon voimalla ei ole yleensä vaivan arvoista. Välimuistin koko voidaan tietysti määritellä tarvittavan turvatason mukaan.

Eräs toinen ratkaisu käyttäjätunnistuksen suojaamiseen on sellainen, jossa vaaditaan runsaasti laskentaa [Dword and Naor, 1992]. Jos käyttäjän kone joutuu laskemaan ratkaisua tiettyyn ongelmaan esimerkiksi viisi sekuntia, niin se vaikeuttaa hyökkäyksiä. Tietokoneiden laskentatehon kasvu täytyy kuitenkin huomioida sekä se, että hyökkääjällä on mahdollisesti huomattavasti enemmän laskentatehoa kuin tavallisella käyttäjällä. Joka tapauksessa hyökkäys vaikeutuu vain jonkin verran. Lisäksi käytettävyyden vuoksi laskennallinen ongelma ei saa olla liian raskas, koska käyttäjä saattaa käyttää vaihtelevasti hyvin eri tehoisia tietokoneita.

Protokollassa voisi käyttää sellaista logiikkaa, joka ottaa huomioon viiveet käyttäjän aiemmista onnistuneista kirjautumiskerroista. Tällöin laskentatehtävät vaikeutuvat lähes välittömästi myös silloin, kun krakkerilla on paljon laskentakapasiteettia. Vaikeimmalle laskentatehtävälle täytyy kuitenkin määritellä jokin raja, jotta oikea käyttäjä ei joudu palvelunestohyökkäyksen uhriksi. Tällaiset ratkaisut vaativat myös toteutuksen asiakkaan päässä [Pinkas and Sander, 2002], joka voidaan toki toteuttaa esimerkiksi Javascriptillä. CAPTCHA-toteutukset ovat käytännössä turvallisempia.

Protokolla on heterogeenisempänä turvallisempi, jos itsenäisiä CAPTCHA-toteutusten valmistajia on runsaasti. Jos vain yksi ohjelmistovalmistaja tarjoaisi pysyvän toteutuksen, niin kohta joku kehittäisi ohjelman, joka tunnistaa kuviot ja ratkaisee ongelman riittävän usein. Esimerkiksi Mori ja Malik [2008] ovat kehittäneet ratkaisun, jossa Yahoon CAPTCHA-toteutuksen yksittäinen ongelma murtuu 92 prosentin todennäköisyydellä.

Kun katsoin muutamaa sanoihin perustuvaa ongelmaa, niin huomasin niissä muutaman heikkouden. Värillisenkin taustan ja kirjainten välinen kontrasti on riittävä siihen, että kirjainten reunat voidaan tunnistaa. Kuva voidaan siis muuttaa vektorimuotoon. Usein riittää tunnistaa kirjaimesta vain sen yläosa, jotta saadaan jo todennäköinen kirjain. Erityisesti sanakirjan sanoja käyttävät ratkaisut murtuvat helpommin tämän vuoksi. Esimerkiksi Yahoon toteutuksessa kirjainten välinen etäisyys on aina melko sama, ja kontrasti on erittäin suuri mustavalkoisuuden vuoksi.

Sanoihin ja kirjaimiin perustuvat ratkaisut ovat ehkä käyttökelpoisimpia kansainvälisessä käytössä. Erityisesti lausutut kielen sanat ja asioiden nimeäminen aiheuttavat liikaa ongelmia käyttäjille, jotka eivät osaa sujuvasti esimerkiksi englantia. Esineiden ja asioiden tunnistaminen valokuvista olisi muuten todella mainio ratkaisu, mutta käyttäjän täytyy tuntea kuvan kohde sillä kie-

lellä tai kielillä, joita järjestelmän ylläpitäjä tukee. Kuviin voidaan lisätä esimerkiksi kohinaa [Ahn *et al.*, 2004].

Mietin tähän sellaisen ratkaisun, jossa satunnaisesti valituista kuvista valitaan esimerkiksi kaksi tai kolme. Todellinen lukumäärä täytyisi testata käytännön testeissä, jotta voidaan havaita se kuvien ja tehtävien muutosten määrä, jolla kuvista saa useimmiten vielä jotain selvää. Kuvat laitetaan *kerroksiin* (layer), joista kaksi päällimmäistä ovat osittain ja hieman satunnaisesti läpinäkyviä siten, että alimmaisimmastakin kuvasta ihminen onnistuu useimmiten hahmottamaan tarpeeksi. Läpinäkyvyys voi vielä vaihdella kuvan eri kohdissa.

Kutakin kuvaa pyöritetään satunnaisesti keskipisteensä ympäri. Sen jälkeen sitä siirretään X- ja Y-koordinaatistossa satunnaisesti. Kuvia voidaan myös skaalata hieman. Tämän jälkeen kuvan värit laitetaan sekaisin satunnaisesti valitulla värisuodattimella. Yksinkertainen suodatin voi esimerkiksi jättää pois RGB-väriavaruuden kaksi värikomponenttia. Tällöin kuvasta saadaan täysin puna-, vihreä- tai sinisävyinen. Näiden äärivaihtoehtojen väliltä löytyy täysvärisessä kuvassa miljoonia vaihtoehtoja. Kuvasta voidaan tehdä myös mustavalkoinen. Vaihtoehtoihin voidaan ottaa mukaan psykedeelisempiäkin vaihtoehtoja.

Värisuodatuksen jälkeen lisätään vielä kohinaa kohinasuodattimilla. Tällainen voi olla perinteistä digikuvien kohinaa. Lisäksi voidaan vääristää kuvan geometriaa satunnaisesti vaihdellen pitkin kuvaa. Siihen voidaan lisätä esimerkiksi aaltoja, joiden suunta on satunnainen, ja niiden aallonpituus vaihtelee satunnaisesti jokaisessa aallossa. Kuvaa voidaan sotkea myös siten, että valitaan yksi tai useampia satunnainen piste, joista aaltorintamat lähtevät. Muita vastaavia kuvankäsittelyyn liittyviä tekniikoita ovat esimerkiksi mosaiikkikuviot. Jokaiselle kuvalle voidaan valita oma suodatin. Perinteistä kuvakohinaa kannattaa todennäköisesti lisätä aina, koska sitä on vaikea poistaa kuvasta. Jos alkuperäisiä kuvia on esimerkiksi satatuhatta, niin kombinaatioita voi muodostua esimerkiksi kolmella päällekkäisellä kuvalla seuraavasti:

100000 * 99999 * 99998 (kuvien valinta)
* 359 ^ 3 (kierto akseliensa ympäri)
* 1000 ^ 2 (läpinäkyvyyden vaihtelujen kombinaatiot)
* (100 ^ 2) ^ 2 (siirto X- ja Y-akselien mukaisesti)
* 100 ^ 2 (skaalaus)
* 1000000 ^ 3 (värisuodattimet)
* 10000000 ^ 3 (kohinasuodattimet)
= 4.6 * 10 ^ 79.

Tämä takaa jo ainakin sen, että kukaan ei ala murtaa ratkaisua brute force -menetelmällä. Laskussa on tarkoituksella oletettu kaikki luvut, joten ne ovat vain suuntaa antavia. Vaikka kombinaatioiden täytyisi käydä läpi keskimäärin vain 10 ^ 19, niin brute force -menetelmiä ei kannattaisi käyttää, koska sama työmäärä tarvitaan keskimäärin 64-bittisen salausavaimen murtamiseen. Täytyy kuitenkin huomata, että vaikka satunnaislukugeneraattori olisi laadukas, niin satunnaisuudesta suurin osa katoaa siihen, että alkuperäinen tieto on edelleen tunnistettavissa. Tämä ratkaisu ei ole mikään salausalgoritmi. Turvallisuutta rajoittaa myös erityisen paljon se, että lähdekuvia on rajallinen lukumäärä. Ratkaisusta on kuitenkin suurta apua, jos roskapostien lähettäjän tai muun tahon tietokoneelta kestää edes minuutin selvittää ratkaisu. Tällöin miljoonien sähköpostiviestien päivittäinen lähettäminen vaihtuisi vahingoniloisella tavalla tuhansiin, mikä tarkoittaisi heille liiketoiminnan vaikeutumista.

Käyttäjällä voisi olla lista tuttujen osoitteista, joista hyväksytään viestit ilman CAPTCHA-testiä. Käytettävyyssyistä samasta organisaatiosta ja yhteistyökumppaneilta tulevat sähköpostiviestit täytyisi yleensä hyväksyä ilman tätä menettelyä. Käyttäjiä täytyisi siis pystyä käsittelemään vähintään yksittäisinä, ryhmissä ja domainin mukaan. Ohjelmistoissa kannattaisi olla päivitysmahdollisuus, koska roskapostittajat kehittävät mahdollisesti sovelluksia testien kiertoon.

# 6. Seitsemän ratkaisua

Esitän seitsemän ratkaisua käytännön ongelmiin. Olen suunnitellut ne kaikki kevään 2008 aikana. Varsinkin kuluttajille suunnatuissa tuotteissa on usein selviä kompromisseja, koska suurin osa tuotteista yritetään toteuttaa tavallisina ohjelmistoina. Mietin ratkaisuja mieluummin siltä pohjalta, että niitä olisi tarkoitus käyttää suojaamaan esimerkiksi valtion kannalta vähintään *salaiseksi* (secret) tai jopa *erittäin salaiseksi* (top secret) luokiteltua tietoa. Esittämäni ratkaisut eivät ole tietenkään täysin turvallisia, koska sellaisia ei ole mahdollista toteuttaa. Lisäksi ne tarvitsevat vertaisarviointia. Otsikot ja kuvat ovat englanninkielisinä täsmällisempiä. Onhan käytännössä kaikki kryptologiaa käsittelevä kirjallisuuskin englanniksi. Kuvia ei tarvitse myöskään tehdä uusiksi, jos tarvitsen niistä myöhemmin englanninkieliset versiot toiseen käyttötarkoitukseen.

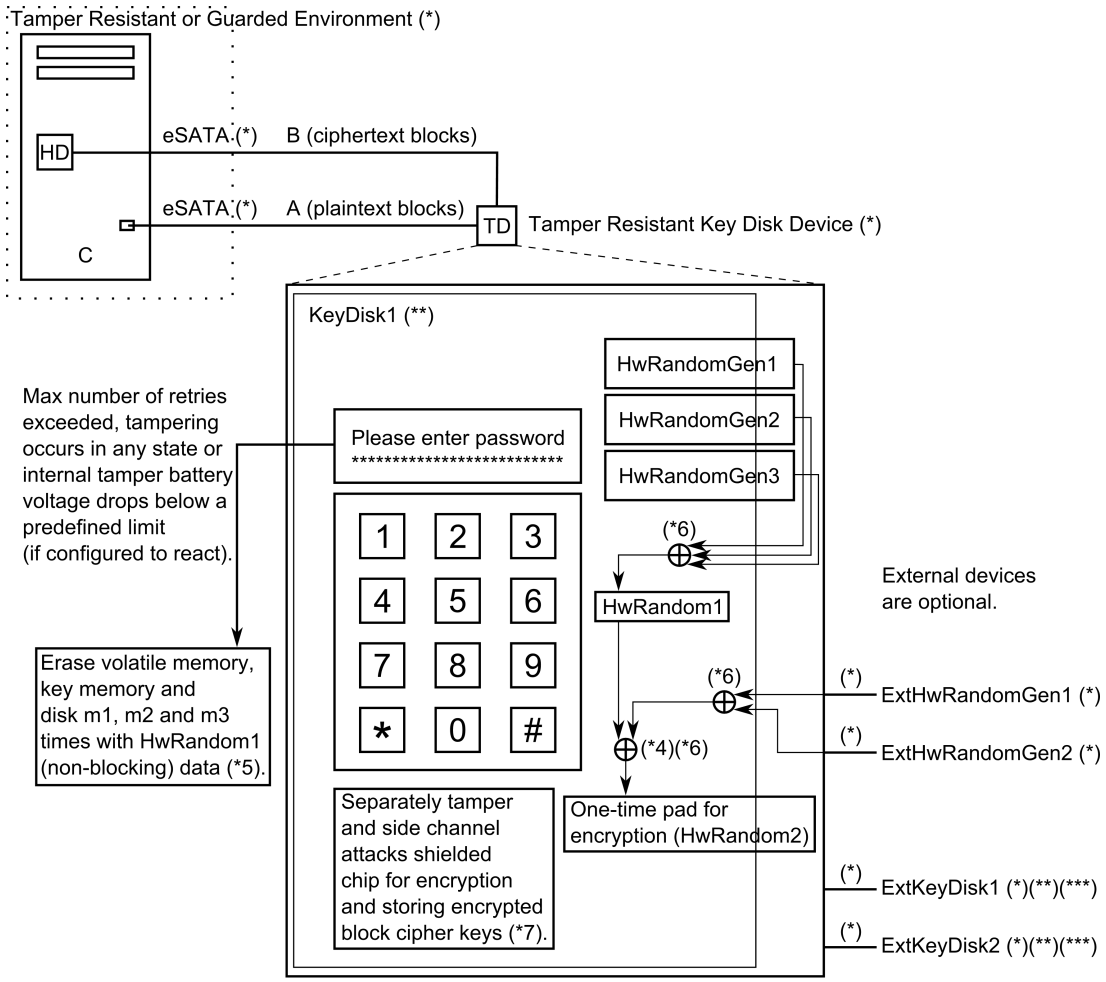
## 6.1. eSATA Disk Encryption Scheme for *Limited Purposes*

One-time pad on ainut täydellisen turvallisuuden salausalgoritmi, mutta vain jos sitä käytetään täsmälleen oikealla tavalla [Schneier, 1996]. KGB käytti sitä kommunikaatiossa vakoojiensa kanssa, mutta he rikkoivat aikoinaan pienessä osassa viestejä turvallisuuden käyttämällä avaimia useampaan kertaan [Benson, 2008], kun ilmeisesti konekirjoittajat eivät enää muuten pystyneet tuottamaan riittävää määrää avainmateriaalia suuren kysynnän vuoksi. Neuvostoliitto kun joutui sotaan Saksan kanssa vuonna 1941. Avaimet luotiin siis manuaalisesti kirjoituskoneella ja sisälsivät siten paljon epäsatunnaisuutta, mutta silti suurinta osaa viesteistä ei ainakaan virallisten tietojen mukaan ole saatu auki lähes 40-vuoden kryptoanalyysin aikana, johon osallistui satoja kryptoanalyytikkoja. Jos algoritmia käytetään oikein, niin se on edelleen melkein täydellinen algoritmi diplomaatti- ja vakoilumaailmassa lyhyiden viestien lähettämiseen. Avaimen kuljettaminen ja tuhoaminen turvallisesti käytön jälkeen ovat nimittäin suurimpia ongelmia.

Valitaan mahdollisiksi merkeiksi isot ja pienet kirjaimet mukaan lukien ääkköset sekä numerot, jolloin merkistössä on 68 merkkiä. Esimerkiksi muodostamaani salakielitekstiä ”zf8zFldt” tai vastaavaa erityisen pitkää viestiä ei pysty mikään organisaatio murtamaan matemaattisesti ilman avainta, vaikka jokainen maailman atomi valjastettaisiin murtamaan näitä kahta viestiä miljardiksi vuodeksi. Ei vaikka salausavain on täsmälleen vain tuon viestin mittainen eli esimerkkitapauksessa 48,7-bittinen (8 * lg 68 tai 2^48 < 68^8 <~ 2^49). Tämä johtuu algoritmin tavallaan täydellisestä unicity distance -arvosta, mikä tekee siitä teoreettisesti murtamattoman. Ainoastaan jos esimerkiksi *pimeä aine* (dark

matter), *pimeä energia* (dark energy) tai muu tieteelle vaikea tai tuntematon asia tallentaa jokaisen tekomme tarkasti muistiin, niin sitten ”joku” voi mahdollisesti käydä kurkkaamassa menneisyydessä. Salausavaimen satunnaisuus on kuitenkin erittäin kriittistä. Viestin salaukseen tai sen purkamiseen oikealla avaimella ei tarvita edes tietokonetta, vaan sen voi tehdä vaikka paperilla ja kynällä keskellä viidakkoa.

Tämä algoritmi on kuitenkin suurimmassa osassa arkipäivän sovelluksia käyttökelvoton. Avainta kun voi käyttää vain kerran, ja avain vaatii salatun tiedon verran eli todella paljon aitoa satunnaisuutta. Lisäksi se täytyisi välittää salakielitekstin purkajalle turvallisesti tavalla tai toisella. Algoritmi on altis myös monille kryptoanalyysin hyökkäyksille, jotka eivät toimi lohkosalaajia vastaan. Seuraava ratkaisu vaatii selvästi laadukkaan laitteistotason toteutuksen satunnaislukugeneraattorin osalta. Jos joku toteuttaisi tämän turvautuen ohjelmistopohjaiseen satunnaislukugeneraattoriin, niin valitsisin epäilemättä kyseisen tuotteen sijasta perinteisen lohkosalaajaan perustuvan tuotteen. Muuten todennäköisyys kryptoanalyysin onnistumiselle olisi liian suuri noiden pelkkien XOR-operaatioiden kanssa. Kiintolevylle kirjoitus kun vaatii niin suuren määrän uutta satunnaista dataa, että sitä on käytännössä mahdotonta kerätä käyttöjärjestelmän satunnaislukugeneraattorista, näppäimistöpainalluksista tai muusta vastaavasta ihmisten luomasta kaaoksesta. Ratkaisu on esitetty kuvassa 6.1.




(*) Tamper and known side channel attacks like TEMPEST and power monitoring attacks require special attention.
(**) Erasing large one-time pads is a major problem in tamper attack, so they must be encrypted with a block cipher in ECB mode. H(HD.block[i] + KeyDisk1.block[i] + ExtKeyDisk1.block[i] + ..) must be saved to KeyDisk1 and verified on each read. One-time pads must NEVER be used twice! Device TD can alternatively be implemented as block cipher device without any one-time pads or external disk devices.
(***) The external disk device must be tamper and side channel attacks resistant. It must contain similar user interface, hardware random number generators and similar shielded chip for encryption and storing block cipher keys. The device and external disk device know each others' public key due to initial setup. The external disk device sends random data to TD, TD adds 256 bits random padding to it and signs the message for example with RSA. Chosen-plaintext attack is possible against RSA without padding. The disk device checks the signature. The devices must encrypt their communication with a session key, which is generated from HwRandom2, padded with HwRandom1 data and encrypted with the key disk public key.
(*4) Electrical tamper attacks e.g. voltage manipulation could alter the XOR output with external random number generators. This and blocking data delay attacks require special attention and may require removing external generators altogether.
(*5) Erasing must be started with all those memory locations containing key material and equivalent plaintext.
(*6) If hardware number generators correlate, advanced mixing functions like block cipher must be used instead.
(*7) Good quality hash function is required and its output length must equal to block cipher key length. The device must support at least up to 40 - 80 digit passwords depending on the chosen cipher.

Encrypt block cipher keys:

diskKey[i] = HwRandom2 data key block
diskKeys = diskKey[0] + ... + diskKey[n]
salt = 128 bits HwRandom1 data
K = H(password + salt)
h = H(diskKeys)
C = E(diskKeys + h, K)
chk = H(C + salt)
Store C and salt to memory
Read the written bytes, calculate hash and compare to chk. If they do not match, a write error occured.
Erase old stored encrypted memory blocks m4 times.

Decrypt block cipher keys:

K = H(password + salt)
D(C, K) = diskKeys + h
compare h and H(diskKeys)

Kuva 6.1. eSATA Disk Encryption Scheme for **Limited Purposes**.

Ratkaisu perustuu SATA-väylän eSATA-toteutukseen, jolloin laite voi olla ulkoinen. HD on kuvassa tietokoneen alkuperäinen tavallinen eSATA-liittimellä varustettu kiintolevy. Se voidaan säilyttää myös erikseen vaikka kassakaapissa. Kaapelien maksimipituus on rajoitettu standardissa. A tarkoittaa SATA-tietoliikennettä tietokoneelle tavallisena kiintolevynä näkyvään ulkoiseen salauslaitteeseen TD. B tarkoittaa vastaavasti salattujen kiintolevylohkojen siirtämistä alkuperäiselle kiintolevylle. Salaus on siis läpinäkyvää tietokoneelle ja kiintolevylle HD. Salauslaitteessa on kiintolevy KeyDisk1, joka on kooltaan suurempi kuin kiintolevy HD. Salaus tehdään tavu- tai bittikohtaisesti one-time padilla eli XOR-operaatiolla. Kirjoitettaessa käytetään aina uutta täsmälleen tallennettavan tiedon kokoista one-time padia, joka tallennetaan avainlevylle. Lisäksi tallennetaan salauslaitteen levylle esimerkiksi SHA-512-summa tavujonosta, johon on yhdistetty peräkkäin jokaisen salaukseen osallistuvan kiintolevyn kyseinen lohko.

Laitteessa on useita mahdollisesti eri tekniikkaan perustuvia laitteistotason satunnaislukugeneraattoreita, jotka on merkitty tunnuksilla HwRandomGen{1-3}. Uusi avainmateriaali muodostetaan ottamalla XOR-operaatio kaikkien satunnaislukugeneraattorien muodostamista tulosteista. Laitteeseen voi lisätä tarvittaessa ulkoisia satunnaislukugeneraattoreita, jotka on merkitty tunnuksilla ExtHwRandomGen{1,2}. Ne voidaan liittää laitteeseen kirjaimellisesti kuin legopalikat. Lisäksi voidaan käyttää useampaa ulkoista avainlevyä, jotka on merkitty tunnuksilla ExtKeyDisk{1,2}. Tällöin saadaan *jaettu salaisuus* (secret splitting), jolloin kiintolevyn HD avaamiseen tarvitaan jokainen kiintolevy. Levyjä hallitaan salauslaitteessa TD, mutta ulkopuoliset levyt vaativat sen, että salauslaite tunnistaa itsensä esimerkiksi digitaalisella allekirjoituksella. Näin avainlevyjen kopiointi vaikeutuu. Satunnaislukugeneraattorit menevät mahdollisesti ajan myötä rikki, joten käyttäjän tulisi olla mahdollista tarkistaa niiden luomaa tulostetta vaikka laitteiden näytöiltä. Tilastolliseen analyysiin perustuvia tarkistuksia kannattaa myös suorittaa automaattisesti.

Muun muassa kiintolevyn käynnistyslohkon voidaan olettaa olevan sisällöltään tiedetty. Tämän vuoksi pelkkä one-time padin käyttö mahdollistaa tuon käynnistyslohkon salausavaimen selvittämisen kokonaan tai osittain. Tämä saadaan ottamalla XOR-operaatio tuosta lohkosta ja tiedetystä sisällöstä. Tämä on *valitun selväkielitekstin hyökkäys* (chosen-plaintext attack) [Schneier, 1996]. Nämä ovat juuri niitä one-time padien sudenkuoppia, joista Schneier [2002] varoittaa. Nyt hyökkääjä voi saastuttaa käynnistyslohkon, jolloin koko toteutus vaarantuu. Saastunut käynnistyskoodi tallentaa tämän jälkeen esimerkiksi tietokoneen sisälle asennetulle äänettömälle SSD (Solid

State Disk)-levylle kaikki käyttöjärjestelmän lukemat avatut kiintolevylohkot. Tämän vuoksi laitteelle TD täytyy tallentaa myös lohkojen tarkistussumma kuten aiemmin kerroin. Hyökkääjällä ei ole mitään käsitystä oikeasta tarkistussummasta, jos häneltä puuttuu yksikin avainlevy. Jos minkä tahansa levyn lohko on muuttunut, niin järjestelmä on vaarantunut, mistä laite TD tiedottaa vähintään esimerkiksi merkkiäänellä ja laitteen näytöllä. Laitteisiin tarvitaan käytännössä pieni näyttö, hallintanäppäimistö ja käyttöliittymä. Jokaisen avainlevyn one-time padit täytyy salata erikseen lohkosalaajalla ja joukolla avaimia. Jokaisella laitteella on siis oma salasana. Muuten tamper-hyökkäyksessä kiintolevystä ei ehditä ylikirjoittaa turvallisesti kuin pieni osa. Loput tarkemmat säännöt olen merkinnyt kuvan tekstiosuuteen. Kuvasta näkee, että turvallisuuden tavoittelu menee monimutkaiseksi.

Vaikka tämä ratkaisu kuulostaisi hyvältä kaupalliselta idealta, niin se soveltuu lähinnä vain hyvin rajoitettuun käyttöön. Jos esimerkiksi tarvitsee jakaa tieto täydellisesti osiin arkistointia varten, niin avainlevyjä voidaan lisätä useampia, jotka sijoitetaan hajautetusti eri kiinteistöihin. Tällöin yksittäisellä levyllä ei ole hyökkääjän käsissä mitään arvoa, vaikka laskentakapasiteettia olisi äärettömästi. Tietokoneen täytyy sijaita turvallisessa ympäristössä. Muuten hyökkääjä voi manipuloida esimerkiksi SATA-väylää siten, että hän saa kopion kaikesta puretusta datasta. Laitteiston täytyisi olla suojattu mahdollisimman hyvin myös esimerkiksi TEMPEST-hyökkäykseltä. *Virrankulutuksen tarkkailu* (power monitoring attack) on vaarallinen hyökkäys erityisesti lohkosalaajan salausavaimien kannalta. Jos suoritusympäristö voidaan olettaa turvalliseksi, niin virta voidaan tuoda salauslaitteeseen ulkoisesta virtalähteestä. Tietokoneen ja salauslaitteen virta kannattaa ottaa puhtaasti akuista, jotta hyökkäys vaikeutuisi.

Lohkosalaajien kanssa voidaan rakentaa useimpia tilanteita ajatellen käytännöllisempi ratkaisu esimerkiksi kiintolevyjen salaukseen kehitetyn XTS-tilan kanssa. Laite on hyvin pitkälle samanlainen ja siihen voidaan jopa yhdistää aiemmin kuvattuja avainlevyjä. Satunnaislukugeneraattoreita tarvitaan kuitenkin pääosin vain salausavainten luonnissa eli harvoin. Laitteessa on kiintolevyn sijaan ainoastaan Flash-muistia, joka on mahdollista ylikirjoittaa useaan kertaan nopeasti silloin, kun joku yrittää avata laitteen. En ota tässä kantaa tamper resistant -tekniikoiden todelliseen turvallisuuteen tiedustelupalveluita vastaan, mutta ainakin ne vaikeuttavat salausavainten selvittämistä. Lisäksi lohkosalaajia käytettäessä voidaan tehdä helpommin käyttäjäryhmiä, jolloin käyttäjällä voi olla pääsy vain tiettyyn osaan kiintolevyn varausyksiköitä. Voidaan luoda myös piilotettuja osioita kuten ohjelmallisissa tuotteissa. Tätäkään ratkaisua ei

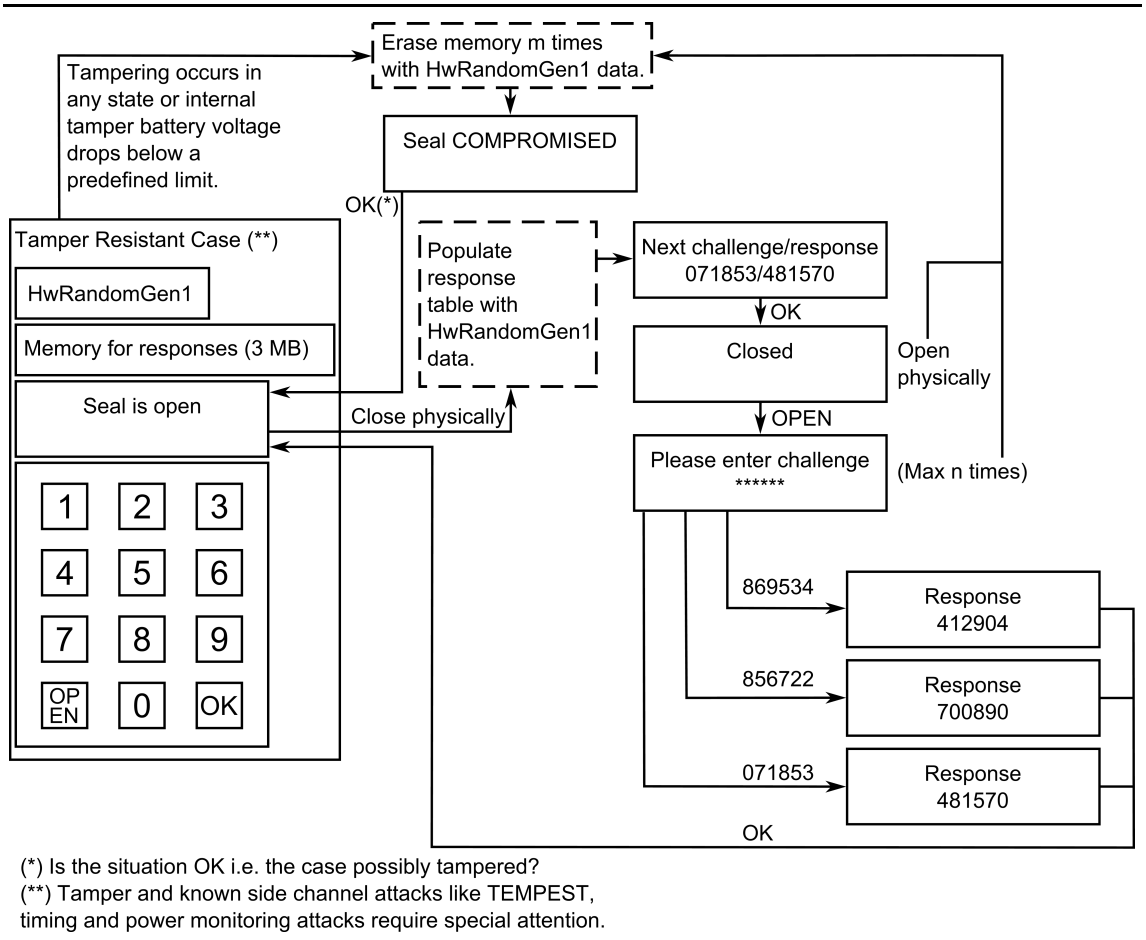
voi suositella kotitietokoneeseen, sillä hyökkääjä voi kerätä eSATA-kaapelin liikennettä. Myös mikropiirejä on mahdollista manipuloida.

Tavallinen kotona tai töissä sijaitseva mikrotietokone ei ole joka tapauksessa turvallinen ajoympäristö, joten markkinoilla olevat ohjelmalliset lohkosalaajiin perustuvat tuotteet ovat järkevä ja kustannustehokas vaihtoehto. Osassa tuotteita voi luoda käynnistyslevyn esimerkiksi USB-muistista, jolta salausohjelmisto lataa pienen käyttöjärjestelmänsä. Näin koneen kiintolevyn saastutettu käynnistyslohko ei vaaranna toteutusta. USB-väylän tai muun laitteiston manipulointi kyllä vaarantaa, joten taas ollaan umpikujassa.

### 6.2. Tamper Resistant Challenge-Response Based Seal

Kuvan 6.2a elektroninen sinetti soveltuu käytettäväksi esimerkiksi tavallisiin arkistokaappeihin, joissa on yleensä turvattomat lukot. Voipa sitä käyttää myös vaikka lukitussa kaapissa sijaitsevan tärkeän asiakirjakansion sinetöintiin. Dokumentit, palvelinhuoneet ja vastaavat täytyisi saada pidettyä luottamuksellisuuden kannalta suojassa esimerkiksi vakoojalta, jonka tiedustelupalvelu on värvännyt yrityksen henkilökunnasta. Murrettu sinetti kun tarkoittaisi käytännössä sitä, että yritys parantaisi turvatasoaan merkittävästi. Vakoojan toiminta vaikeutuisi ja hän jäisi todennäköisesti kiinni.

Kyseessä on tamper resistant -tekniikkaa käyttävä laite. Jos sitä yrittää avata luvatta, niin se ylikirjoittaa muistipiirinsä niin monta kertaa kuin mahdollista laitteistotason satunnaislukugeneraattorin luomalla datalla, jos se kykenee luomaan satunnaista dataa muistipiirin kirjoitusnopeudella. Ratkaisun sijaan voi käyttää esimerkiksi Abloyn riippulukkojen malleja PL321/20 tai PL330 Protec-lukkopesällä. Ne toimivat tavallaan sinetin tavoin, kun tuota lukkopesää on nykytiedon valossa melkein mahdoton tiirikoida. Rikottua lukkoa on sanotaanko vaikea perustella seuraavana työpäivänä. Riippulukolla tai sinetillä voi lukita asiakirjakansion tai vaikka kaupan muovipussin, kunhan käyttää kekseliäisyyttä.

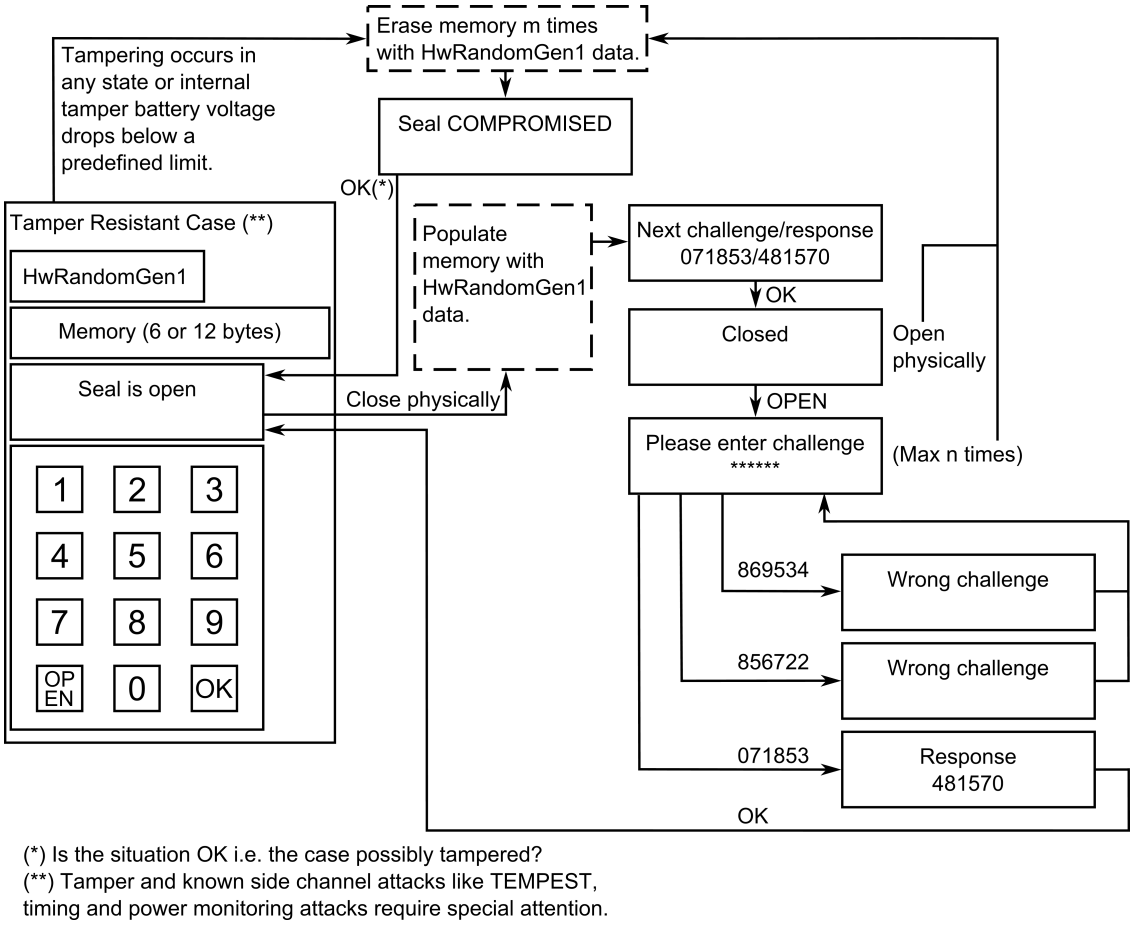



Kuva 6.2a. Tamper Resistant Challenge-Response Based Seal.

Kuvaan ei ole merkitty kaikkia tilasiirtymiä selvyyden vuoksi. Kun laitteen sulkee, niin se ylikirjoittaa muistiin miljoona kuusinumeroista eli kolmetavuista numeroa, jolloin löytyy vaste jokaiselle haasteelle indeksin perusteella. Laite valitsee satunnaisesti jonkin haasteen. Se näyttää haasteen ja sitä vastaavan vasteen ruudulla, jotka käyttäjä kirjoittaa ylös vaikka paperille. OK-painikkeen jälkeen sinetti on lukittu. Jos sinetin avaa fyysisesti tai syöttää haasteen väärin n kertaa, niin laite ylikirjoittaa muistinsa m kertaa satunnaislukugeneraattorin luomalla datalla. Hyökkääjä ei tiedä, mikä haaste-vaste-pari on oikea. Kun käyttäjä painaa vasteen tarkistuksen jälkeen OK-painiketta, niin laite palaa avattu-tilaan.

Tiedustelupalveluilla on keinoja murtaa tamper resistant -tekniikoita. Tällöin he voivat kloonata muistin sisällön uuteen väärennettyyn sinettiin. Varsinkin tällainen irrallinen sinetti on teoriassa mahdollista vaihtaa väärennettyyn. Löysin Unistolta hieman samanlaisen Unisto Crypta 2K -tuotteen, mutta siinä hyökkääjä näkee aina oikean vasteen nappia painamalla, joten mitään salaisuutta ei ole. Tiedustelupalvelun väärennysoperaatio helpottuu turhankin paljon. Ratkaisussani hyökkääjä ei tiedä oikeaa haastetta tai vastetta, joten mieluiten koko muisti pitäisi onnistua kloonaamaan.

Kuvassa 6.2b on esitetty toinen vaihtoehto toteutukselle. Palaamme siis yksinkertaisempaan malliin. Siinä käytetään vain yhtä kertakäyttöistä haaste-vaste-paria, jolloin toteutus on edullisempi toteuttaa, ja vaste ehditään ylikirjoittaa useamman kerran tamper-hyökkäyksen aikana. Käyttäjältä kysytään haaste eli salasana, jonka täytyy olla oikein. Kun haaste on oikein, niin vaste näytetään ruudulla. Jos laitetta on mahdollisesti manipuloitu, niin kriittisissä kohteissa sinettiä ei voi enää sen jälkeen käyttää, mutta sen voi ehkä siirtää vähemmän tärkeiden kohteiden suojelemiseen.




Kuva 6.2b. Modified Tamper Resistant Challenge-Response Based Seal.

### 6.3. Tamper Resistant Password Safe

Aidosti satunnaisten salasanojen muistaminen on vaikeaa ja niitä kertyy paljon. Ohjelmallista salasanatietokantaa turvallisempi olisi tamper resistant -tekniikkaan perustuva mukana kuljetettava salasanatietokanta. Sen tulisi olla mahdollisimman pieni eli MP3-soittimen kokoinen. Tällainen ratkaisu löytyy ainakin Mandylion Research Labs -nimiseltä yritykseltä. Yrityksen tuotteita on käytetty Yhdysvaltain armeijassa, mikä on sinänsä tietysti epäilyttävää kriittisimmän käytön kannalta. Yritys kun saa rahoitusta saman maan

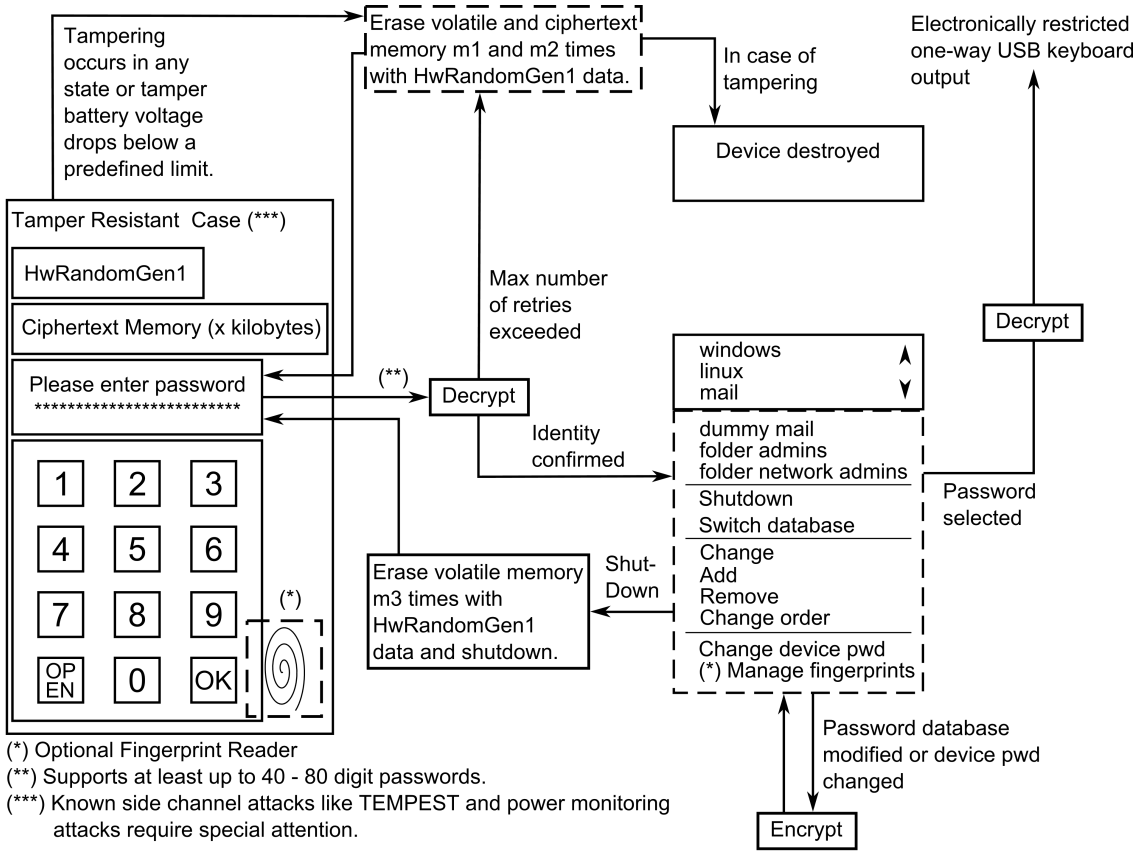
puolustushallinnolta. Tuotteen kaupalliseen versioon on voitu tehdä takaportteja esimerkiksi ohjelmistoon tai salausalgoritmien toteutukseen. Tamper resistant -toteutusta on ehkä heikennetty, tai salasanojen generointiin tarkoitettuun satunnaislukugeneraattoriin on lisätty tarkoituksella jaksollisuutta. Ehkä yritys toimittaa esimerkiksi tuotearviointeja varten turvallisia versioita ja hieman turvattomampia kuluttajille ja tietyille yrityksille. Turvattomimpaan tiedusteluun tarkoitettuun versioon on voitu upottaa radiovastaanotin, joka aktivoi salasanan perusteella radiolähettimen. Varsinkin CIA on varmasti kiinnostunut tällaisesta manipuloinnista. Vaikka tuote olisi valmistettu EU:ssa, niin silti valtiot saattavat vakoilla toisia EU-maita. kaupallisiin tuotteisiin liittyy aina tällaisia riskejä. Luotettavinta olisi hankkia ratkaisu, joka on valmistettu luotettavassa kotimaisessa turvallisuusteknologian yrityksessä.

Tuotteessa on tietokoneella ajettava hallintaliittymä ja muuta turhaa, joita vainoharhaisimmat käyttäjät eivät halua. Lisäksi laitteen avaaminen nuolinäppäinten kombinaatiolla ei kuulosta kryptoanalyysin kannalta turvalliselta. Ehkä siinä on kaksitasoinen tamper resistant -toteutus, jolloin varsinainen salausavain on erikseen suojatussa yksinkertaisen mikrosirun ja muistipiirin yhdistelmässä. Näin tuolla mikrosirulla olisi kaksinkertainen suojaus. Siru luovuttaa salausavaimen toiselle piirille, kunhan kombinaatio on oikea.

Kuvan 6.3 ratkaisu on toiminnoiltaan yksinkertaisempi. Sitä voisi käyttää esimerkiksi kertakäyttöisten salasanojen syöttämiseen pankkiholvin avauksessa. USB-liitin esimerkiksi Kason rakentaman holvin massiivisessa seinämässä on kieltämättä huvittava ajatus, mutta se lisäisi turvatasoa, kun käytetään lisäksi perinteisempiä lukkomekanismeja ja biometristä käyttäjätunnistusta. Näin käyttäjän täytyy muistaa jotain, omistaa jotain ja olla jotain. Lisäksi aikalukko estää holvin avaamisen poliittisesti epäkorrektiin kellonaikaan.

Kyseessä on yksisuuntainen USB-näppäimistölaite, joka ei salli muun muassa firmware-päivityksiä. Laite tukee hierarkkista listaa salasanoista hakemistojen tapaan. Lisäksi tuetaan useita salasanatietokantoja, jolloin käyttäjä voi käyttää samaa laitetta eri salasanalla ja listalla käyttäjätunnuksia esimerkiksi kotona ja töissä. Sormenjälkitunnistin on valinnainen, ja biometristä tunnistetta ei käytetä salauksessa. Salaus tehdään siten, että tietokannan salasanan, suolaarvon ja hash-funktion avulla saadaan jokaiselle salattavalle salasanalle oma salausavain. Samoin lista tietyn salasanatietokannan sisältämistä käyttäjätunnuksista on salattu omalla salausavaimellaan. Jos hyökkääjä ei saa käsiinsä hyvin paljon satunnaisuutta sisältävää salasanaa, niin hash-funktio tai vaihtoehtoisesti jokainen salausavain täytyisi murtaa erikseen. Jos side channel attack on mahdollinen, niin siinä tapauksessa hyökkääjä saa haltuunsa enintään ne salasanat, jotka käyttäjä avaa hyökkäyksen aikana. Ohjelmamuistiin kun ei

lueta niitä salakielilohkoja, joita ei tarvita. Salasanoihin ja listaan käyttäjätunnuksista lisätään ylimääräistä padding-dataa sen vuoksi, että hyökkääjälle ei selviä salakielitekstistä käyttäjätunnusten tai salasanojen mahdollinen lyhyys.




Encrypt:

Check that other indexFiles do not use the
same password. See Decrypt for details.
(*4) Pad data so that calculated content (e.g. password)
length is rounded up to next 16 bytes.
padding = HwRandomGen1 data (*4)
padding2 = HwRandomGen1 data (*4)
IV = HwRandomGen1 data
IV2 = HwRandomGen1 data
salt = 128 bits HwRandomGen1 data
salt2 = 128 bits HwRandomGen1 data

K = H(password + salt)
P = pwdLen[i] + pwd[i]
h1 = H(P)
C = E(P + h1 + padding, K, IV)
pwdFile[j][i] = salt + IV + C
Store pwdFile[j][i] to memory and verify the written bytes.
Erase possible old pwdFile[j][i].

K2 = H(password + salt2)
passwordNames = pwdCount + pwdNameLen[0] + pwdName[0] + ... + pwdNameLen[i] + pwdName[i] + ...
+ pwdNameLen[n] + pwdName[n]
h2 = H(passwordNames)
C2 = E(passwordNames + h2 + padding2, K2, IV2)
indexFile[j] = salt2 + IV2 + C2
Store indexFile[j] to memory and verify the written bytes.
Erase possible old indexFile[j].

Decrypt:

find indexFile[j] such as
indexFile[j] = salt[j] + IV[j] + C[j]
K[j] = H(password + salt[j])
D(C[j], K[j], IV[j]) = passwordNames + h2 + padding2
H(passwordNames) == h2
Populate user interface with passwordNames.

When password is selected from user interface:
pwdFile[j][i] = salt + IV + C
K = H(password, salt)
D(C, K, IV) = P + h1 + padding
Verify that H(P) == h1.
P = pwdLen[i] + pwd[i]
Send pwd[i] to USB interface.

Kuva 6.3. Tamper Resistant Password Safe.

### 6.4. Cryptographic Alarm System

Kuten aiemmin kerroin, niin nykyisiä hälytysjärjestelmiä on mahdollista puijata teoriassa melko yksinkertaisilla menetelmillä. Kuvan 6.4 ratkaisussa laitteiden välinen kommunikaatio perustuu kryptologiaan. Kommunikaatio on toteutettu rekursiivisesti koko laitepuuhun, ja samaa protokollaa käytetään jokaisen laiteparin välillä. Viestit lähetetään ja käsitellään laiteparin välillä sarjassa, jotta synkronointi ei aiheuta ylimääräistä monimutkaisuutta. Toki laite voi periaatteessa lähettää kyselyitä rinnakkain useille laitteille. Tämä voi olla järkevää varsinkin silloin, jos keskuslaitteeseen on liitetty suuri määrä hälytinlaitteita. Viiveet verkossa kun voisivat aiheuttaa hälytyksen ilman rinnakkaisuutta.

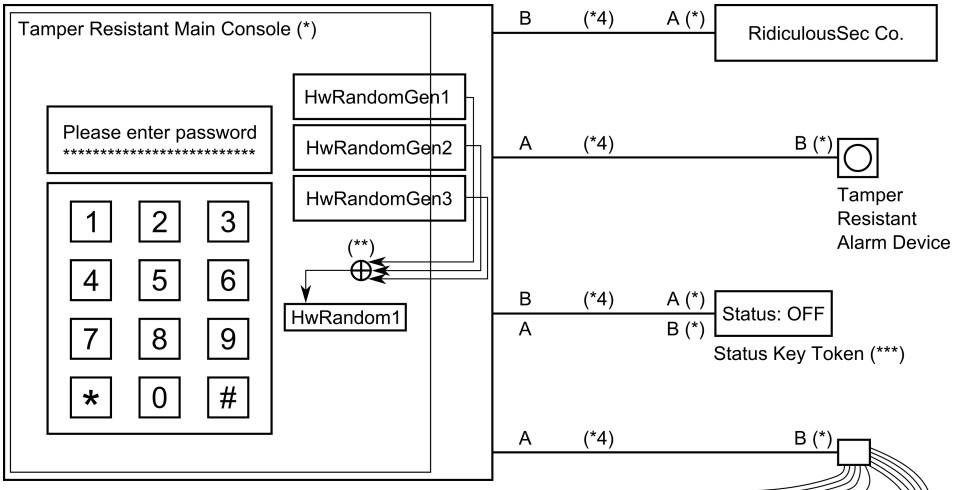
Kerran päivässä tai jopa tunnin välein vartiointiliike aloittaa sessioavainparin vaihto-operaation, joka suoritetaan rekursiivisesti koko järjestelmään. Kukin laitepari sopii keskenään kaksi toisistaan riippumatonta yhteistä sessioavainta Diffie-Hellman-algoritmilla, jolla voidaan sopia yhteinen salausavain kahden osapuolen välille. Laite B yhdistää kaikki algoritmin julkiset luvut, ottaa tuloksesta hash-arvon, allekirjoittaa tuloksen salaisella avaimellaan ja lähettää muodostamansa luvut sekä allekirjoituksen laitteelle A. Näin laite A voi varmistua siitä, että kyseessä on todella laite B. Jos halutaan vähentää käytettävien algoritmien määrää ja siten todennäköisyyttä tietoturva-aukoille, niin laitteen A luoma sessioavain voidaan välittää salattuna RSA:lla Diffie-Hellmanin käytön sijaan.

Esimerkiksi kerran sekunnissa laite A generoi lohkon verran satunnaista dataa, salaa sen yhteisesti sovitulla sessioavaimella ja lähettää salakielitekstin laitteelle B. Se puolestaan avaa lohkon salauksen ja ottaa selväkielitekstistä hash-funktion arvon. Toisin sanoen pong on hash ping (pong is hash ping). Sen jälkeen laite salaa hash-arvon toisella symmetrisellä avaimella ja lähettää uuden salakielitekstin laitteelle A, joka avaa ja tarkistaa viestin sisällön. Koska käytössä on laitteistotason satunnaislukugeneraattorit, niin ylimääräinen avain aiheuttaa karkeasti ottaen muutaman muuttujan ja funktiokutsun verran lisää koodia. Hyökkääjä ei tiedä viestiä eikä kumpaakaan avainta. Hän tietää vain hash-funktion aiheuttaman muutoksen. Hyökkäys vaatii pahimmillaan esimerkiksi 256-bittisen AES:n tapauksessa 2 ^ 640 yrityskertaa tunnin aikana, mitä voidaan pitää melkein järjettömänä ajatuksena. On todennäköisesti helpompaa murtaa noita epäsymmetrisiä algoritmeja. Allekirjoitukseen käytetyt epäsymmetriset avainparit vaihdetaan kerran kuukaudessa. Laitteen A luoma aikaleima saa olla melko pitkä, millä vaikeutetaan kryptoanalyysiä. B allekirjoittaa vanhalla avaimella uuden avaimen julkisen avaimen, ID:n ja aikaleiman. Ennen allekirjoitusta selväkieliteksti ajetaan hash-funktion läpi.

Hälytys tapahtuu, jos allekirjoitus tai muu data ei täsmää jossain protokollan vaiheessa, tai esimerkiksi muutaman sekunnin aikaviive ylittyy. Aikaviiveitä täytyy mahdollisesti säätää, jos järjestelmässä on suuri määrä kiinteistöjä. Hälytys aiheutetaan siten, että esimerkiksi keskusyksikkö ei vastaa minuuttiin vartiointiliikkeen device query- tai muihin viesteihin. Toisin sanoen vartiointiliikkeessä tapahtuu tällöin hälytys jo muutaman sekunnin jälkeen. Kun tapahtunut hälytys mitätöidään eli palautetaan järjestelmä normaalitilaan, niin mahdollisesti kesken jäänyt epäsymmetristen avainten vaihto-operaatio suoritetaan loppuun. Tämä tapahtuu lähettämällä kyseinen viesti uudelleen rekursiivisesti niille laitteille, joiden avainparia ei onnistuttu vaihtamaan. Muuten epäsymmetrinen avainpari jäisi vaihtamatta jopa vuosiksi, jos hyökkääjä aiheuttaisi uuden hälytyksen aina kerran kuukaudessa. Tämän jälkeen aloitetaan sessioavainparin vaihto-operaatio.

Jos jotain laitetta yrittää avata, niin se muuttuu muistin ylikirjoituksen vuoksi käyttökelvottomaksi, jolloin järjestelmä täytyy konfiguroida uudelleen. Muutenhan aiheutuisi jatkuvasti uusia hälytyksiä. Kun hälytykset ovat pois päältä esimerkiksi työpäivän ajan, niin viestejä ei lähetetä hälytinlaitteille. Keskusyksikön täytyy kuitenkin vastata vartiointiliikkeen kyselyihin normaalisti. Laite pitää aina roolissa B kirjaa siitä, koska on tullut viimeksi kysely. Tämän ja hälytyksen aikarajan perusteella voidaan päätellä onko hälytykset päällä vai ei. Laitteiden kellojen täytyisi olla mahdollisimman tarkkoja. Muuten eri nopeudella kulkevat kellot mahdollistavat teoriassa yhden hyökkäystavan. Vastaavasti kellojen synkronointi aiheuttaisi ongelmia ja samanlaisen hyökkäyksen.

Tietoliikenne voidaan toteuttaa fyysisesti tavallisilla verkkokaapeleilla ja TCP:llä, jolloin saadaan samalla myös tietoliikenteen virheenkorjaus ja IP-osoitteet. Buffer overflow- ja muihin hyökkäystapoihin täytyy tietenkin kiinnittää erityistä huomiota. On kyseenalaista rakentaa näin monimutkaisia järjestelmiä, jos ne voidaan sivuuttaa huolimattomuuden tai kiireellisen toteutusaikataulun vuoksi varsin yksinkertaisesti. Epäsymmetriset algoritmit on myös helppo toteuttaa turvattomasti. Tietoliikenteen virheenkorjaus täytyy myös miettiä tarkasti, jotta vääriä hälytyksiä ei tulisi turhaan. Järjestelmän asennuksessa tai kokoonpanon muuttuessa täytyy tarkistaa laitteen A näytöstä uuden laitteen B julkisen avaimen sormenjälki kuten tavallisissa SSH-asiakasohjelmissa silloin, kun esimerkiksi palvelimen epäsymmetrinen avainpari on vaihtunut. Kuvassa on myös varsin vainoharhainen ratkaisu tavallisten ikkunoiden tilalle. Ikkunan kuituoptinen toteutus täytyy rakentaa osittain seinien sisälle siten, että tiloihin ei pääse ikkunaa irrottamalla. Ikkuna voidaan rakentaa kerroksittain luodin- ja iskunkestäväksi sekä optisesti yksisuuntaiseksi.




(*) Tamper and known side channel attacks like TEMPEST and power monitoring attacks require special attention. All devices must have hardware random number generators. If tampering occurs or internal tamper battery voltage drops a predefined limit, the device must erase key and volatile memory m1 and m2 times. High precision clocks must be used, but clock synchronization is not used.
(**) If hardware number generators correlate, advanced mixing functions like block cipher must be used instead.
(***) This optional device is used to prevent Alarm System Panel Clone Attack and as digital access control key. Slightly modified encryption scheme is used and recursion is disabled.
(*4) The following encryption scheme is used. All requests are executed recursively for each node starting from the security company. Each device must get valid response from all child nodes (B) before sending response to A. Requests are sent to B sequentially. Each request must complete on each connection before starting a new request.

**Session key pair exchange** (once per day or even per hour):

A and B agree n, n2, g and g2 for Diffie-Hellman algorithm.
A knows public key of B and corresponding key ID.
X = A computes Diffie-Hellman public large integer from n, g and secret large integer x.
X2 = A computes Diffie-Hellman public large integer from n2, g2 and secret large integer x2.
A sends ID, X and X2 to B.
Y = B computes Diffie-Hellman public large integer from n, g and secret large integer y.
Y2 = B computes Diffie-Hellman public large integer from n2, g2 and secret large integer y2.
s = B signs H(n + n2 + g + g2 + X + X2 + Y + Y2) with RSA or DSA with key having key ID.
B sends Y, Y2 and s to A.
A verifies s with B public key.
K = Diffie-Hellman negotiated key derived from n, g, X, Y and secret large integer x or y
K2 = Diffie-Hellman negotiated key derived from n2, g2, X2, Y2 and secret large integer x2 or y2
Instead of Diffie-Hellman, RSA encryption can be used to deliver a session key pair to B.

**Device query** (once per second):

ping = random data generated by A with size of block cipher block
C = E(ping, K)
A sends C to B.
ping = D(C, K)
pong = H(ping)
C2 = E(pong, K2)
B sends C2 to A.
pong = D(C2, K2)
A verifies that pong == H(ping).

**Asymmetric key pair exchange** (once per month):

aKeyOld [ID, public, private] = currently used asymmetric key pair of B
aKeyNew [ID, public, private] = new asymmetric key pair generated by B
timestamp = 256 bits random data generated by A
A sends timestamp and aKeyOld.ID to B.
signNew = sign(H(aKeyNew.ID + aKeyNew.public + timestamp), aKeyOld.private)
B sends aKeyNew.ID, aKeyNew.public and signNew to A.
A verifies that signNew matches H(aKeyNew.ID + aKeyNew.public + timestamp).
After that aKeyOld is deprecated for A and not accepted any more.
In error situations key pair exchange can be restarted with previous key ID.
B stores old keys for example one year.

Alarm is initiated in role A if signature or other data does not match, or time limit or retry count for single key exchange or device query exceeds. Each device initiates alarm in role B by not responding to messages sent by A for a minute. System reset after alarm is initiated by root element after the main console continues responding and continues with possibly unfinished asymmetric key pair exchange followed by a session key pair exchange. When the alarm system is set off, the main console does not send queries to alarm devices. Each device in role B knows time of the last received message. If time interval is less than alarm time limit and e.g. movement is detected between the queries, alarm is initiated. If time interval between two queries is greater than the alarm time limit, this means the alarm system is off and alarm is not initiated. In tamper situations the system must be reconfigured, since the tampered device has erased its memory. Error correction in network connections (Ethernet) is critical for the whole system to reduce probability of false alarms. In initial setup public key signatures must be validated manually like in SSH clients when connecting to a new host.

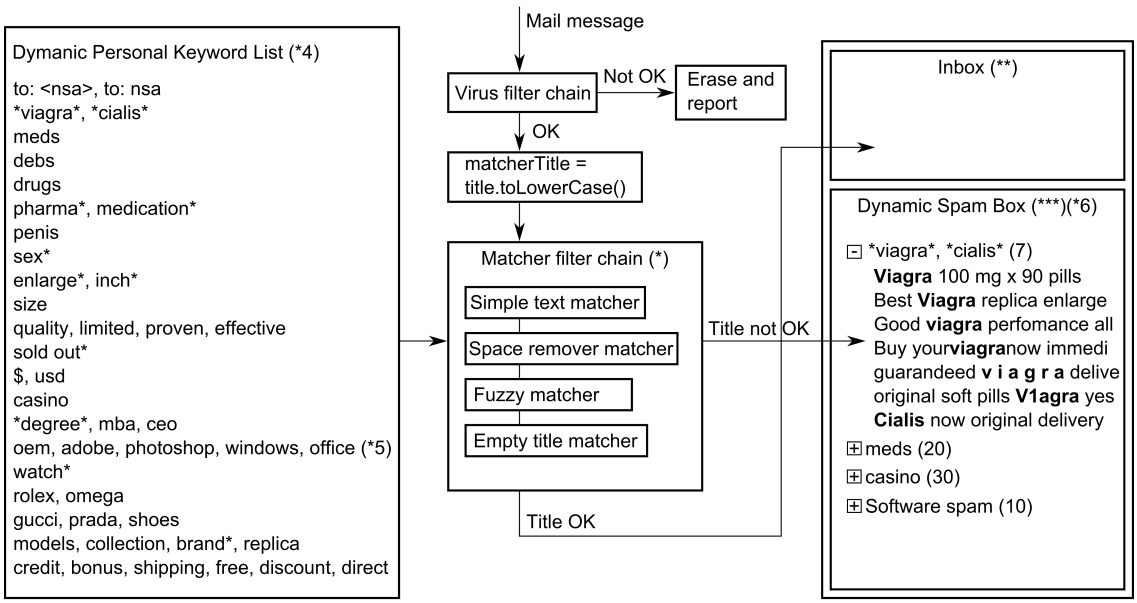
Kuva 6.4. Cryptographic Alarm System.

### 6.5. Human Concept and Shape Recognition Based Spam Filter

Tämän ratkaisun keksin siitä, miten karsin roskapostia viestien otsikoiden perusteella. Yksinkertaisuudestaan huolimatta siinä on mukana ainakin ripaus käytettävyyden, käsitteellisen mallintamisen, algoritmien ja ohjelmistokehityksen tietämystä. En lue roskapostien otsikoita yleensä kunnolla vaan etsin niistä avainsanoja. Viestien otsikoissa on tällä hetkellä yleensä aina jotain avainsanoja. Otan huomioon myös sen, että roskapostien otsikot ovat varsin pitkiä ja pääosin englanniksi. Toisin sanoen lyhyt suomenkielinen viesti erottuu edukseen roskan joukosta. Automaattisiin suodattimiin ei voi toistaiseksi täysin luottaa, minkä vuoksi moni ei käytä niitä lainkaan. Ainakin roskapostiviestejä joutuu tarkistamaan jälkikäteen ennen tuhoamista. Yleensä roskaksi tunnistettuja viestejä ei järjestetä avainsanojen mukaan, jolloin niiden tarkistaminen on työlästä. Suuri osa tutkimuksesta tuntuu keskittyvän siihen, että roskaposti täytyisi tunnistaa mieluiten kokonaan automaattisesti [Sundström, 2008].

Ihmisen aivot ovat kehittyneet tunnistamaan muotoja tekstiä huomattavasti tehokkaammin. Kun katsomme vaikka kattilaa tai tuolia, niin meidän ei tarvitse edes erikseen ajatella kyseistä termiä. Lisäksi näitä tunnistustapahtumia tapahtuu rinnakkain ja sellaisella nopeudella, että esimerkiksi formulakuski pysyy radalla. Havaitsemme myös esimerkiksi nopeasti tutun ihmisen väkijoukosta. Lisäksi ihmiset mitä ilmeisimmin käsittelevät samaan kontekstiin kuuluvia käsitteitä ja termejä nopeammin kuin eri kontekstiin kuuluvia. Voidaan puhua melkoisesta käsitteen- ja hahmontunnistuksen supertietokoneesta. Tämä supertietokone kestää tietyn määrän pakkasta, kuumuutta, vettä ja muita ankaria olosuhteita. Ainut suurempi rajoitus on se, että akkuja täytyy ladata mieluiten noin kolmannes päivästä optimaalisen suorituskyvyn takaamiseksi.

Ratkaisussani ei kiinnitetä lainkaan huomiota viestien sisältöön vaan ainoastaan otsikoihin. Lähes aina viestin otsikon ja vaihtoehtoisesti lähettäjän nimen perusteella voidaan päätellä, onko viesti roskapostia. Näin ei ole mitään hyötyä siitä, että roskapostia lähettävä taho muuttaisi viestin sisällön kuvaksi. Jos otsikko on tyhjä, niin viestin voi analysoida lähettäjän nimen perusteella. Kotimainen tai tuttu nimi erottuu tuntemattomien joukosta. Käytän termiä *avainsana* (keyword), koska loppukäyttäjillä ei ole todennäköisesti useinkaan mitään käsitystä siitä, mikä on käsite tai termi. Kaikki otsikoiden kirjaimet muutetaan ensin vertailua varten pieniksi, mikä helpottaa merkkijonojen vertailua. Avainsanat syötetään myös pienillä kirjaimilla. Viestilaatikossa otsikot näytetään kuitenkin alkuperäisessä muodossa. Ehdotelma on esitetty kuvassa 6.5.




(*) Inheritance or other way to call other filters should be provided, since for example space remover matcher can inherit or use simple text matcher.
(**) It should be possible to select some word and add it (context menu and program menu etc.) to existing keyword group. It must be possible to create new groups when adding a keyword. Help including good examples is mandatory.
(***) When some of folders (for example casino) is emptied, the whole folder is removed. So spam box contains only folders, which have emails. Ctrl+a or similar shortcut selects only emails in the selected folder. It should be possible to change mail ordering like in current email programs and Windows file search. The found keyword should be highlighted in some way. Bold characters would be enough, because for example red characters would draw too much attention from other words in the title. The whole design needs rigorous usability testing.
(*4) Keyword is deleted if match is not found for a long time and a predefined number of messages have been processed. Some kind of recycle bin type functionality would be useful. In this way it is possible to restore keywords in error situations. It must be possible to change order of the keyword groups.
(*5) User has defined alias "Software spam" for this group.
(*6) Messages are ordered within each folder in the following order by default.
1. Keyword frequency
2. Keyword starting location (alignment) in title. See example.
3. Title length
4. Identical titles
5. Sender name.

Kuva 6.5. Human Concept and Shape Recognition Based Spam Filter.

Roskapostit ja avainsanajoukko eivät ole ratkaisussani käyttäjälle samanarvoisia. Jos viesti sisältää avainsanoja ”viagra”, ”cialis" tai vastaavia, niin jopa tuhansien viestien joukko on käyty nopeasti läpi, kun viestit on esitetty esittämässäni järjestyksessä. Sen sijaan avainsanan ”models” sisältämien viestien läpikäyminen on hitaampaa. Järjestys tukee kielen lukemista vasemmalta oikealle ja ylhäältä alaspäin sekä sanojen esiintymistä pystysuunnassa mahdollisimman lähellä toisiaan. Näin silmien ei tarvitse siirtyä riveillä edestakaisin. Liiallista avainsanojen korostamista tulee välttää, jotta otsikoiden lukeminen ei vaikeudu. Viestit järjestetään ensisijaisesti avainsanojen frekvenssin mukaan. Näin viestit pysyvät kokonaisuuksina ja aivot voivat käsitellä ne nopeammin. Toisiaan muistuttavat viestit ja avainsanat voidaan käsitellä tavallaan hahmoina. Käytän itse tätä menetelmää, kun esimerkiksi järjestän hakemistolistauksessa tai tiedostohaussa tiedostojen nimet päätteen tai muokkauspäivämää-

rän mukaan. Tyhjät otsikot eivät sisällä avainsanoja, joten ne lajitellaan omaan kansioonsa ja järjestetään lähettäjän nimen mukaan.

Suodatinketju täytyisi olla muokattavissa. Lisäksi niiden järjestystä tulisi olla mahdollista muuttaa. Myöhemmin voidaan tehdä vaikka suodatin tunnistamaan satunnaiset merkkijonot, jos roskapostittajat keksivät vastaiskuna laittaa otsikoksi pelkkiä hash-funktion arvoja. On erittäin tärkeää esittää toimintaperiaate helposti ymmärrettävällä tavalla sähköpostiohjelman ohjeistuksessa ja käyttöliittymässä. Avainsanajoukkoa vastaava kansio poistetaan näkyvistä, kun käyttäjä on tuhonnut siitä kaikki viestit. Näin hän tuntee saavansa jotain aikaan ja voi poistella viestejä joukoittain vaikka eri päivinä. Näin ongelma jaetaan osiin ja ei mene hermo tuhansistakaan roskapostiviesteistä. Toisin sanoen hajota ja hallitse. Sääntöjä täytyy olla mahdollista lisätä, poistaa ja muokata yhdestä asetusnäkymästä. Avainsanan lisääminen avainsanajoukkoon tapahtuu lisäämällä se suoraan asetuksista tai valitsemalla viesteistä sanoja. Kun jonkin sanan sisältävä roskaposti alkaa ilmestyä turhan usein Inbox-laatikossa, niin käyttäjä voi valita kyseisen sanan ja tehdä siitä säännön.

Avainsana poistetaan automaattisesti, jos sille ei löydy osumia tietyn kynnysarvon ylittävän viestimäärän kuten viidentuhannen viestin sekä ohjelman muutaman eri viikoilla tapahtuvan käyttökerran jälkeen. Käyttäjä voi myös lukita tarkoituksella jotkin säännöt, jolloin niitä ei poisteta koskaan. Jos laskettaisiin vain viestien lukumäärää, niin ohjelmistokehittäjän vahingossa itselleen aiheuttama viidentuhannen viestin joukko tuhoaisi kaikki säännöt. Poistetut säännöt voidaan säilyttää jonkin aikaa. Tällä sääntöjen karsimisella pidetään sääntöjoukko kohtuullisen kokoisena, vaikka sinne lisättäisiinkin vahingossa turhia sääntöjä. Säännöissä ei tarvita välttämättä muita jokerimerkkejä kuin tähtimerkkiä. Jos sana ei sisällä jokerimerkkejä, niin sen täytyy esiintyä sellaisenaan eikä esimerkiksi osana toista sanaa. Liiallinen monimutkaisuus ja esimerkiksi regular expressionit ajavat tavalliset käyttäjät pois. Toki ne voidaan periaatteessa tarjota edistyneille käyttäjille.

Käyttäjille muodostuu erilaisia avainsanajoukkoja, mikä on ikävää roskapostittajalle. Ohjelman asennuksessa voidaan toki hakea valmistajan kotisivulta ajantasainen sääntöjoukko, jolla pääsee alkuun. Oman listani muodostin käymällä läpi satoja roskapostiviestien otsikoita. Systeemin heterogeenisyys lisää sen kestokykyä erilaisia hyökkäyksiä vastaan. Tämä pätee eläimen ja ihmisen DNA:han bakteereita ja viruksia vastaan, ympäristöön, käyttöjärjestelmiin, web-selaimiin, sähköpostiohjelmiin ja melkein mihin tahansa.

### 6.6. Digital Integrity Protocol for Paper Documents

Jos useampi henkilö joutuu allekirjoittamaan paperimuotoisen sopimuksen, niin ei ole aina käytännöllistä siirtyä koko porukalla samaan paikkaan pelkän allekirjoituksen vuoksi. Jokainen voi sen sijaan allekirjoittaa erillisen allekirjoitussivun. Ne lähetetään postin kautta yhteen paikkaan, jossa niistä ja varsinaisesta sopimustekstistä koostetaan lopullinen sopimus. Joku voisi kuitenkin vaihtaa nidontavaiheessa tai myöhemmin sopimustekstin toiseen. Tämä voitaisiin estää kullekin allekirjoitussivulle lisättävällä seuraavanlaisella digitaalisen tarkistussumman sisältävällä viestiosalla, joka on johdettavissa tarvittaessa pelkästä paperidokumentin tekstistä. Allekirjoitussivulle tulee tietenkin perinteinen kynällä tehtävä allekirjoitus. Näin allekirjoitus pätee vain sopimukseen, joka täsmää tietyllä tavalla muodostettuun tarkistussummaan.

```
-----BEGIN PAPER DOCUMENT INTEGRITY MESSAGE-----
title: Top Secret Government Weapon Delivery Contract 2008-W6-1
character set:
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890
algorithm: SHA-256
pages: 8
block size: 1
block checksums: 927 3BB CE6 640 863 89F F73 227
hash: BFCF7527770B616E7B10367C46F3770D7BFD525CED4902D33D11636CD01894DF
-----END PAPER DOCUMENT INTEGRITY MESSAGE-----
```

Viestiosassa on listattu kaikki huomioitavat merkit eli käytettävä *merkistö* (character set). Lisäksi mainitaan dokumentin *otsikko* (title), käytettävä hash-funktio (algorithm), sivujen lukumäärä (pages), tarkasteltavien lohkojen koko sivuina (block size), kunkin lohkon erillinen tarkistussumma muun muassa OCR (Optical Character Recognition)-tunnistusvirheiden tarkistusta varten (block checksums) ja koko dokumentista tietyllä tapaa muodostettu hash-arvo (hash). Dokumentit luodaan ja tarkistetaan kuvan 6.6 protokollan mukaisesti.

**Document creation process**

1. Sender and recipients write contract document with e.g. Word.
2. Sender e-mails the final document file to each recipient.
3. Each recipient feeds the document file to
DIPPD (Digital Integrity Protocol for Paper Documents) program.
4. Recipient selects or uses default values for *character set*, *algorithm* and *block size*.
5. DIPPD calculates the integrity message block. Other than text content is ignored (Research is needed for image encoding). All types of white spaces, characters not listed in *character set* and integrity messages are ignored. *Block size* means number of pages in each block. The block concept is only used for block checksums. Checksum is calculated independently for each block with the selected *algorithm*. *Hash* is the actual value used for integrity check. It is calculated from all the accepted characters in the whole document.
6. Recipient adds the integrity message to signature page and saves this version.
7. Recipient prints the signature page and signs with (old school) pen.
8. Recipient archives the document file and printed version.
9. Recipient mails the signed paper back to sender or other instance.
10. Sender or other instance checks the signature and integrity on each person's signature page.

**Document integrity check process**

1. The sender has the original document file, so it can be used for calculating integrity messages for all the selected *character set*, *algorithm* and *block size* combinations before archiving the combined final paper contract. For example government offices can use scanner and OCR program to read all the pages. Only text content is needed for DIPPD.
2. Block checksums are checked. In this way it is possible to detect OCR program recognition errors. Errors in invalid pages are corrected e.g. manually and DIPPD is re-run. Integrity messages and signatures are ignored while calculating block checksums or document hash, since they are not part of the original document.
3. DIPPD validates all the integrity messages by calculating document hash for each existing integrity message. Checked integrity messages or at least number of messages is shown. This prevents for example manipulation of integrity message structure, if the person responsible for document validation is accurate.

Kuva 6.6. Digital Integrity Protocol for Paper Documents.

Dokumentti kirjoitetaan perinteisesti esimerkiksi Wordilla. Tiedosto lähetetään kaikille sellaisille tahoille, joiden täytyy allekirjoittaa kyseinen sopimus tai muu asiakirja. He syöttävät tiedoston DIPPD (Digital Integrity Protocol for Paper Documents)-ohjelmalle. Ohjelma käy tiedostosta läpi kaikki tekstiosat. Välilyönnit, rivinvaihdot ja muut vastaavat tyhjät merkit jätetään huomiotta. Vain sellaiset merkit otetaan huomioon, jotka on listattu character set -kentässä. Myöskään mahdollisia olemassa olevia integrity message -viestejä ei oteta mukaan tarkasteluun. Dokumentti käydään läpi ensin määritellyn kokoisissa (block size) lohkoissa ja lasketaan hash-funktiolla (algorithm) kunkin lohkon tarkistussumma. Block checksums -kenttään laitetaan hash-funktion arvoista vain kolme ensimmäistä tavua. Tällöin virheellinen sivu menee läpi tarkistuk-

sesta noin 0,02 prosentin todennäköisyydellä. Lopuksi lasketaan samaan dataan perustuen tarkistussumma koko dokumentille. Tuettavia hash-funktioita voisivat olla esimerkiksi MD5 ja SHA-perhe.

Käyttäjä lisää muodostetun viestilohkon dokumentin henkilökohtaiselle tai yhteiselle allekirjoitussivulle ja tulostaa kyseisen sivun. Hän arkistoi dokumentin digitaalisen ja tulostetun version, ja lähettää allekirjoitetun sivun esimerkiksi postin kautta paikkaan, jossa lopullinen sopimus nidotaan. Vastaanottaja tarkistaa allekirjoitukset ja viestilohkojen oikeellisuuden ennen dokumentin täytäntöönpanoa ja arkistointia. Sillä on todennäköisesti alkuperäinen digitaalinen versio dokumentista, joten allekirjoitussivuilla olevat viestilohkot on mahdollista muodostaa syöttämällä alkuperäinen tiedosto ja kentät character set, algorithm ja block size DIPPD-ohjelmalle. Ohjelman muodostamaa viestilohkoa verrataan tulostettuun versioon.

Kun paperimuotoinen dokumentti tarkistetaan esimerkiksi viranomaisen toimesta, niin se syötetään ensin OCR-ohjelmalle. Sivuja tarkastellaan lohkoissa, joten virheellisen tarkistussumman sisältävät sivut täytyy korjata ensin mahdollisesti manuaalisesti. Kun dokumentti menee ohjelmasta läpi ja viestilohkojen lukumäärä sekä perinteiset allekirjoitukset täsmäävät, niin dokumentti on validi. Ratkaisu vaatii ylimääräisiä resursseja erityisesti tarkistusvaiheessa. Tällaisten ja vastaavien tekniikoiden vieminen ainakin virastoihin ja niiden prosesseihin kestää jopa vuosikymmeniä. Lisäksi kuvia ja taulukoita ei huomioida ratkaisussani. Niiden koodaus vaatii lisätutkimusta.

### 6.7. Improvement to Abloy Key Cards

Abloy-avaimen kombinaatiosta on mahdollista johtaa taulukon perusteella peitekoodi, jota tarvitaan avainkopion teettämisessä liikkeessä [Fagerlund, 2007]. Tällaisia taulukoita on ainakin Abloyn valtuuttamilla lukkoliikkeillä, ja sellainen voidaan johtaa myös joukosta lukkojen mukana tulevia avainkortteja. Näin rikollisten on mahdollista teettää valtuutetussa liikkeessä avain vain valokuvan perusteella. Ongelma voitaisiin korjata kryptologian avulla. Alla on ehdotelma uudesta avainkorttimallista, jonka sisältämät korostetut tietokentät peitetään samalla tekniikalla kuin perinteisissä Veikkauksen arvoissa. Harvempi lukon ostaja kun lopulta tilaa uusia avainkopioita, ja luvatta raaputetut tietokentät tarkoittavat lukon vaarantumista. Avainkortin säilyttäminen turvallisesti muilta piilossa on aina oma ongelmansa.

```
Abloy Protec Key Card

Date issued: 2008-05-25
Protec key code: 5344257362P05
Secret key ID: 703928
Ciphertext half-block: ddfd31357c783f0dedd73df1376702ae
```

Kaikki uudet avainkopiot valmistetaan mieluiten tehtaalla, tai varmistetaan sieltä vähintään avainkortin oikeellisuus. Kortin luonnissa kaikki satunnaisuus kerätään laitteistolla toteutetuista satunnaislukugeneraattoreista. Aluksi valitaan aidosti satunnainen 256-bittinen lohkosalaajan salausavain, jolla on satunnaisesti valittu kuusinumeroinen päiväkohtainen ID. Esimerkissäni yhden päivän aikana voidaan siis valmistaa enintään miljoona lukkoa eri sarjoituksella. Jos ID on jo käytössä, niin valitaan uusi satunnainen ID. Muutaman epäonnistuneen yrityskerran jälkeen etsitään iteratiivisesti seuraava vapaa ID. Näin ei voida päätellä, montako lukkoa tehdas valmistaa päivässä.

Seuraavaksi salataan lohkosalaajalla avaimen kombinaatio eli Protec key code ja satunnaisesti valitut täytebitit (padding) tuolla salausavaimella. Täytebittejä tarvitaan sen verran, että yksi salausalgoritmin lohko saadaan täyteen. Lopputulos eli salakieliteksti jaetaan kahteen osaan, joista ensimmäinen on ciphertext half-block. Protec key code muodostuu kymmenestä numerosta välillä 1 - 7. Avainkoodissa on oltava lukon tyyppi, ja tässä tapauksessa P merkitsee Protec-lukkoa. Loput kaksi kirjainta tarkoittavat Protec-lukon tarkempaa tyyppiä.

Valmistuksen aikana tehtaan tietokantaan tallennetaan tiedot date issued, secret key ID, secret key, padding ja epäonnistuneiden tilauskertojen lukumäärä, joka on tietenkin aluksi nolla. Jokaisella lukolla on siis oma salausavaimen ID, salausavain ja täytebitit. Tehtaan tietokantaan ei jää tietoa lukon tyypistä tai kombinaatiosta, mikä olisi turvallisuusriski. Kun asiakas tilaa liikkeestä uutta avainta, niin kortin sisältö lähetetään salattuna tehtaalle. Esimerkiksi OCR (Optical Character Recognition)-ohjelmalla luetaan kortin sisältö joko liikkeessä tai tehtaalla. Korttiin voidaan lisätä vielä kenttä kortin tekstistä muodostettua tarkistussummaa varten, jolla havaitaan mahdolliset OCR-lukuvirheet.

Seuraavaksi etsitään tietokannasta tietyn päivän tietty salausavain ja vastaava padding-data. Lisäksi tarkistetaan epäonnistuneiden tilauskertojen lukumäärä. Salaus suoritetaan uudestaan samalla tavalla ja verrataan liikkeen lähettämään puolikkaaseen salakielitekstiin. Jos ne täsmäävät, niin luodaan key codessa mainittu oikeantyyppinen avain, joka lähetetään liikkeeseen. Jotta hyökkääjä voisi teettää kortin avulla eri avaimen, niin hänen täytyisi murtaa 256-bittinen lohkosalaaja. Silloinkin mahdollisia toimivia salausavaimia on

parhaimmillaan 2 ^ 128, koska asiakkaan avainkortissa on vain puolet salakielitekstin sisällöstä. Hyökkääjä ei siis saa selville oikeaa salausavainta, jolla voitaisiin luoda toimiva salakieliteksti toiselle avainkombinaatiolle. Lohkosalaajan sijaan voidaan käyttää myös hash-funktiota.

Lisäturvaa saadaan vielä käyttämällä digitaalista allekirjoitusta. Allekirjoitus suoritetaan tehtaalla siihen tarkoitukseen pyhitetyllä tietokoneella tai suojatulla laitteella (black box). Tein tätä varten GnuPG:llä 3072-bittisen DSA (Digital Signature Algorithm)-avainparin nimeltään ”Abloy Key Card Signing Key 2008-05”, jota ei voi käyttää tiedon salaukseen. Sillä allekirjoitetaan jokainen tehtaalta yhden kuukauden aikana lähtevä avainkortti. Jopa jokaiselle kuukaudelle voidaan siis luoda erillinen avainpari. Käytin tavallista suurempaa DSA-avainta, koska PGP-toteutuksen ei tarvitse olla yhteensopiva muiden kuin tehtaan määrittelemien toteutusten kanssa. Alla on käyttämäni komennot avainparin luontiin, allekirjoitukseen, allekirjoituksen tarkistukseen ja julkisen avaimen tallentamiseen tiedostoon abloy200805.asc, joka voidaan toimittaa tarvittaessa myös lukkoliikkeille.

```
gpg --gen-key --enable-dsa2
gpg --local-user "2008-05" --clearsign message.txt
gpg --verify message.txt.asc
gpg --armor --output abloy200805.asc --export "2008-05"

-----BEGIN PGP SIGNED MESSAGE-----
Hash: SHA256

Abloy Protec Key Card

Date issued: 2008-05-25
Protec key code: 5344257362P05
Secret key ID: 703928
Ciphertext half-block: ddfd31357c783f0dedd73df1376702ae
-----BEGIN PGP SIGNATURE-----
Version: GnuPG v1.4.9 (MingW32)

iF4EAREIAAYFAkg5scUACgkQ3YH5T/TKEmvwlwEAmAFkISopBxm9WnG9Lujd58HU
o9dljTvrPaI92cWdAnQBAIzGUlaM7ELDmTGzhdiyCAXh21Q9p/hAOS08iptZS/zR
=miUW
-----END PGP SIGNATURE-----
```

Kun kuukausi on vaihtunut, niin vanha salainen avain voidaan tuhota. Avainkorttien valmistus kannattaa tehdä tehtaalla eri tietokoneella kuin niiden tarkistus jo sen vuoksi, että muun muassa PGP-toteutuksessa voi olla tietoturva-aukkoja. Komentojen alla on edellisen esimerkin allekirjoitettu versio. Jos siitä jätetään pois secret key ID ja puolikas salakieliteksti, niin on mahdollista tilata tehtaalta vääriä avaimia, jos yksikin DSA-avain saadaan murrettua tulevaisuudessa.

Ratkaisuni tarvitsee tallennustilaa käytännössä täysin lineaarisesti luotuihin sarjoituksiin nähden. Kaikki työvaiheet on mahdollista suorittaa rinnakkain, joten skaalautuvuuden kanssa ei pitäisi tulla ongelmia. Protokolla tukee myös kaikkia vanhoja, nykyisiä ja tulevia avainjärjestelmiä. Siitä ei ole myöskään haittaa, vaikka uuden avaintyypin avainavaruus kasvaisi merkittävästi. Toisin kävisi esimerkiksi tehtaan palvelimella sijaitsevan avaintyyppikohtaisen salaisen hakutaulun kanssa, jossa olisi jokaista avainkombinaatiota vastaava satunnainen vastearvo.

Toteutuksessa voidaan käyttää relaatiotietokannan sijaan jopa tiedosto-pohjaista tietokantaa, vaikka toki relaatiotietokannasta saadaan tarvittaessa halutut tiedot ulos tiedostoina. Avainkorttien luonnissa ei tarvitse tehdä lain-kaan hakuja, ja jokaisen päivän tiedot pysyvät omissa tiedostoissaan. Tiedostot voidaan arkistoida ja siirtää tarkistuskoneelle yksitellen. Hakemistohierarkia on varsin yksinkertainen. Jokaisella vuodella on oma hakemisto, jonka alla on kuukausittainen hakemisto. Sen alla on puolestaan kaikki kyseisen kuukauden päiväkohtaiset tiedostot. Tiedostoista voidaan ottaa helposti hash-arvoja ja allekirjoittaa lista digitaalisesti. Näin tarkistuskoneen sisältämien tiedostojen eheys voidaan tarkistaa ajoittain manipuloinnin ja virheiden varalta.

Tilantarve putoaa käytännössä nollaan ja toteutus yksinkertaistuu, jos kaikkien avainkorttien luontiin käytetään vain yhtä esimerkiksi kuukausittain vaihtuvaa lohkosalaajan avainta sekä allekirjoitusavainta. Silloin kortissa ei tarvita kenttää secret key ID. 256-bittistä salausta ei saada todennäköisesti murrettua, vaikka vähintäänkin brute force -hyökkäys on mahdollista suorittaa kahden samalla salausavaimella luodun avainkortin perusteella. Puolikas sala-kieliteksti ei vaikeuta tällöin kryptoanalyysia oikeastaan lainkaan. Hyökkääjän täytyisi murtaa lisäksi DSA-avain. On helpompaa opiskella lukkosepäksi ja hankkia laitteet avainten sorvaamiseen tai etsiä epävirallinen lukkoliike.

# 7. Yhteenveto

Tietoturva on kokonaisuus, jossa yksikin heikkous jollain osa-alueella vaarantaa kaiken. Fyysistä turvallisuutta kuten turvallisia lukkoja väheksytään edelleen liikaa ohjelmistoalalla. Suurin osa yritysten tietokoneista on alttiita tiedustelupalveluiden manipuloinnille, vaikka niillä käsitellään usein varsin salaistakin tietoa. Toisin sanoen luotetaan liikaa kiintolevysalaukseen ja muihin ohjelmallisiin turvatekniikoihin, vaikka näppäimistöä voi manipuloida jopa kotikonstein keräämään kaikki näppäinpainallukset kuten sen kiintolevysalauksen salasanan. Näin ulkomaisen tiedustelupalvelun värväämä koti- tai ulkomainen työntekijä onnistuu vaarantamaan koko yrityksen. Tämä on maailmanlaajuinen ongelma.

Tavallinen työpaikan mikrotietokone ja sen kuvaputki- tai litteä näyttö sekä esimerkiksi DVI-kaapeli ovat varsinaisia radiolähettimiä. Muutaman tuhannen euron eli tiedustelun kannalta ilmaisella laitteistolla voidaan saada selville tietoa miljoonien eurojen arvosta. Jopa tavallisen langallisen näppäimistön johdin tai muu elektroninen toteutus saattaa olla altis TEMPEST-hyökkäykselle. Huolestuttavaa on myös se, että kuluttajilla ja yrityksillä on melkein pakkomielle muuttaa kaikki laitteet langattomiksi. Tämä on tiedustelun kannalta kuin kävelisi Napolin kaduilla kädessään aito timanteilla koristeltu platina-Rolex. Osa langattomista toteutuksista alkaa toki olla ehkä riittävän turvallisia kehittyneempien salaus- ja VPN-tekniikoiden vuoksi.

Myöskään vahva salausalgoritmi ei takaa turvallisuutta, jos sitä käytetään väärällä tavalla. Se voi itse asiassa helpottaa hyökkäystä merkittävästi, kun ohjelmistokehittäjälle tai esimerkiksi palvelimien ylläpitäjälle tulee väärä turvallisuuden tunne. Siksi kryptologisia protokollia tai varsinkaan algoritmeja ei tulisi koskaan kehittää tavallisen ohjelmistokehittäjän roolissa, jos ei ole erityisesti perehtynyt aihepiiriin. Nimittäin maailman parhaat kryptologian asiantuntijatkin tekevät tosissaan töitä vuodesta toiseen, ja silti joku saattaa murtaa algoritmin vuosien päästä. Salausalgoritmeja käyttävistä protokollistakin löytyy toisinaan haavoittuvuuksia vasta vuosien päästä niiden käyttöönotosta. Salausalgoritmien käytössä on sudenkuoppia, joihin täytyy kiinnittää aina erityistä huomiota. Muuten algoritmin murtaminen muuttuu nopeasti todella vaikeasta tehtävästä kryptoanalyysin tai siihen liittyvän matematiikan kannalta melkein triviaaliksi tehtäväksi.

Tiedustelupalvelu ottaa käyttöön kryptoanalyysin vasta sitten, kun tiedustelu luokittelee tiedon kiinnostavaksi, ja mitään muita keinoja ei löydy. Riittävän heikko kryptologinen algoritmi tai protokolla voidaan myös murtaa ilman suurempaa pohdintaa resurssien käytöstä, jos kohdehenkilön tai yrityksen pro-

fiili on kiinnostava. Nykyisiä GSM-puheluita, tekstiviestejä tai GPRS-yhteyksiä ei voi pitää turvallisena ilman VPN:ää tai erillistä puheen salaavaa tekniikkaa, joka on yleensä puhelimeen kiinnitettävä erillinen laite. Puhelut voidaan kerätä langattomasti, ja A5/1- sekä A5/2-jonosalaajia ei pidetä erityisen turvallisina. GSM-protokollista löytyneet heikkoudet [Barkan *et al.*, 2006] mahdollistavat salauksen ohittamisen. Näin hyökkääjä voi mahdollisesti kaapata puhelun tai muuttaa esimerkiksi GPRS-yhteyden yli kulkevaa web-liikennettä ja saastuttaa näin käyttäjän koneen selaimen tietoturva-aukon kautta, jos VPN-toteutusta ei käytetä. Krakkerit ovat aina erityisen kiinnostuneita murtautumistavoista, joista ei jää käytännössä lainkaan todisteita kuten IP-osoitteita.

Yleensä jokin muu heikkous kuten ohjelmakoodin tai kryptologiaa käyttävän protokollan tietoturva-aukot, ihmisten virheet, epätietoisuus ja epäsatunnaiset salasanat mahdollistavat käytettävistä resursseista riippumatta suuremmat mahdollisuudet tietomurron onnistumiselle. Salauksessa käytettävät heikosti satunnaisuutta sisältävät salasanat rikkovat melkein aina salauksen turvallisuuden kuin vasara kristallimaljakon. 256-bittinen salausalgoritmi muuttuu näin hetkessä jopa 20-bittiseksi. Onko salasanatietokantasi 20-bittisen salauksen takana? En ehdi näpsäyttää edes sormia ennen kuin se on murrettu tavallisen sanakirjahyökkäyksen avulla.

Ihmiset eivät oikeasti muista useita pitkiä aidosti satunnaisia salasanoja, joten siihen ongelmaan tarvitaan uutta tekniikkaa. Esitin tähän ratkaisuna muun muassa erillisellä laitteella toteutetun salasanatietokannan. Ohjelmistopohjainen salasanatietokanta on kohtuullinen kompromissi, kunhan sen salasana on turvallinen. Biometriset tunnistusmenetelmät taas soveltuvat paremmin varsinaiseen käyttäjän tunnistamiseen eli mallien vertailuun. Salasanojen tai salausavainten luontiin ne eivät sovellu, koska käyttäjän ominaisuuksia ei voi pitää salaisuuksina. Muihin ongelmiin esitin kuusi ratkaisua, jotka ovat toteutettavissa ja enemmän tai vähemmän käytännöllisiä.

Murtautuja etsii yleensä aina helpoimman tien kiertää järjestelmän suojaukset, joten sen jokaisen osa-alueen täytyy olla tasapainossa. Tämän vuoksi ohjelmistoturvallisuutta kannattaa edelleen painottaa, sillä verkon yli murtautuminen on edullista, ja siinä on pienempi todennäköisyys jäädä kiinni. Melkein kaiken tyyppiset ohjelmistot ja ohjelmakoodia sisältävät laitteet ovat sisältäneet tietoturva-aukkoja. Muutamana esimerkkinä käyvät kalliitkin langalliset tai langattomat tietoliikennelaitteet, palomuurit, virustorjuntaohjelmistot, toimisto-ohjelmistot, web-selaimet, sähköpostiohjelmat, pikaviestiohjelmat, vertaisverkko-ohjelmat, tietokannat, käyttöjärjestelmät, mediasoittimet kuten Winamp, pakkausohjelmat kuten WinZip ja 7-Zip, Java-virtuaalikoneet, web-selainten plug-init, murtautumisyrityksiä havaitsevat ohjelmat kuten

Snort, tietoliikennettä kaappaavat ohjelmat kuten tcpdump ja Ethereal, laiteajurit, nimipalvelimet, web-palvelimet, käyttäjätunnistukseen tarkoitetut palvelimet, matkapuhelimet, pelikonsolit, kopiosuojaukset, VPN-toteutukset ja SSH-toteutukset.

Kun lähetin ensimmäisen kerran sähköpostia BBS-purkista Supra Fax Modem V.32bis -modeemillani, niin Internet oli lähinnä vain yliopistojen välistä tietoliikennettä. Tuo laite siirsi tietoa enintään 14400 bittiä eli 0,0018 megatavua sekunnissa. Toisin sanoen nykyisten keskimäärin todella huonosti optimoitujen web-sivun avaamiseen menisi helposti 5 - 10 minuuttia sivua kohden. Yhden yksikerroksisen Blu-ray-levyn kokoisen elokuvan siirtämiseen sellaisenaan ilman uudelleenpakkausta MPEG2:sta tehokkaammilla koodekeilla olisi vienyt minuuttiveloitukseen perustuvalla laskutuksella vähintään 160 päivää. Tietoturvasta olivat huolissaan lähinnä BBS-purkkien ylläpitäjät, kun kaikki palvelimen käyttäjät olivat hakkereita tai niitä koulujen nörteimpiä minä mukaan lukien. Suurin osa tietokoneista oli eristetty täysin tai ainakin suurimman osan ajasta kaikista tietoverkoista. Web ja verkkorikollisuus olivat tuntemattomia käsitteitä.

Nykyään tilanne on kirjaimellisesti kuin toiselta vuosituhannelta. Suuri osa kansasta käyttää Internetiä, melkein kaikki tietokoneet ja suuri osa muista laitteista digiboksista lähtien ovat verkossa, ja vain harva oikeasti tietää, miten haavoittuvia heidän järjestelmänsä todella ovat. Krakkereilla on varsinaiset kissanpäivät. Aiempaan verrattuna on niin paljon enemmän erilaisia keinoja murtautua tietokoneisiin. Jopa Nothcutin ja muiden [2001] kirjan jälkeen hyökkäystapojen määrä on kasvanut merkittävästi. Krakkerit voivat kerätä puoliautomaattisesti maailmanlaajuisesti Internetistä suuren joukon satunnaisista IP-osoitteista löydettyjä orjakoneita, joilla voi esimerkiksi lähettää roskapostia, murtaa salauksia, murtautua paremmin suojattuihin viranomaisten tietojärjestelmiin, aiheuttaa palvelunestohyökkäyksiä, rikkoa tiedostojen eheyttä, kerätä käyttäjän salasanat sähköpostiin ja muualle, naureskella käyttäjän hassuille intiimeille valokuville ja niin edelleen. Usein he ovat vain uteliaita, pitävät haasteista ja haluavat kunniaa. He eivät ole aina pahantahtoisia kuten rikolliset.

Tavallisissa tietomurroissa yksittäiset taitavat krakkerit ja krakkeriryhmät ovat täydellinen referenssi. Varsinaisilla osaajilla on todella tarkkaa tietämystä järjestelmien toiminnasta ja niiden heikkouksista. Kadulta satunnaisesti valittu henkilö ei ymmärtäisi oikeastaan mitään, jos tämä guru selittäisi hänelle tiettyyn aliverkkoon kohdistuvan hyökkäyksen kaikkine vaiheineen TCP/IP-pinojen hienovaraisesta tiedustelusta lähtien.

Tutkielmani käsitteli melko paljon tiedustelupalveluita. Otin tällaisen näkökulman, koska niillä voidaan olettaa olevan käytössä kehittyneimmät tekniikat ja suuret resurssit. Lisäksi kryptoanalyysia käsittelevässä kirjallisuudessa erityisesti NSA:lla oletetaan olevan massiivinen määrä erikoispiireillä toteutettua laskentakapasiteettia ja salaista tietämystä kryptoanalyysista. Näitä organisaatioita voi pitää teoreettisesti suurimpana uhkana, mutta ne eivät voi monestakaan kuten poliittisesta syystä käyttää hyväkseen tai ainakaan julkaista suurinta osaa keräämästään tiedosta. Se tieto on itse asiassa huippusalaista, vaikka kyseessä olisi tiedustelun kannalta arvottomasta tiedosta kuten nakkikioskin kirjanpidosta. Kyse on myös siitä tasapainosta, että suurvallat eivät saa toisiinsa nähden liian suurta teknistä etumatkaa. Kyseiset organisaatiot ovat kuitenkin täydellinen referenssi kryptoanalyysissa. Jos et kehitä uutta ohjusten ohjausjärjestelmää, niin vakoojia pomppii enintään joka toisessa pusikossa. Jos laboratoriosi kirjahyllyllä on krytron-putkia sekä kullalla päällystettyjä plutoniumpalloja, ja olet keksinyt teoreettisesti aiempaa huomattavasti tehokkaammalla periaatteella toimivan ydinaseen, niin heitä on rivissä vähintään tusina jokaisen puunrungon takana.

Tietoturvan ongelmista on tietenkin turha huolestua liikaa. Niiden ja muiden ongelmien kanssa täytyy elää, koska täydellistä turvallisuutta ei ole mahdollista saavuttaa. Suomi on tällä hetkellä varsin turvallinen ja hyvä paikka elää verrattuna hyvin moneen muuhun maahan tai ajanjaksoon. Ihmiskunnalla on suurempiakin ongelmia kuten ympäristön saastuminen, eläin- ja kasvilajien sukupuutto, aavikoituminen, pula puhtaasta vedestä, laivojen painolastivesien mukana kulkeutuvat eliöt, jatkuvan väestönkasvun mukana lisääntyvät taudit, nälkä, köyhyys, sodat, väkivalta, sekä öljyn ja muiden raaka-aineiden saatavuus. Ongelmat ruokkivat toisiaan.

Öljy on vain yksi niistä raaka-aineista, joiden louhiminen maasta tulee lopulta aivan liian kalliiksi. Esimerkiksi metallien louhiminen vaatii epälineaarisesti yhä enemmän energiaa, kun malmin metallipitoisuus heikkenee. Joka tuutissa korostettu ilmaston lämpeneminen ja otsonikato ovat vasta alkua ympäristöongelmille. Haluammeko vaikeuttaa tulevien sukupolvien elämää merkittävästi? Mitähän mieltä he ovat meidän järjettömistä ostoskeskuksista, suurista omakotitaloista tai ylimitoitetuista asunnoista, kesämökeistä, kertakäyttökulttuurista, taulutelevisioista, sisustamisesta, mainoslehtisistä, Kaukoidän matkoista, pihvintuotannosta ja henkilöautoilusta? Ainut käytännön ratkaisu on kohtuullinen kulutustaso. En ole ainakaan itse joutunut luopumaan todellisuudessa oikeastaan mistään, vaikka en omista esimerkiksi suurta asuntoa tai autoa. Päinvastoin.

## Viiteluettelo

[Abloy, 2008] Abloy, Abloy Protec. 2008. Saatavana osoitteesta *http://www.abloy.fi/* -> tuotteet -> ABLOY-esitteet.

[Aboba et al., 2004] Bernard Aboba, Larry J. Blunk, John R. Vollbrecht, James Carlson and Henrik Levkowetz (ed.), Extensible authentication protocol (EAP). Request for Comments (RFC) 3748. 2004. Available as *http://tools.ietf.org/html/rfc3748*.

[Adelstein, 2006] Frank Adelstein, Live forensics: diagnosing your system without killing it first. *Comm. ACM* **49**, 2 (February 2006), 63-66.

[Ahn et al., 2004] Luis von Ahn, Manuel Blum and John Langford, Telling humans and computers apart automatically. *Comm. ACM* **47**, 2 (February 2004), 56-60.

[Almgren and Lindqvist, 2001] Magnus Almgren and Ulf Lindqvist, Application integrated data collection for security monitoring. *From Recent Advances in Intrusion Detection (RAID 2001)*, Davis, California, October 2001, 22-36. Available as *http://www.sdl.sri.com/papers/raid2001/*.

[Anderson and Kuhn, 1996] Ross J. Anderson and Markus G. Kuhn, Tamper resistance - a cautionary note. *The second USENIX workshop on electronic commerce proceedings*, California, 1996. Available as *http://www.cl.cam.ac.uk/~mgk25/tamper.pdf*.

[Anderson and Kuhn, 1997] Ross J. Anderson and Markus G. Kuhn, Low cost attacks on tamper resistant devices. *In M. Lomas et al. (ed.), Security Protocols, 5th international workshop, Proceedings*, France, 1997. Available as *http://www.cl.cam.ac.uk/~mgk25/tamper2.pdf*.

[Barkan et al., 2006] Elad Barkan, Eli Biham and Nathan Keller, Instant ciphertext-only cryptanalysis of GSM encrypted communication. Israel Institute of Technology, 2006. Available as *http://cryptome.org/gsm-crack-bbk.pdf*.

[Benson, 2008] Robert L. Benson, The Venona story. National Security Agency, 2008. Available as *http://www.nsa.gov/publications/publi00039.cfm*.

[Black, 2008] Paul E. Black (ed.) et al., Dictionary of algorithms and data structures. National Institute of Standards and Technology, 2008. Available as *http://www.nist.gov/dads/*.

[Blaze, 2003] Matt Blaze, Notes on picking pin tumbler locks. University of Pennsylvania, 2003. Available as *http://www.crypto.com/papers/notes/picking/*.

[Borisov et al., 2008] Nikita Borisov, Ian Goldberg and David Wagner, Security of the WEP algorithm. ISAAC research group, University of California, 2008. Available as *http://www.isaac.cs.berkeley.edu/isaac/wep-faq.html*.

[CERT-FI, 2007] CERT-FI, Varoitus 07/2007: Suomalaisia verkkopalveluiden käyttäjätunnuksia sisältävä tiedosto verkossa. CERT-FI, 2007. Saatavana osoitteesta *http://www.cert.fi/varoitukset/2007/varoitus-2007-7.html*.

[Chen, 2008] Elaine Chen, The best of both worlds, in a Cliq. Assa Abloy Future Labs, 2008. Available as *http://www.assaabloyfuturelab.com/FutureLab/Templates/Page2Cols____1621.aspx*.

[COPACOBANA, 2008] COPACOBANA, A codebreaker for DES and other ciphers, Ruhr University of Bochum and Christian-Albrechts University Kiel, 2008. Available as *http://www.copacobana.org/*.

[Crawford, 2002] William Crawford, Six JDBC tips for enterprise web applications. *Oracle Magazine*, September/October 2002.

[Cryptography Research, 2003] Cryptography Research, Evaluation of VIA C3 Nemeniah random number generator. 2003. Available as *http://www.via.com.tw/en/downloads/whitepapers/initiatives/padlock/evaluation_padlock_rng.pdf*.

[Daemen and Rijmen, 1999] Joan Daemen and Vincent Rijmen, The Rijndael block cipher. Document version 2, 1999. Available as *http://www.iaik.tu-graz.ac.at/research/krypto/AES/old/~rijmen/rijndael/rijndaeldocV2.zip*.

[Davida et al., 1998] George I. Davida, Yair Frankel, Brian J. Matt and René Peralta, On the relation of error correction and cryptography to an off line biometric based identification scheme, 1998. Available as *http://citeseer.ist.psu.edu/davida99relation.html*.

[Dorrendorf et al., 2007] Leo Dorrendorf, Zvi Gutterman and Benny Pinkas, Cryptanalysis of the random number generator of the Windows operating system. The Hebrew University of Jerusalem and University of Haifa, 2007. Available as *http://eprint.iacr.org/2007/419.pdf*.

[Dunker, 2003] Mary Dunker, Don't blink: iris recognition for biometric identification. SANS Institute, SANS Security Essentials, November 2003. Available as *https://www2.sans.org/reading_room/whitepapers/authentication/1341.php*.

[Dword and Naor, 1992] Cynthia Dword and Moni Naor, Pricing via processing or combating junk mail. *Proceedings of the 12th annual international cryptology conference on advances in cryptology* (1992), 139-147. Available as *http://www.wisdom.weizmann.ac.il/~naor/PAPERS/pvp.ps*.

[Eastlake et al., 2005] Donald E. Eastlake, Jeffrey I. Schiller and Steve Crocker, Randomness requirements for security. Request for Comments (RFC) 4086, 2005. Available as *http://rfc.sunsite.dk/rfc/rfc4086.html*.

[Fagerlund, 2007] Jaakko Fagerlund, Abloy avainten peitekoodit. Haittalevy blogi, 2007. Saatavana osoitteesta *http://haittalevy.blogspot.com/2007/12/abloy-avainten-peitekoodit.html*.

[Felten and Schneider, 2000] Edward W. Felten and Michael A. Schneider, Timing attacks on web privacy. *Proceedings of the 7th ACM conference on Computer and communications security* (2000), 25-32.

[Fenzi and Wreski, 2004] Kevin Fenzi and Dave Wreski, Linux security howto. 2004. Available as *http://www.linuxsecurity.com/docs/LDP/Security-HOWTO/*.

[Forrest et al., 1997] Stephanie Forrest, Steven A. Hofmeyr and Anil Somayaji, Computer immunology. *Comm. ACM* **40**, 10 (October 1997), 88-96.

[Foster et al., 1998] Ian Foster, Carl Kesselman, Gene Tsudik and Steven Tuecke, A security architecture for computational grids. *Proceedings of the 5th ACM conference on Computer and communications security* (1998), 83-92.

[Fregonese et al., 1999] Giulio Fregonese, Alessandro Zorer and Giovanni Cortese, Architectural framework modeling in telecommunication domain. *Proceedings of the 1999 international conference on Software engineering*, 526-534.

[Friedl, 2007] Stephen J. Friedl, SQL injection attacks by example. 2007. Available as *http://www.unixwiz.net/techtips/sql-injection.html*.

[Gamma et al., 1994] Erich Gamma, Richard Helm, Ralph Johnson and Johs Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison Wesley, 1994.

[Ghosh, 1998] Anup K. Ghosh, *E-commerce Security: Weak Links, Best Defenses*. John Wiley & Sons, 1998.

[Gleni and Petratos, 2004] Sofia Gleni and Panagiotis Petratos, DNA smart card for financial transactions. *ACM Crossroads*, Issue **11.1**, Fall 2004.

[Halevi and Krawczyk, 1998] Shai Halevi and Hugo Krawczyk, Public-key cryptography and password protocols. *Proceedings of the 5th ACM conference on Computer and communications security* (1998), 122-131.

[Haller, 1995] N. Haller, The S/KEY one-time password system. Request for Comments (RFC) 1760, 1995. Available as *http://www.ietf.org/rfc/rfc1760.txt*.

[Haller et al., 1998] N. Haller, C. Metz, P. Nesser and M. Straw, A one-time password system. Request for Comments (RFC) 2289, 1998. Available as *http://www.ietf.org/rfc/rfc2289.txt*.

[Hämäläinen ja Mäkikangas, 2002] Pertti Hämäläinen ja Tero Mäkikangas, Kuinka tunnistaa käyttäjä? *Tietokone*, **6-7** (2002), 93-98.

[IBM, 2003] IBM, Firewalls and demilitarized zone configurations. IBM, 2003. Available as

*http://publib.boulder.ibm.com/infocenter/wsphelp/index.jsp?topic=/com.ibm.websphere.nd.doc/info/ae/ae/cins_firewall.html*.

[Interpol, 2000] Interpol, Method for fingerprint identification. Interpol European Expert Group on Fingerprint Identification - IEEGFI, 2000. Available as *http://www.interpol.int/public/forensic/fingerprints/workingparties/ieegfi/ieegfi.asp*.

[Interpol, 2004] Interpol, Method for fingerprint identification, part II. Interpol European Expert Group on Fingerprint Identification - IEEGFI, 2004. Available as *http://www.interpol.int/public/forensic/fingerprints/workingparties/ieegfi2/*.

[Jonsson and Kaliski, 2003] Jakob Jonsson and Burt Kaliski, Public-key cryptography standards (PKCS) #1: RSA cryptography specifications version 2.1. Request for Comments (RFC) 3447. RSA Laboratories, 2003. Available as *http://tools.ietf.org/html/rfc3447*.

[Kudo and Hada, 2000] Michiharu Kudo and Satoshi Hada, XML document security based on provisional authorization. *Proceedings of the 7th ACM conference on Computer and communications security* (2000), 87-96.

[Kuhn, 2003] Markus G. Kuhn, Compromising emanations: eavesdropping risks of computer displays. University of Cambridge, Computer Laboratory, Report **577**, December 2003. Available as *http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-577.pdf*.

[Liu and Silverman, 2001] Simon Liu and Mark Silverman, A practical guide to biometric security technology. IEEE Computer Society, *IT Professional*, vol. **3**, no. 1, 2001.

[Low, 1998] Douglas Low, Protecting Java code via code obfuscation. *ACM Crossroads*, Spring 1998. Available as *http://www.cs.arizona.edu/~collberg/Research/Students/DouglasLow/obfuscation.html*.

[McGraw and Viega, 2000] Gary McGraw and John Viega, Protecting passwords: part 1 & 2. IBM DeveloperWorks, 2000. Available as *http://www-106.ibm.com/developerworks/library/s-pass1/* and *http://www-106.ibm.com/developerworks/library/s-pass2/*.

[Monrose et al., 1999] Fabian Monrose, Michael K. Reiter and Susanne Wetzel, Password hardening based on keystroke dynamics. *Proceedings of the 6th ACM conference on Computer and communications security* (1999), 73-82. Available as *http://www.cs.jhu.edu/~fabian/papers/acm.ccs6.pdf*.

[Mori and Malik, 2008] Greg Mori and Jitendra Malik, Breaking a visual CAPTCHA. UC Berkeley Computer Vision Group and Simon Fraser University, 2008. Available as *http://www.cs.sfu.ca/~mori/research/gimpy/*.

[Naor, 1996] Moni Naor, Verification of a human in the loop or identification via the Turing test. September 13th, 1996. Available as *http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.pdf*.

[NIST, 1999] National Institute of Standards and Technology, Data Encryption Standard (DES). Federal Information Processing Standards Publications (FIPS PUBS), reaffirmed publication 46-3, 1999. Available as *http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf*.

[NIST, 2001] National Institute of Standards and Technology, Advanced Encryption Standard (AES). Federal Information Processing Standards Publications (FIPS PUBS), publication 197, 2001. Available as *http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf*.

[NIST, 2002] National Institute of Standards and Technology, Secure hash standard. Federal Information Processing Standards Publications (FIPS PUBS), publication 180-2, 2002. Available as *http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf.*

[NIST, 2008] National Institute of Standards and Technology, National and international biometric standards, development bodies and published standards. The Biometrics Resource Center Website, 2008. Available as *http://www.itl.nist.gov/div893/biometrics/standards.html*.

[Northcutt et al., 2001] Stephen Northcutt, Judy Novak and Donald McLachlan, *Network Intrusion Detection An Analyst's Handbook*. Second edition. New Riders Publishing, 2001. Finnish language edition is also available named *Verkkomurtojen havaitseminen*.

[NSA, 1982] National Security Agency, NACSIM 5000, TEMPEST fundamentals. National Security Agency, 1982. Available as *http://cryptome.sabotage.org/nacsim-5000.htm*.

[Oechslin, 2003] Philippe Oechslin, Making a faster cryptanalytic time-memory trade-off. *Advances in Cryptology - CRYPTO 2003, 23rd Annual International Cryptology Conference,* 2003. Available as *http://lasecwww.epfl.ch/~oechslin/publications/crypto03.pdf*.

[OUSPG, 2008] OUSPG, PROTOS Genome test suite c10-archive. Oulu University Secure Programming Group, 2008. Available as *http://www.ee.oulu.fi/research/ouspg/protos/testing/c10/archive/*.

[Oxenhandler, 2003] Daniel Oxenhandler, Designing a secure local area network. SANS Institute, 2003. Available as *http://www.sans.org/reading_room/whitepapers/bestprac/*.

[Palahniuk, 1996] Chuck Palahniuk, *Fight Club*. W. W. Norton & Company, 1996.

[Pinkas and Sander, 2002] Benny Pinkas and Tomas Sander, Securing passwords against dictionary attacks. *Proceedings of the ACM Computer and Communications Security Conference* (2002). 161-170. Available as *http://www.pinkas.net/PAPERS/pwdweb.pdf*.

[Raymond, 2003] Eric S. Raymond, The Jargon File. Version 4.4.7, 2003. The newest version is available as *http://www.catb.org/jargon/*. Printed version named *The New Hacker's Dictionary* (Third Edition, 1996) is available from MIT Press.

[Rigney et al., 2000] Carl Rigney, Allan C. Rubens, William Allen Simpson and Steve Willens, Remote Authentication Dial in User Service (RADIUS). Request for Comments (RFC) 2865, 2000. Available as *http://tools.ietf.org/html/rfc2865.*

[Rivest, 1992] Ron Rivest, The MD5 message-digest algorithm. MIT Laboratory for Computer Science and RSA Data Security, Inc., Network Working Group, Request for Comments (RFC) 1321, 1992. Available as *http://www.ietf.org/rfc/rfc1321.txt*.

[SANS Institute, 2007] SANS Institute, Top-20 2007 security risks, annual update. SANS Institute, 2007. Available as *http://www.sans.org/top20/*.

[Santala, 2001] Niko Santala, Biometriset tunnistusmenetelmät. Tietoturvallisuuden erityiskysymyksiä, Juhani Paavilainen ja Marko Helenius (toim.), Tampereen yliopisto, Tietojenkäsittelytieteiden laitos, Raportti **B-2001-8**, 132-147, 2001.

[Schneier, 1996] Bruce Schneier, *Applied Cryptography.* Second edition. John Wiley & Sons, 1996.

[Schneier, 1999] Bruce Schneier, Inside Risks: The uses and abuses of biometrics. *Comm. ACM* **42**, 8 (August 1999), 136.

[Schneier, 2002] Bruce Schneier, One-time pads. Crypto-Gram Newsletter by Bruce Schneier, 2002. Available as *http://www.schneier.com/crypto-gram-0210.html#7.*

[SecurityFocus, 2008] SecurityFocus. 2008. BugTraq and vulnerability database available as *http://www.securityfocus.com/*.

[Skorobogatov, 2005] Sergei Skorobogatov, Semi-invasive attacks - a new approach to hardware security analysis. Ph. D. Thesis, Computer Laboratory, University of Cambridge, 2005. Available as *http://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-630.html*.

[Smith and Weingart, 1998] Sean W. Smith and Steve Weingart, Building a high-performance, programmable secure coprocessor. IBM T.J. Watson Research Center, USA, October 16, 1998. Available as *http://www.research.ibm.com/secure_systems_department/projects/scop/papers/arch.pdf*.

[Soutar et al., 1998] Colin Soutar, Danny Roberge, Alex Stoianov, Rene Gilroy and B.V.K. Vijaya Kumar, Biometric encryption. Bioscrypt Inc., 1998. Available as *http://www.bioscrypt.com/assets/documents/whitepapers/Biometric_Encryption.pdf*.

[Staniford et al., 2001] Stuart Staniford, Gary Grim and Roelof Jonkman, Flash worms: thirty seconds to infect the Internet. Silicon Defense, 2001. Available as *http://richie.idc.ul.ie/eoin/SILICON%20DEFENSE%20-%20Flash%20Worm%20Analysis.htm*.

[Stillerman et al., 1999] Matthew Stillerman, Carla Marceau and Maureen Stillman, Intrusion detection for distributed applications. *Comm. ACM* **42**, 7 (July 1999), 62-69.

[Sundström, 2008] Harri Sundström, Roskapostin estäminen. Pro gradu -tutkielma, Tampereen yliopisto, Tietojenkäsittelytieteiden laitos, tammikuu 2008. Saatavana osoitteesta *http://www.cs.uta.fi/research/theses/masters/Sundstrom_Harri.pdf*.

[Suojelupoliisi, 2008] Suojelupoliisi, Suojelupoliisin vuosikertomus 2007, tiedustelutoiminta Suomessa. Saatavana osoitteesta *http://www.poliisi.fi/poliisi/supo* -> Vuosikertomus 2007 -> Tiedustelutoiminta Suomessa.

[Tobias, 2007] Marc Weber Tobias, Medeco locks: are they secure enough? In.security.org, 2007. Available as *http://www.thesidebar.org/insecurity/?p=89*.

[Uludag et al., 2004] Umut Uludag, Sharath Pankanti, Salil Prabhakar and Anil K. Jain, Biometric cryptosystems: issues and challenges. *Proceedings of the IEEE*, Vol. **92**, No. 6, June 2004. Available as *http://citeseer.ist.psu.edu/uludag04biometric.html*.

[Vincent, 2003] Ryan Vincent, Attack reveals GNU project's vulnerability. NewsFactor Network, August 14, 2003. Available as *http://www.newsfactor.com/perl/story/22090.html*.

[Walker, 1985] S. T. Walker, Network security overview. *Proceedings of the IEEE symposium on Security and privacy*, 1985, 62-76.

[Weaver, 2001] Nicholas C. Weaver, Warhol worms: the potential for very fast Internet plagues. University of California at Berkeley, 2001. Available as *http://www.iwar.org.uk/comsec/resources/worms/warhol-worm.htm*.

[Whitman et al., 2001] Michael E. Whitman, Anthony M. Townsend and Robert J. Aalberts, Information systems security and the need for policy. In: Gurpreet Dhillon (ed.), *Information Security Management: Global Challenges in the New Millenium*. Idea Group Publishing, 2001, 9-18.

[Wikipedia, 2008a] Wikipedia encyclopedia, Hardware random number generator. Wikipedia, 2008. Available as *http://en.wikipedia.org/wiki/Hardware_random_number_generator*.

[Wikipedia, 2008b] Wikipedia encyclopedia, Nuclear weapon design. Wikipedia, 2008. Available as *http://en.wikipedia.org/wiki/Nuclear_weapon_design*.

[Yale, 2008] Yale, The history of Yale locks. Yale, 2008. Available as *http://www.yalelock.com/Yale/Templates/GlobalNormalWithMenu____142.aspx*.

[Ye et al., 2001] Nong Ye, Joseph Giordano and John FeldMan, A process control approach to cyber attack detection. *Comm. ACM* **44**, 8 (August 2001), 76-82.

[Ylönen et al., 2006] Tatu Ylönen, Tero Kivinen, Timo J. Rinne, Sami Lehtinen, Markku-Juhani O. Saarinen et al., The Secure Shell (SSH) transport layer protocol. Request for Comments (RFC) 4253. SSH Communications Security Corp., 2006. Available as *http://tools.ietf.org/html/rfc4253*.