Hyökkäykset verkoissa

Seuraavassa listassa on lueteltu muutamia hyökkäystyyppejä:

- Verkon yli tapahtuva sosiaalinen huijaus (pyydetään salasanoja, rahaa, yms).
- Haittaohjelmien levitys liitetiedostojen avulla ja www-sivujen avulla.
- Verkkolaitteiden ja ohjelmistojen etäanalysointi.
- Palvelunestohyökkäykset.

Verkon rakenteen selvittäminen I

- Eräs hyökkääjän ensimmäisistä tavoitteista on saada tietoa verkon isäntäkoneista. Tämä tieto sisältää mm. ne IP-osoitteet, jotka ovat käytössä, koneiden käyttöjärjestelmät ja tarjotut palvelut.
- *Verkon kartoitus* tarkoittaa toimintaa, jolla saadaan selville verkon isäntäkoneet, kun taas *tunnustelu* (probing) tarkoittaa toimintaa, jolla pyritään saamaan tietoa yksittäisestä koneesta.
- Tyypillinen hyökkäys etenee siten, että ensin kartoitetaan verkkoa aktiivisten koneiden selville saamiseksi. Sen jälkeen koneita tunnustellaan. Tavoitteena on saada selville palvelu, jolla on tunnettuja heikkouksia turvallisuudessa. Lopuksi tehdään varsinainen hyökkäys valittua konetta ja palvelua vastaan.

Verkon kartoitus I

- Yksinkertaisin kartoitusmenetelmä on yleislähettää ping-sanoma ja katsoa, kuka vastaa. Jos kohdeverkko on luokan B verkko *10.10.x.x*, hyökkääjä lähettää esimerkiksi paketin

  ```
  11:42:16.33 attacker.com > 10.10.255.255 ICMP: Echo Reques
  ```

- Useita paketteja lähetetään, jolla varmistetaan, että mahdollisimman moni saa paketin. Lähdeosoitetta ei voi väärentää, sillä vastaussanomien täytyy saapua hyökkääjälle.
- Hyökkääjä voi kuitenkin lähettää samalla paketteja, joissa on väärennetty lähdeosoite. Tämä vaikeuttaa todellisen hyökkääjän selville saamista. Tällainen hyökkäys on helppo havaita ja palomuuri voidaan virittää siten, ettei se päästä läpi yleislähetyksiä.

Verkon kartoitus II

- Toinen tapa hyökätä on lähettää paketti jokaiselle mahdolliselle koneelle verkossa. Hyökkääjä valitsee TCP:tä käyttävän portin, josta pääsee palomuurin läpi kohdekoneille. Sen jälkeen hän alkaa lähettää paketteja jokaiseen mahdolliseen IP-osoitteeseen.
- Oletetaan, että palomuuri sallii telnet-yhteyden (portti 23) jokaisesta ulkopuolisesta IP-osoitteesta. Tällöin ensimmäiset paketit voisivat näyttää seuraavilta:

```
11:47:34.09 attacker.com.2213 > 10.10.1.1.23 S
11:47:34.19 attacker.com.2213 > 10.10.1.2.23 S
11:47:34.27 attacker.com.2213 > 10.10.1.3.23 S
    jne
```

- Tämäkin on helppo havaita, jos hyökkääjä on noin yksinkertainen. Jos kohdekoneet valitaankin satunnaisesti ja paketit lähetetään epäsäännöllisin välein, hyökkäys voi olla vaikea havaita.

Verkon kartoitus III

- Oveluudella on kuitenkin rajansa. Jos verkossa on 65 000 mahdollista konetta ja lähetetään yksi paketti minuutissa, kestää puolitoista kuukautta käydä kaikki läpi.
- Jos hyökkäjä on näin kärsivällinen, menetelmä saattaa toimia. Ei kuitenkaan tarvita kuin yksi valpas ylläpitäjä, joka näkee yhteyspyynnöt ja tulee uteliaaksi. Siten minuutin väli ei ole aina riittävä.
- On mahdollista hyökätä vielä älykkäämmin. Hyökkääjän kannalta on riittävää löytää pieni määrä koneita, jolloin hidas ja satunnaistettu kartoitus onnistuu paremmin. Etsintä lopetetaan, kun riittävä määrä koneita on löytynyt.

Verkon kartoitus IV

- On muitakin tapoja tehdä kartoitus vaikeasti havaittavaksi. Edellä kuvatuissa lähetyksissä käytettiin SYN-paketteja, joista pidetään kirjaa monissa koneissa. Samoin monet tunkeutumisen estojärjestelmät seuraavat näitä paketteja. Vaihtamalla lippua tehdään havaitseminen vaikeammaksi:

```
10:47:34.33 attacker.com.2213 > 10.10.17.121.23 S ack
11:13:21.24 attacker.com.2213 > 10.10.3.207.23 S ack
12:11:11.53 attacker.com.2213 > 10.10.51.14.23 S ack
```

- Tässä tapauksessa näyttää siltä, että hyökkäjä vain vastaa yhteyspyyntöihin verkon sisältä. Edelleen voidaan lisätä älykkyyttä muokkaamalla paketteja niin, että ne näyttävät tulevan Web-palvelimelta surfailun tuloksena (portti 80):

```
10:47:34.33 attacker.com.80 > 10.10.17.121.23 S ack
11:13:21.24 attacker.com.80 > 10.10.3.207.23 S ack
12:11:11.53 attacker.com.80 > 10.10.51.14.23 S ack
```

Verkon kartoitus V

- Jotta tällainen hyökkäys havaittaisiin, tunkeutumisen estojärjestelmän tulee olla joko tilat muistava (muistaa, ettei ollut SYN-paketteja, joihin odotetaan vastauksia) tai pitää kirjaa kaikista aktiivisista verkon koneista.
- Toinen vaihtoehto on lähettää reset-paketteja:

```
11:47:34.09 attacker.com.2213 > 10.10.17.121.23 R
11:53:43.12 attacker.com.2213 > 10.10.3.207.23 R
12:31:24.01 attacker.com.2213 > 10.10.51.14.23 R
```

- Reset-paketit ilmaisevat, että jotain on mennyt vikaan yhteyden solmimisessa. Nämä paketit läpäisevät useimmat palomuurit, joskin tilat muistava palomuuri pystyy päättelemään, ettei ollut mitään yhteysyrityksiä, joten reset-paketit voidaan hylätä.
- Näitä paketteja ei myöskään kirjata ylös niin usein kuin esim. SYN-paketteja. Ne ovat myös tarpeeksi yleisiä niin, etteivät tunkeutumisen estojärjestelmät kiinnitä niihin huomiota.

Verkon kartoitus VI

- Ellei systeemi ole tilansa muistava, on hyvät mahdollisuudet, että tällainen hyökkäys menee läpi. Reset-hyökkäyksistä on itse asiassa monia versioita. Näitä on käsitelty artikkelissa [**?**].

Sormenjäljet I

- Sormenjäljet tarkoittaa yksityiskohtia, joita käyttöjärjestelmät jättävät paketteihin. Ohjelmien avulla voidaan seurata näitä paketteja ja päätellä käyttöjärjestelmä.
- Sormenjälkien selvittely voidaan tehdä hyvin deterministisesti. Eri käyttöjärjestelmät reagoivat eri tavalla eri ärsykkeisiin, ja tätä voidaan käyttää hyväksi käyttöjärjestelmää ja sen versiota määriteltäessä.
- On kuitenkin tilanteita, joissa myös tilastollisilla menetelmillä on merkitystä. Hyvä yleislähde sormenjälkiin on [**?**].
- Perustekniikkka on lähettää paketti, jossa on outo lippukombinaatio. Eri KJ:t vastaavat outoihin kombinaatioihin eri tavalla.
- Esimerkiksi odottamaton FIN-paketti saa aikaan joissakin systeemeissä vastauksen, vaikkakin oikeaoppinen tapa olisi hylätä paketti. Vanhat Linux-versiot toistivat oudot lippukombinaatiot vastauspaketeissaan.

Sormenjäljet II

- Toinen tapa erotella käyttöjärjestelmiä on tutkia pakettien järjestysnumerointia. Numerointi voi olla satunnaista tai determinististä. Keräämällä tilastoa koneen lähettämistä paketeista voidaan koneita jakaa erilaisiin luokkiin.
- Tämä tapa on passiivista sormenjälkien tunnistamista, koska siinä ei lähetetä omia paketteja. Samoin se vaatii suuren määrän paketteja, ennenkuin johtopäätökisä voidaan tehdä.
- Monet KJ:t asettavat oletusarvoisesti don't fragment -bitin. Toiset taas asettavat ne vain tietyissä tilanteissa. Uudet KJ:t asettavat bitin useammin kuin vanhat.
- Käyttöjärjestelmillä on eri vaihtoehtoja virhetilanteista raportointiin ICMP:n avulla. Jos esimerkiksi kone saa suuren määrän paketteja suljettuun porttiin, sen ei tarvitse generoida sanomaa ICMP destination unreachable jokaiselle. Eri KJ:t tekevät erilaisia valintoja näissä tilanteissa.

Sormenjäljet III

- Valinnainen kenttä (options field) on hyvä lähde sormenjäljille. Koska kenttä on valinnainen, sen toteutus on vapaata. Siten kentän piirteet antavat paljon tietoa KJ:tä määriteltäessä.
- Muita yksityiskohtia, joita voidaan käyttää sormenjälkien tunnistamisessa, ovat TTL-kenttä (time to live), ikkunan koko, palvelun tyyppi ja urgent-osoitin. Nämä sopivat erityisesti passiiviseen lähetymistapaan.

Tunnustelu I

- Useilla palveluilla on tunnetut heikkoutensa, joita hyökkääjä voi käyttää hyväkseen. Eri versioilla on eri heikkoudet, joten ei riitä, että löydetään tietty palvelu tietyssä koneessa, vaan on lisäksi selvitettävä palvelun versio.
- Aluksi hyökkääjä selvittää palvelut porttiskannauksen avulla. Yksinkertaismillaan voidaan tutkia kaikki portit, mutta tämä on helppo havaita. Parempi menetelmä on tutkia vain ne portit, joita voidaan käyttää hyväksi:

```
13:12:22:33 attacker.com.2113 > 10.10.17.121.telnet S
13:12:22:54 attacker.com.2114 > 10.10.17.121.smtp S
13:12:22:73 attacker.com.2115 > 10.10.17.121.finger S
13:12:22:97 attacker.com.2116 > 10.10.17.121.http S
13:12:23:13 attacker.com.2117 > 10.10.17.121.imap S
```

Tunnustelu II

```
13:12:23:27 attacker.com.2118 > 10.10.17.121.rlogin S
13:12:23:41 attacker.com.2119 > 10.10.17.121.printer S
```

- Kun palvelu on löytynyt, helpoin tapa selvittää sen versio on katsoa sen lähettämää vastausta. Jos esimerkiksi kirjoitetaan komento `telnet mycomputer.com 25`, saadaan seuraavan kaltainen vastaus

  ```
  220 mycomputer.com ESMTP Sendmail 8.9.3/8.9.3; Sat, 8 D
  17:31:05-0500
  ```

  Komentamalla `help` tämän viestin kohdalla tuottaa selityksen
  214-This is Sendmail version 8.9.3
  214-Topics:
  214-HELO EHLO MAIL RCPT DATA
  214-RSET NOOP QUIT HELP VRFY
  214-EXPN VERB ETRN DSN
  214-For more info use "HELP <topic>".

Tunnustelu III

214-To report bugs in the implementation send email to
214-sendmail-bugs@sendmail.org.
214-For local information send email to postmaster at
214- your site
214 End of HELP info

Skannerit I

- Skannerilla tarkoitetaan tässä ohjelmaa, joka analysoi verkon yli toisia verkkoja ja niiden palvelimia ja palomuureja haavoittuvuuksien löytämiseksi.
- Skannerit ovat tehokas tapa löytää tietoturva-aukkoja ja niitä käytetäänkin paljon omia verkkoja analysoitaessa. Toisten verkkojen tutkiminen skannereilla on kiellettyä, mutta hyökkäyksissä sitä kuitenkin tehdään.
- Ensimmäinen skanneri oli Chris Klausin ISS (Internet Security Scanner) vuodelta 1992.
- Suurta huomiota herätti SATAN (Security Administrator Tool for Analyzing Networks, Dan Farmer ja Wietse Venema 1995).
- Nykyään skannerit ovat yhä kehittyneempiä. Uusimpia ovat mm. Nessus, SARA, Nikto, eEye ja Microsoft Baseline Security Analyser.

Esimerkkejä erilaisista palvelunestohyökkäyksistä

- Nämä hyökkäykset pyrkivät joko ajamaan alas verkon, tietokoneen tai prosessin tai muuten vaikeuttamaan palvelun tarjontaa.
- Seuraava lista ei ole täydellinen, sillä uusia palvelunestohyökkäyksiä keksitään lähes päivittäin. Tässä listassa on lähinnä sellaisia palvelunestohyökkäyksiä, jotka ovat esiintyneet aikaisemmin, jopa kauan sitten, ovat hyvin tunnettuja ja perustuvat TCP/IP-pinon heikkouksiin.

Maahyökkäys

- Maahyökkäyksessä (Land Attack) konstruoidaan TCP SYN -paketti, jossa kohde- ja lähdeosoite ovat samoja.
- Joissakin vanhoissa järjestelmissä tällainen paketti aiheutti vastaanottajan puolella lukkiuman, jonka johdosta kone täytyi käynnistää uudelleen. Hyökkäykseen tarvitaan siis vain yksi ainoa paketti.
- Hyökkäys ei liene kovin relevantti nykyisissä ympäristöissä, mutta ohjelmistojen uusiokäytön ja mahdollisten koodausvirheiden vuoksi se saattaa taas tulevaisuudessa putkahtaa esiin.
- Menetelmä on tyypillinen palvelunestohyökkäyksissä. Outo tai mahdoton paketti konstruoidaan ja lähetetään. Vastaanottopäässä tällainen paketti aiheuttaa hämmennystä, joka johtuu joko toteutusvirheestä tai protokollan epätäydellisyydestä.

Neptunus

- Neptunus tai SYN-tulva käyttää hyväkseen sitä tietoa, että jokaista puoliksi avoinna olevaa TCP-yhteyttä kohti tcpd (tcp-demoni) luo tietueen, jossa pidetään yhteystietoja.
- Jos yhteyttä ei luoda loppuun asti tietyn ajan kuluessa, yhteys katkaistaan ja tietueen varaama tila vapautetaan.
- Jos kuitenkin riittävä määrä yhteyksiä alustetaan ennen kuin ajastin laukeaa, tietorakenne vuotaa yli. Se aiheuttaa taulukon ylivuodon (segmentation fault) ja tietokoneen lukkiintumisen.
- Tässä hyökkäyksessä konstruoidaan paketti, jonka IP-lähdeosoite on saavuttamaton. Toisin sanoen mikään kone ei vastaa SYN/ACK-sanomaan, jonka kohdekone lähettää paketin saatuaan. Siten yhteys jää auki.

Ping O'Death

- Tämä hyökkäys koostuu ICMP echo-pyynnöstä (ping), jossa on liian pitkä (yli 64 KB) "hyötykuorma". Vanhemmat käyttöjärjestelmät lukkiintuvat tai käynnistyvät uudelleen, kun puskuri, johon paketti varastoidaan, vuotaa yli.
- Ensimmäiset Windows 95 -versiot sisälsivät ping-ohjelman, joka salli käyttäjän määritellä paketin koon, vaikka se olisikin ollut liian pitkä. Tämä teki järjestelmän suosituksi hyökkäyskohteeksi.
- Samoin kuin maahyökkäyksessä tämäkin hyökkäys vaatii vain yhden paketin. Nykyiset käyttöjärjestelmät eivät ole enää haavoittuvia tälle hyökkäykselle.

Prosessitaulu I

- Prosessitauluhyökkäys kehitettiin MIT:n Lincoln DARPA-laboratorioissa testaamaan tunkeutumisen estojärjestelmiä. Tavoitteena oli kehittää uusia hyökkäyksiä, jotta nähtäisiin, voitaisiinko ne paljastaa.
- Hyökkäyksen idea perustuu siihen, että joka kerran kun TCP-yhteyspyyntö saapuu, prosessi haarautuu (fork). Jos pyyntöjä saapuu hyvin paljon, prosessitaulu täyttyy. Kun taulu on täynnä, uusia prosesseja ei voida luoda. Seurauksena on tilanne, jossa tietokone ei voi tehdä mitään.
- Tämä hyökkäys täytyy kohdistaa prosesseille, jotka sallivat monia yhteyksiä yhtäaikaa. Esimerkiksi sendmail ei hyväksi uusia yhteyksiä, jos niitä on jo ennestään paljon.

Prosessitaulu II

- Sen sijaan finger sallii aina uusia yhteyksiä. Jotkut aikaisemmat fingerin versiot eivät käyttäneet ajastinta yhteyksien sulkemiseen, vaan ne jäivät auki niin pitkäksi aikaa, kunnes toinen sulki ne. Jokainen yhteys sai oman tunnuksensa ja jos riittävästi yhteyksiä avattiin, prosessitaulu täyttyi.

Targa3

Targa3-hyökkäyksessä lähetetään laittomia paketteja kohteelle. Nämä virheelliset paketit saavat jotkut systeemit kaatumaan. Vaikka systeemi ei kaatuisikaan, virheelliset paketit kuluttavat tavallista enemmän resursseja. Tyypillisesti paketeissa on vialla jotain seuraavista:

- Virheellinen paloittelu, protokolla, koko tai IP-otsakkeen arvo.
- Virheelliset optiot.
- Virheelliset TCP-segmentit.
- Virheelliset reitityslipput.

Smurf-hyökkäys

- Smurf-hyökkäyksessä on kolme osapuolta: hyökkääjä, kohde ja välittäjä, joka höynäytetään tekemään varsinainen hyökkäys.
- Hyökkääjä konstruoi echo request -paketteja, joissa on lähteenä aiottu kohde ja kohteena välittäjä. Nämä paketit yleislähetetään, jotta maksimoidaan vastausten lukumäärä.
- Kaikki koneet välittäjän verkossa vastaavat kohteelle, joka ei voi käsitellä niin suurta määrää paketteja. Kohde kaatuu tai ainakin hidastuu siinä määrin, ettei kykene toimimaan normaalisti.
- Hyökkääjä ei näy kohteen lokitiedostoissa. Hyökkääjä paljastuu vain välittäjän verkon lokeista.

Teardrop-hyökkäys

Teardrop perustuu siihen, että jotkut vanhat TCP/IP-toteutukset eivät käsittele asianmukaisesti tilannetta, jossa paketti on pilkottu, mutta palaset eivät ole erillisiä. Jos koneessa on tämän vian sisältävä TCP/IP-toteutus ja hyökkääjä lähettää paketin, joka näyttää oikealta, mutta jonka palaset eivät ole erillisiä, kone kaatuu.

UDP-tulva

- UDP-tulva saa aikaan, että kaksi konetta hyökkäävät toisiaan vastaan. On tietty määrä portteja, jotka vastaavat paketilla saatuaan paketin. Echo (portti 7) ja chargen (portti 19) ovat tällaisia. Echo kaiuttaa paketin takaisin, kun taas chargen generoi merkkivirran.
- Tarkastellaan UDP-pakettia, jolla on lähdeporttina 7 ja kohdeporttina 19. Paketti generoi joitakin merkkejä kohdekoneesta, jotka merkit lähetetään lähdekoneen echo-porttiin.
- Lähde kaiuttaa nämä paketit takaisin, mikä puolestaan aiheuttaa lisää paketteliikennetta koneiden välille jne. Jossain vaiheessa molemmat koneet kuluttavat kaiken aikansa lähettämällä paketteja edestakaisin kunnes toinen tai molemmat kaatuvat.
- Jos ulkopuolelta tulee paketteja, joiden lähdeosoite on sisäpuolella, palomuurin pitäisi pysäyttää nämä paketit. Tällaiset paketit ovat melkein varmasti väärennettyjä. Jos kysymyksessä ei olisikaan hyökkäys, tällaiset paketit ovat merkki siitä, että jotain on mennyt vikaan.

Postipommi

- Postipommi on hyökkäys käyttäjää vastaan, mutta se voi kaataa myös koneen. Hyökkäyksessä lähetetään monia viestejä jollekin käyttäjälle sähköpostiosoitteeseen.
- Jos viestien määrä on satoja, käyttäjä kärsii suuresti. Jos määrä on tuhansia ja viestit ovat pitkiä, postijono täyttyy ja kone voi kaatua. Myös levy voi täyttyä, mikä aiheuttaa myös muiden käyttäjien viestien tuhoutumisen tai ainakin välittämisen hidastamisen.
- Postipommi on helppo toteuttaa ja varsin suosittu. Vaikka postiosoite voidaan väärentää, IP-osoite pysyy aitona. Siten uhrin on mahdollista ottaa yhteys lähettävän koneen ylläpitoon ja pyytää keskeyttämään hyökkäys.

Web-pommi

- Tämä on samanlainen hyökkäys kuin postipommi, mutta lähetetään suuri määrä Web-pyyntöjä yhdelle Web-palvelimelle. Vaikutukset ovat samat kuin postipommissa.
- Tämäkin hyökkäys on helppo havaita, sillä suuri määrä pyyntöjä tulee yhdestä ja samasta paikasta. Koska sekä email että Web käyttävät TCP:tä, IP-lähdeosoitetta ei voi väärentää, sillä kättelyn täytyy onnistua ennen kuin sovellukset alkavat toimia.
- Tämän vuoksi hyökkääjät käyttävät usein välikättä, josta aloittavat posti- tai Web-hyökkäyksensä.
- Uudempi tapa on käyttää botnet-verkkoja hajautettuun hyökkäykseen. Tällaista hyökkäystä vastaan ei ole juurikaan keinoja suojautua.

Osapuolten todentaminen I

- Osapuolten todentaminen on yksi tärkeimmistä tehtävistä hajautetuissa järjestelmissä. Monessa tilanteessa todentaminen täytyy automatisoida, joten esimerkisi etukäteen jaettujen salasanojen käyttö ei ole mahdollista.
- Monet todentamistekniikat perustuvat digitaaliseen allekirjoitukseen, joka puolestaan perustuu julkisen avaimen salaukseen.
- Lisäksi on olemassa menetelmiä, joilla yhteinen salaisuus voidaan generoida automaattisesti kahden osapuolen välille, joilla ei ole etukäteen mitään yhteistä salaisuutta (Diffie-Hellman). Tällaista tarvitaan tilanteissa, joissa täytyy lähettää luottamuksellista tietoa toiselle osapuolelle, vaikka osapuolilla ei ole ennestään mitään yhteistä salaista avainta.

Julkisen avaimen salaus RSA I

- Julkisen avaimen salauksen eli *epäsymmetrisen salauksen* edellytyksenä on, että löytyy salausmenetelmä, jossa on mahdotonta saada selville salaista purkuavainta D_K, vaikka salausavain E_K tunnetaan.
- Julkisen avaimen salauksen etuna on, että kuka tahansa voi lähettää salakirjoitetun viestin vastaanottajalle ilman, että molempien täytyy sopia etukäteen salaisesta avaimesta. Vastaanottaja on kuitenkin ainoa, joka pystyy purkamaan viestin salaisella avaimellaan.
- Julkisen avaimen salauksen idean julkaisivat ensimmäisinä Diffie ja Hellman vuonna 1976. Joissakin lähteissä mainitaan myös Merkle samanaikaisena keksijänä.
- Heidän ehdottamansa menetelmä oli kuitenkin lähinnä teoreettinen ja varsin epäkäytännöllinen.

Julkisen avaimen salaus RSA II

- Ensimmäinen julkisen avaimen julkinen käytännöllinen menetelmä oli RSA, jonka kehittivät Rivest, Shamir ja Adleman 1977. RSA on edelleenkin laajimmin käytössä oleva julkisen avaimen järjestelmä.
- Vuonna 1997 CESG (brittiläinen kryptografian osasto) julkaisi dokumentteja, jotka osoittivat, että James Ellis oli jo vuonna 1970 keksinyt julkisen avaimen salauksen.
- Samoin Clifford Clocks oli 1973 kuvannut RSA:n version, missä salauseksponentti e oli sama kuin modulus n.
- RSAn jälkeen on ehdotettu monia muita, mutta ne eivät ole saaneet samanlaista suosiota.
- Julkisen avaimen menetelmät eivät voi taata turvallisuutta joka tilanteessa. Jos vastapuolella nimittäin on salateksti, hän voi salata jokaisen mahdollisen selvätekstin ja verrata tulosta salatekstiin. Mikäli tulos on sama, on selväteksti paljastunut. Siten mahdollisia selvätekstejä on oltava suunnaton määrä.

RSA I

RSA:ssa avaimet ovat muodoltaan seuraavia:

- julkinen avain on lukupari (e, n);
- salainen avain on lukupari (d, n);
- selväteksti jaetaan lohkoihin, lohkon koon binäärilukuna tulee olla alle n eli lohko käsittää korkeintaan $\log_2(n)$ bittiä.

Salaus tapahtuu kaavalla

$$C = M^e \mod n.$$

Purkuun käytetään kaavaa

$$M = C^d \mod n = (M^e)^d \mod n = M^{ed} \mod n.$$

Ennenkuin systeemi toimii,

- on löydettävä sopivat luvut e, d ja n,
- on laskettava tehokkaasti M^e ja C^d kaikilla $M < n$,
- d ei saa olla helposti laskettavissa e:stä ja n:stä.

Vaatimukset avaimille I

Tällä hetkellä vaatimukset avaimille ovat:

- n:n on oltava välillä $1024 - 2048$ bittiä,
- p:n ja q:n tulee olla "lähellä"toisiaan, välillä $10^{75} - 100^{100}$,
- $p-1$:n ja $q-1$:n tulee sisältää suuri alkulukutekijä ja
- $\mathrm{syt}(p-1, q-1)$:n tulee olla pieni.

Tarkastellaan vielä avainten pituuksia. Koska laitteistot kehittyvät koko ajan, täytyy avaimen pituuksia kasvattaa aika ajoin. Seuraavassa taulukossa on sekä symmetrisiltä että epäsymmetrisiltä salausmenetelmiltä vaadittavia avainpituuksia tuleville vuosille. Arviot perustuvat ECRYPT II -suosituksiin vuodelta 2009. Vastaavia suosituksia on NIST:ltä, NSA:lta ja monilta muilta sekä yksittäisiltä tutkijoilta että organisaatioilta. Taulukossa DL tarkoittaa diskreettiin logaritmiiin perustuvia tekniikoita ja EC elliptisiin käyriin perustuvia.

Vaatimukset avaimille II

Taso	Suoja	Symm.	Epäsymm
1	Attacks in "real-time" by individuals	32	
2	Very short-term protection against small organizations	64	816
3	Short-term protection against medium organizations	72	1008
4	Smallest general-purpose level, 2009 to 2012	80	1248
5	Legacy standard level, 2009 to 2020	96	1776
6	Medium-term protection from 2009 to 2030	112	2432
7	Long-term protection from 2009 to 2040	128	3248
8	"Foreseeable future", against quantum computers	256	15424

Digitaalinen allekirjoitus RSA:n avulla I

- Esimerkiksi elektronisessa kaupankäynnissä ja varmenteiden käytössä täytyy kyetä osoittamaan, että tietty taho on viestin lähettäjä. *Digitaalinen allekirjoitus* tähtää siihen, että se osoittaa kiistatta lähettäjän ja lähetyksen ajankohdan sekä todentaa sanoman.
- Lisäksi kolmannen osapuolen tulee kyetä verifioimaan allekirjoitus. Todentaminen tässä yhteydessä tarkoittaa, että allekirjoitettua viestiä ei voi muuttaa vaikuttamatta allekirjoitukseen.
- *Suora digitaalinen allekirjoitus* perustuu julkisen avaimen salaukseen. Edellytyksenä on, että salausmenetelmälle pätee ehto

$$E_{K_p}(D_{K_s}(M)) = M.$$

- Esimerkiksi RSA toteuttaa yllä olevan kaavan. Jos ehto pätee, sanoma allekirjoitetaan "salaamalla"se lähettäjän salaisella avaimella D_{K_s}.

Digitaalinen allekirjoitus RSA:n avulla II

- Vastaanottaja purkaa salauksen lähettäjän julkisella avaimella ja toteaa, että tuloksena on selväkielinen viesti.
- Jos lähettäjä haluaa, ettei viestiä voi lukea muut kuin vastaanottaja, viesti voidaan vielä salata vastaanottajan julkisella avaimella.
- Usein allekirjoituksen yhteydessä käytetään myös tiivistefunktiota. Tiiviste allekirjoitetaan, ei alkuperäistä lähdettä.
- *Digitaalisen allekirjoituksen standardi* on muunnos ELGamalin allekirjoituksesta. Siinä allekirjoitus on kohtuullisen kokoinen päinvastoin kuin ElGamalissa. Se kehitettiin osaksi sen vuoksi, että RSA:n käyttöä säätelivät patentit.
- Näiden perustekniikoiden lisäksi on lukuisia muita, joita on lueteltu teoksessa Menezes, van Oorschot, Vanstone: *Handbook of Applied Cryptography,* joka on saatavissa ilmaiseksi verkosta.

Julkisen avaimen infrastruktuuri ja X.509 I

- Jotta julkisen avaimen salausta ja digitaalista allekirjoitusta voitaisiin käyttää, osapuolten julkiset avaimet on löydettävä. Lisäksi on oltava varma, että tietty julkinen avain on juuri tietyn osapuolen avain. Jos näin ei olisi, yksi osapuoli voisi lähettää salaista tietoa kolmannelle osapuolelle tietämättään.
- Julkisten avainten löytyminen on asia, jota ei vielä ole ratkaistu täydellisesti. Paremmin, joskaan ei täydellisesti, on sen sijaan ratkaistu, miten varmistaudutaan julkisen avaimen omistajasta. Yleisesti käytetään varmenteita, jotka luotettu kolmas osapuoli on varmentanut. Tavallisin varmennesysteemi on X.509.
- X.509 on yksi osa X.500-suositussarjassa, joka määrittelee hakemistopalvelun. X.509 määrittelee hakemiston todennuspalvelut. X.509 on tärkeä standardi, koska X.509:n julkisiin avaimiin liittyvät varmenteet ja todennusprotokollat ovat käytössä ainakin ohjelmistoissa S/MIME, IP:n turvapalvelu, SSL/TLS ja SET.

Julkisen avaimen infrastruktuuri ja X.509 II

- X.509 julkaistiin 1988. Myöhemmin standardia muokattiin vastaamaan paremmin turvallisuusvaatimuksia. Uudistettu versio julkistettiin 1993 ja kolmas versio 1995.
- X.509 perustuu julkisen avaimen salaukseen ja digitaalisiin allekirjoituksiin. Standardi suosittelee RSA:ta, muttei vaadi sitä. Digitaalinen allekirjoitus perustuu tiivistefunktion käyttöön. Sitäkään ei ole kiinnitetty.
- X.509:n perustana on julkisen avaimen varmenne, joka liittyy jokaiseen käyttäjään. Oletuksena on, että varmenteet on luonut jokin luotettava osapuoli (CA, Certification Authority, *varmenneviranomainen*). Joko CA tai käyttäjä on pannut varmenteet hakemistoon. Hakemistopalvelin itse ei ole vastuussa julkisten avainten luonnista eikä varmenneperiaatteesta.
- Varmenne sisältää seuraavat osat:

Julkisen avaimen infrastruktuuri ja X.509 III

- *Versio*: Oletusarvo on 1. Jos *Initiator Unique Identifier* tai *Subject Unique Identifier* ovat läsnä, version pitää olla 2. Jos yksi tai useampi laajennus on läsnä, versio on 3.
- *Sarjanumero*: Kokonaisluku, yksikäsitteinen CA:n alueella.
- *Allekirjoitusalgoritmin tunnus*: Tämä tieto toistetaan Signature-kentässä varmenteen lopussa, joten sillä ei ole merkitystä tässä kohdassa.
- *Julkaisijan nimi*: Sen CA:n X.500-nimi, joka loi ja allekirjoitti tämän varmenteen.
- *Voimassaoloaika*: Alku- ja loppuajankohta.
- *Subjektin nimi*: Käyttäjän nimi, johon tämä varmenne viittaa.
- *Subjektin julkisen avaimen informaatio*: Subjektin julkinen avain, algoritmin tunnus ja parametrit.
- *Julkaisijan yksikäsitteinen tunnus*: Valinnainen bittijono, joka määrittelee yksikäsitteisesti CA:n siinä tapauksessa, että samaa X.500-nimeä on käytetty eri olioille.
- *Subjektin yksikäsitteinen tunnus*: Kuten edellä.

Julkisen avaimen infrastruktuuri ja X.509 IV

- *Laajennukset*: Yksi tai useampi laajennuskenttä. Laajennukset lisättiin versioon 3. Ne selitetään myöhemmin.
- *Allekirjoitus*: Kattaa varmenteen kaikki muut kentät. Sisältää toisten kenttien tiivistekoodin salattuna CA:n yksityisellä avaimella. Kenttä sisältää allekirjoitusalgoritmin tunnuksen.

- Standardi käyttää seuraavaa merkintää varmenteen määrittämiseksi:

$$CA << A >>= CA\{V, SN, AI, CA, T_A, A, A_P\},$$

missä $Y << X >>$ on käyttäjän X varmenne, jonka on julkaissut varmenneviranomainen Y, ja $Y\{I\}$ on Y:n muodostama I:n allekirjoitus.

- Se koostuu I:stä, johon on lisätty salakirjoitettu tiivistekoodi. CA allekirjoittaa varmenteen salaisella avaimellaan. Jos vastaava julkinen avain on käyttäjälle tuttu, niin käyttäjä voi varmistua, että varmenne on todella CA:n vahvistama.

Julkisen avaimen infrastruktuuri ja X.509 V

- Kuka tahansa käyttäjä, joka voi käyttää CA:n julkista avainta, voi ottaa toisen käyttäjän varmennetun julkisen avaimen itselleen.
- Ainoastaan varmenneviranomainen voi muokata varmennetta ilman, että tämä paljastuisi. Koska varmenteet ovat väärentämättömiä, ne voidaan asettaa hakemistoon ilman sen kummempia suojauksia.
- Jos kaikki käyttäjät käyttävät samaa CA:ta, kaikki varmenteet voidaan asettaa hakemistoon kaikkien käyttäjien saataville. Lisäksi käyttäjä voi lähettää varmenteensa suoraan muille käyttäjille.
- Kummassakin tapauksessa B:n omistaessa A:n varmenteen B voi luottaa siihen, että salaus A:n julkisella avaimella on turvallista ja A:n allekirjoitukset väärentämättömiä.
- Jos käyttäjiä on paljon, yhteen CA:han turvautuminen ei ole käytännöllistä.

Julkisen avaimen infrastruktuuri ja X.509 VI

- Koska CA allekirjoittaa varmenteet, jokaisella käyttäjällä täytyy olla kopio CA:n julkisesta avaimesta. Tämä julkinen avain täytyy toimittaa jokaiselle käyttäjälle absoluuttisen turvallisesti.
- Siten jos käyttäjiä on monia, on edullisempaa, jos varmenneviranomaisia on useampia, joista jokainen toimittaa oman julkisen avaimensa osalle käyttäjiä.
- Oletetaan, että A on saanut varmenteen CA_1:ltä ja B CA_2:lta. Jos A ei varmasti tunne CA_2:n julkista avainta, niin CA_2:n julkaisema B:n varmenne on hyödytön A:lle.
- A voi lukea B:n varmenteen, muttei voi verifioida allekirjoitusta. Jos kuitenkin CA_1 ja CA_2 ovat turvallisesti vaihtaneet omia julkisia avaimia, niin A voi saada B:n julkisen avaimen seuraavasti:
  1. A hankkii hakemistosta CA_1:n allekirjoittaman CA_2:n varmenteen. Koska A turvallisesti tuntee CA_1:n, A saa CA_2:n julkisen avaimen kyseisestä varmenteesta, joka on CA_1:n vahvistama.

Julkisen avaimen infrastruktuuri ja X.509 VII

2. A palaa hakemistoon ja hankkii CA_2:n allekirjoittaman B:n varmenteen. Koska A tuntee nyt varmasti CA_2:n julkisen avaimen, A voi verifioida allekirjoituksen ja turvallisesti hankkia B:n julkisen avaimen.

- Edellä kuvatussa menetelmässä A on käyttänyt varmenteiden ketjua hankkiessaan B:n julkisen avaimen. X.509:n merkinnöin tämä ketju ilmaistaan kaavalla

$$CA_1 << CA_2 >> CA_2 << B >> .$$

Samalla tavalla B voi hankkia A:n julkisen avaimen käänteisellä ketjulla

$$CA_2 << CA_1 >> CA_1 << A >> .$$

Menetelmä ei rajoitu kahteen varmenteeseen. Ketju voi olla mielivaltaisen pitkä

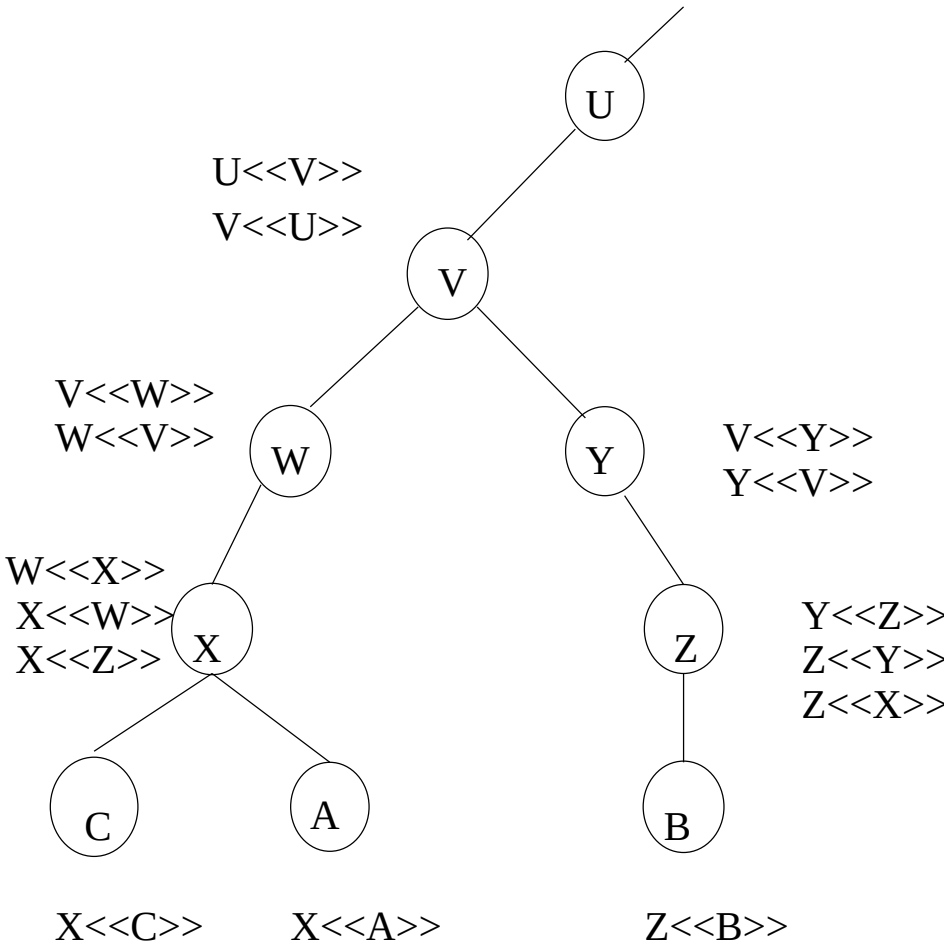
$$CA_1 << CA_2 >> CA_2 << CA_3 >> \cdots CA_n << B >> .$$

Julkisen avaimen infrastruktuuri ja X.509 VIII

Jokaisen parin (CA_i, CA_{i+1}) yllä olevassa ketjussa tulee luoda varmenteet toisilleen.

- Kaikkien näiden varmenteiden tulee olla hakemistossa ja käyttäjän pitää tietää, kuinka ne on linkitetty seuratakseen polkua toisen käyttäjän julkisen avaimen varmenteeseen.
- X.509 ehdottaa, että CA:t järjestetään hierarkkisesti siten, että navigointi on suoraviivaista. Seuraavassa kuvassa on esimerkki hierarkiasta.

Julkisen avaimen infrastruktuuri ja X.509 IX



Kuva: CA-hierarkia

Julkisen avaimen infrastruktuuri ja X.509 X

- Kuvassa U, V, W jne ovat varmenneviranomaisia. Laatikot osoittavat varmenteita, joita kyseinen CA ylläpitää hakemistossaan. CA:n hakemistoalkio sisältää kahden tyyppisiä varmenteita:
  - *Etuvarmenteet* (forward): Muiden varmenneviranomaisten generoimat X:n varmenteet.
  - *Käänteisvarmenteet* (reverse): X:n generoimat varmenteet, jotka ovat muiden varmenneviranomaisten varmenteita.

  Esimerkissä A voi hankkia seuraavat varmenteet hakemistosta saadakseen B:n varmenteen:

  $$X << W >> W << V >> V << Y >> Y << Z >> Z << B >> .$$

  Kun A on saanut nämä varmenteet, se voi purkaa ja todentaa B:n varmenteen.

Julkisen avaimen infrastruktuuri ja X.509 XI

- Varmenteella on päättymisajankohta. Uusi varmenne julkistetaan juuri ennen kuin edellinen vanhenee. Seuraavissa tilanteissa varmenne on aiheellista peruuttaa jo ennen sen vanhenemista.
  1. Käyttäjän salainen avain on luultavasti joutunut väärii käsiin.
  2. CA ei enää varmenna kyseistä käyttäjää.
  3. CA:n varmenne ei ole enää luotettava.
- Jokainen CA ylläpitää listaa kaikista peruutetuista varmenteista, jotka eivät vielä ole vanhentuneita.
- Nämä varmenteet käsittävät sekä CA:n käyttäjille myönnetyt että toisille varmenneviranomaisille vahvistetut varmenteet.

Julkisen avaimen infrastruktuuri ja X.509 XII

- Jokainen peruutettujen varmenteiden lista (CRL, certification revocation list) sijoitetaan hakemistoon. Sen allekirjoittaa julkaisija ja se sisältää julkaisijan nimen, listan luontipäivämäärän, seuraavan CRL:n ilmestymisajankohdan ja alkion jokaista peruutettua varmennetta kohti. Alkio sisältää sarjanumeron ja peruutuspäivämäärän.
- Kun käyttäjä saa varmenteen sanomassa, käyttäjän täytyy ratkaista, onko varmenne peruutettu vai ei. Käyttäjä voi tarkistaa asian hakemistosta joka kerran.
- Viiveitä välttääkseen käyttäjä voi myös ylläpitää paikallista käteismuistia varmenteita, listoja ja peruutettuja varmenteita varten.

Versio 3 I

Versiossa 2 oli puutteita:

1. Subjektikenttä on riittämätön välittämään avaimen omistajan identiteettiä julkisen avaimen käyttäjälle. X.509-nimet ovat lyhyitä, eivätkä anna täyttä tietoa nimen haltijasta.
2. Subjektikenttä on myös riittämätön monille sovelluksille, jotka tunnistavat olion sen sähköpostiosoitteen, URL:n tai muun Internetiin liittyvän tunnuksen avulla.
3. On tarvetta näyttää informaatiota turvapolitiikasta. Tämä mahdollistaisi turvasovelluksen tai -funktion, kuten IPSec, ja X.509-varmenteen yhteensovittamisen.
4. On tarvetta rajoittaa vahinkoja, joita virheellinen tai pahantahtoinen CA voi aiheuttaa. Vahinkoja voidaan vähentää asettamalla rajoituksia tietyn varmenteen käytettävyydelle.

Versio 3 II

5. On tärkeää kyetä tunnistamaan erikseen saman käyttäjän eri avaimet eri aikoina. Tämä piirre tukee avaimen elinkaaren hallintaa.

Yllä esitettyjä vaatimuksia on toteutettu ottamalla käyttöön valinnaisia laajennuksia, joita voidaan lisätä version 2 formaattiin. Jokainen laajennos sisältää *laajennuksen tunnuksen*, *kriittisyysindikaattorin* ja *laajennusarvon*.

Kriittisyysindikaattori osoittaa, voidaanko laajennus jättää turvallisesti huomiotta. Jos indikaattorilla on arvo true ja toteutus ei sisällä laajenusta, varmennetta tulee pitää virheellisenä. Varmenteen laajennukset jakautuvat kolmeen osaan:

- avain- ja käyttötiedot,
- subjektin ja julkaisijan attribuutit,
- varmenteen polkurajoitukset.

Avain- ja käyttötiedot sisältävät seuraavia kohtia:

Versio 3 III

- *Avainauktoriteetin tunnus* identifioi sen julkisen avaimen, jota voidaan käyttää verifioimaan tämän varmenteen tai CRL:n allekirjoitus.
- *Subjektin avaimen tunnus* identifioi varmennetun julkisen avaimen. Hyödyllinen päivitettäessä subjekti-avain -paria. Myös subjektilla voi olla useita avaimia ja vastaavasti erilaisia varmenteita eri tarkoituksiin.
- *Avaimen käyttö* osoittaa rajoitukset, joita avaimella on. Esimerkiksi avainta voidaan käyttää digitaaliseen allekirjoitukseen, kiistämättömyyden osoittamiseen, avaimen salaukseen, datan salaukseen, avaimesta sopimiseen, CA:n allekirjoituksen verifioimiseen varmenteissa tai CRL:ssä.
- *Yksityisen aivaimen käyttöaika.*
- *Varmennepolitiikka*. Varmenteita voidaan käyttää ympäristöissä, joissa voidaan soveltaa monentyyppistä käyttöpolitiikkaa. Tämä laajennus luettelee politiikat, joita varmenne tukee.

Versio 3 IV

- *Käyttötapakuvaukset* ovat ainoastaan CA:n julkistamissa toisen CA:n varmenteissa. Kuvaus osoittaa, että yhtä tai useampia julkaisijan politiikkoja voidaan pitää ekvivalentteina subjektin alueella käytetyn politiikan kanssa.

Attribuutit ovat seuraavia:

- *Subjektin vaihtoehtoiset nimet* ovat tärkeitä tuettaessa joitakin sovelluksia kuten e-mail, EDI, IPSec.
- *Julkaisijan vaihtoehtoiset nimet.*
- *Subjektin hakemistoattribuutit* välittävät minkä tahansa X.500-hakemiston attribuutin arvon.

Rajoitteita voidaan asettaa CA:n julkaisemille toisen CA:n varmenteille:

- *Perusrajoitteet* osoittavat, voiko subjekti toimia CA:na. Jos voi, varmentamisen polun pituus voidaan asettaa.

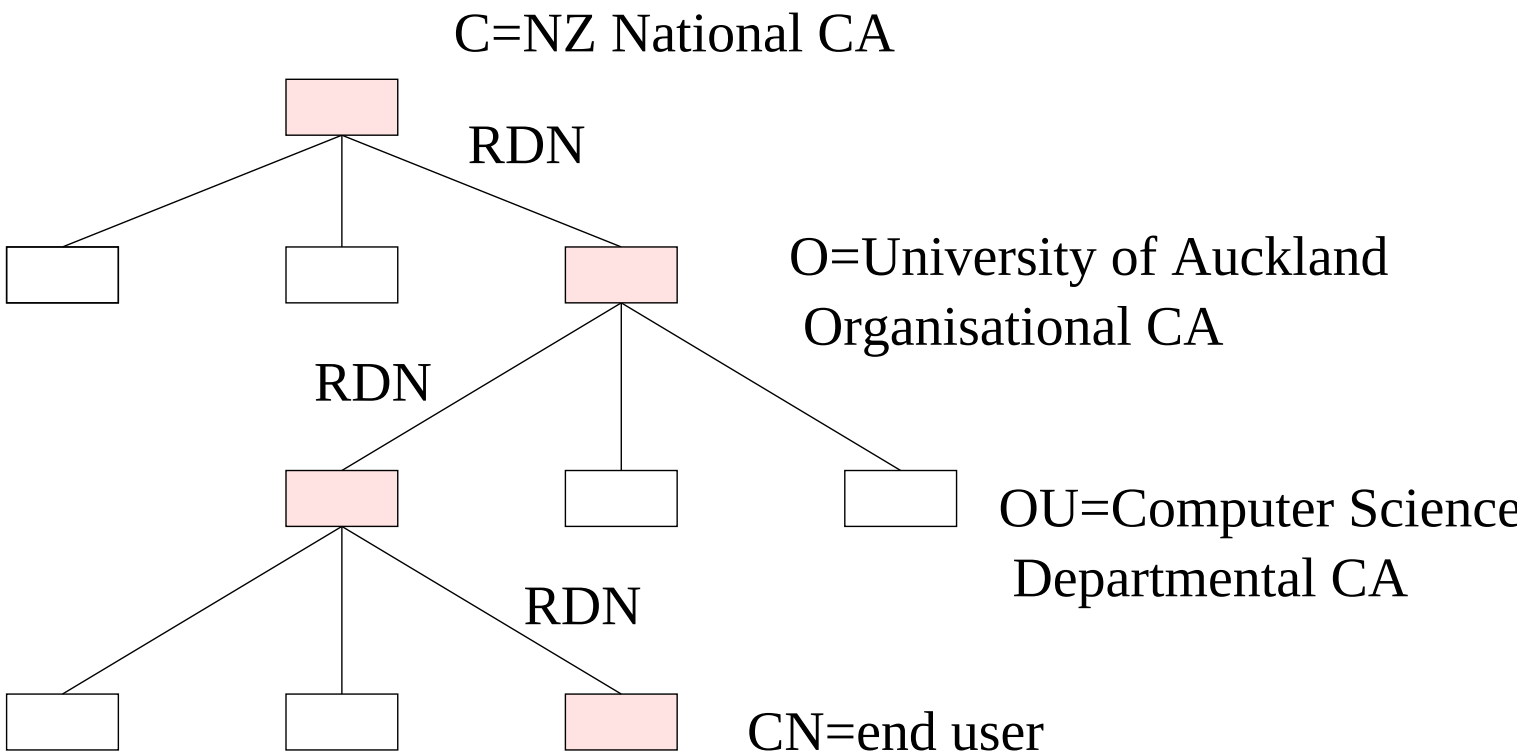
Versio 3 V

- *Nimirajoitteet* määrittelevät nimiavaruuden, josta varmennepolun subjektin tulee olla.
- *Käyttörajoitukset*.

X.509:n ongelmia I

- Seuraavassa käydään läpi julkisen avaimen infrastruktuurin, erityisesti X.509:n, ongelmia. Esitys perustuu lähteeseen [**?**].
- X.509:n alkuperäisenä tavoitteena oli ratkaista X.500-hakemiston pääsynvalvonta. X.500 perustui hierarkkiseen tietokantamalliin, jossa määriteltiin polkuja käyttäen sarjaa suhteellisia RD-nimiä (relative distinguished name, RDN).
- Sarja RD-nimiä muodostaa puolestaan D-nimen (distinguished name, DN). Polun päässä on sitten tietue, jonka attribuutit sisältävät varsinaisen tiedon. Kuviossa 2 on eräs konkreettinen tilanne.

X.509:n ongelmia II

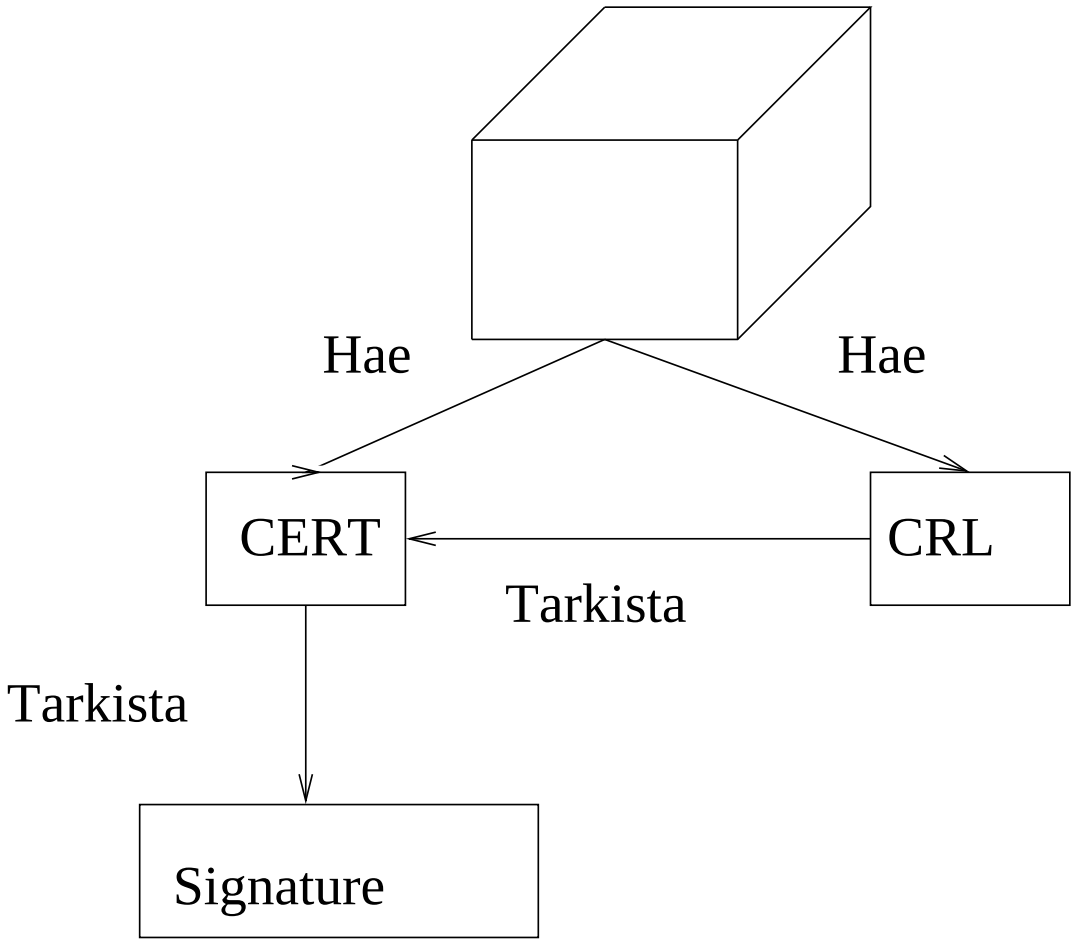


Kuva: X.509:n lähtökohta

X.509:n ongelmia III

- Pääsynvalvonnaksi suunnniteltiin erilaisia mekanismeja, salasanoista digitaalisiin allekirjoituksiin. Allekirjoitusten tapauksessa määriteltiin, että tietokannan eri osilla on omat CA:nsa, jotka luovat varmenteet pääsynvalvontatarpeisiin.
- Alkuperäisen X.509:n version 1 rakenteessa nämä lähtökohdat näkyvät hyvin selvästi. On myöntäjä DN ja subjekti DN, voimassaoloaika ja julkinen avain.
- Mikään ei osoita, kuuluuko varmenne CA:lle vai muulle (koska hakemistossa tämä määräytyy implisiittisesti).
- Ei käy myöskään selville, mihin avainta voidaan käyttää (koska alunperin oli vain yksi käyttötarkoitus, pääsynvalvonta). Ei määritelty edes minkä periaatteen mukaan varmenteet oli myönnetty.
- Nykymuodossa X.509:ää käytetään digitaalisen allekirjoituksen verifiointiin suoran todennuksen asemesta kuvion 3 mukaisesti.

X.509:n ongelmia IV



Kuva: X.509:n käyttö todennuksessa

X.509:n ongelmia V

- Varmennetta käytetään yleensä valtuutukseen hajautetussa ympäristössä: "Saako käyttäjä U suorittaa operaation O resurssiin R?" Tällainen valtuutus vaatisi kolme mekanismia: todennuksen, valtuutuksen ja jälkikäteisen tarkistuksen (audit), eli AAA:n.
- Kerberos suunniteltiin AAA:n pohjalta, mutta lopulta toteutettiin vain todennus ja puolet valtuutuksesta. (Tähän vaikutti julkisen avaimen salauksen tulo markkinoille samaan aikaan.)
- X.509 tuottaa todennuksen, muttei valtuutusta eikä jälkikäteistä tarkistusta.
- Kuvio 3 sisältää monia ongelmia. Suurin ongelma on, ettei ole selvää, mistä varmenne pitäisi hakea. (Globaalia X.500 hakemistoa ei koskaan toteutettu.)
- Ei ole myöskään selvää, mistä haetaan CRL. Käytäntö onkin nykyään, että ohjelmisto sisältää kaikki tarvittavat varmenteet.

X.509:n ongelmia VI

- Uuden varmenteen hankkiminen tapahtuu joko ulkopuolisesti (esim. postitse) tai laiskan evaluoinnin perusteella, jolloin sovellus pitää kirjaa kaikista kohtaamistaan varmenteista myöhempää tarvetta varten.
- Tämä käytäntö ratkaisee varmenteiden jakeluongelman, mutta tekee CRL:n käsittelyn paljon hankalammaksi.
- Vaikka tietäisikin, mistä hakemistosta etsiä varmenteita, ei ole selvää, mitä DN:ää tulisi käyttää tai mikä saman nimisistä on oikea valinta.
- Globaalisti yksikäsitteiset nimet eivät näytä realistisilta. X.509 tähtäsi globaaleihin nimiin, mutta DN-käsitettä ei ymmärretty käytännössä. Tämä johti lokaaleihin nimiin ja DN:t olivat mitä sattuivat. X.509v3 tosin lisäsi vaihtoehtoisen identiteetin varmenteeseen.
- Varmenteiden peruutukseen liittyvät ehkä suurimmat ongelmat:
  - Peruutuslistoja ei päivitetä tarpeeksi tiheästi.
  - Ei tiedetä, mistä CRL haetaan.

X.509:n ongelmia VII

- Haku kestää kauan.
- On tarpeen julkaista uusi CRL usein. Jos se tapahtuu minuutin välein, niin käytännössä se riittää, mutta aiheuttaa paljon rasitetta ja tarjoaa mahdollisuuden palvelunestohyökkäykselle.
- Jos tunnin tai päivän välein, niin ei hyötyä sovelluksille.

- CRL-tarkistukset ovat ilmaisia. Jos käyttäjiä on kymmenisä tuhansia, CRL:t saattavat olla monen megatavun kokoisia. Kansallisen PKI:n CRL:n koko olisi mahtava.
- Ilmaisuus johtaa siihen, ettei CRL:stä välitetä. Esimerkiksi Ruotsin kansalaisvarmenteet ovat ilmaisia, mutta CRL-tarkistus maksoi 25 senttiä 2002.
- Monessa tapauksessa varmenteet vanhenevat samaan aikaan ja kaikki hakevat CRL:n yhtäaikaa. Olisi parempi porrastaa vanhenemisajat. Miten tämä järjestettäisiin ilman, että luotettavuus vähenee, on hankalampi kysymys.

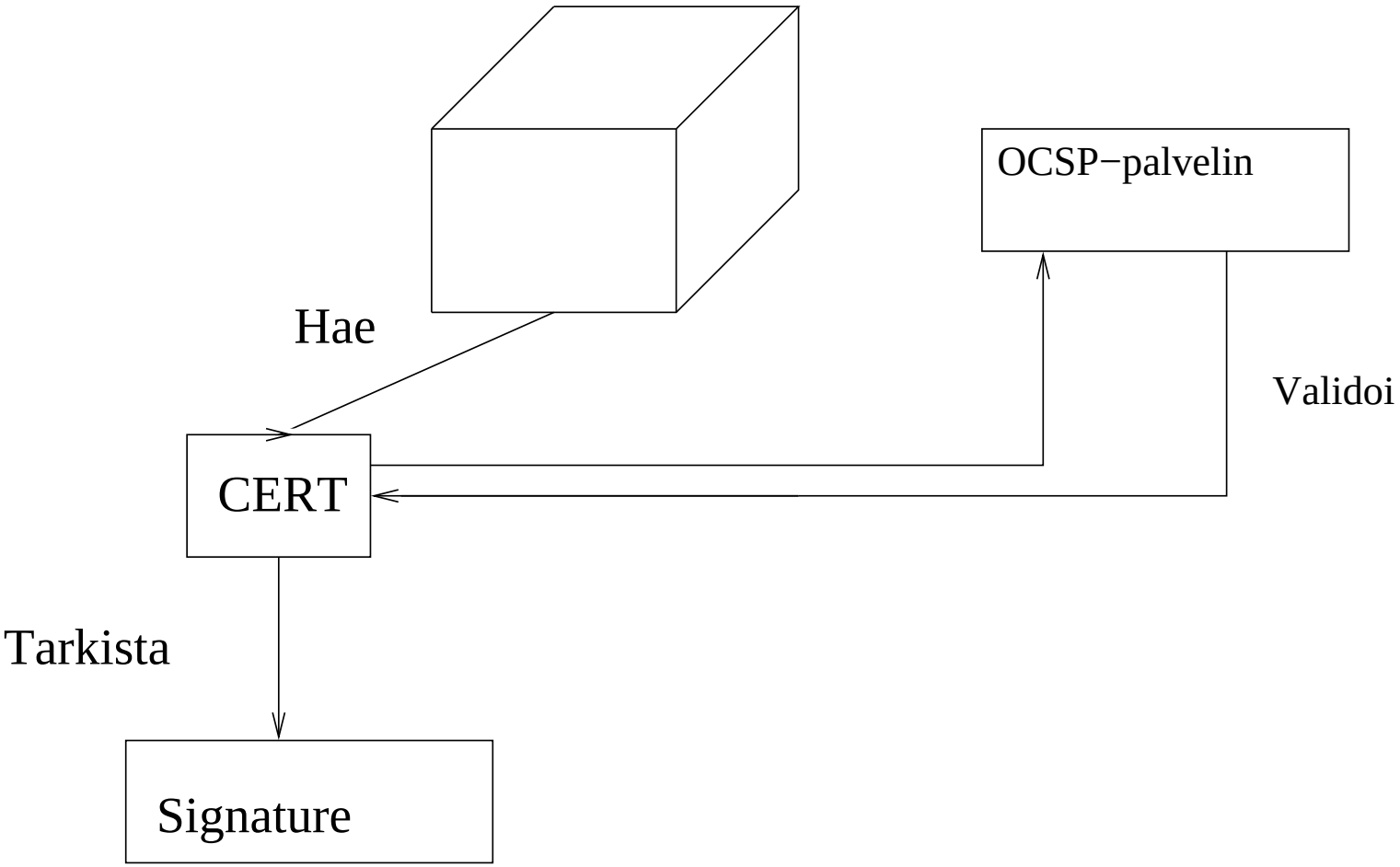
X.509:n ongelmia VIII

- Toinen mahdollisuus olisi käyttää useita päällekkäisiä peruutuslistoja. Jos tarvitsija noutaa CRL:n hetkellä n, sille annetaan hetkellä $n + 5$ vanheneva lista.
- Kolmas mahdollisuus on segmentoida CRL kiireellisyyyden perusteella. Esim. avaimen paljastuminen olisi kiireellisin, työntekijän siirtyminen toiselle osastolle vähemmän kiireellinen. Tämä saattaa aiheuttaa tietoturvaongelman, jos avaimen anastaja vaatii vastaavan varmenteen sijoittamista alempaan kiireellisyysluokkaan.
- Neljäntenä on vielä käyttää yhtä isoa, pitkäaikaista CRL:ää ja samaan aikaan joukkoa pienempiä lyhytkestoisia CRL:iä, jotka modifioivat varsinaista CRL:ää. Tästä ei näytä olevan kovin paljon käytännön kokemuksia.

Varmenteiden hallintaprotokollat I

- Varmenteiden hallintaan on kehitetty joukko protokollia. OCSP:ssä (Online Certificate Status Protocol) on palvelin, joka vastaa peruutuksia koskeviin kyselyihin.
- Palvelin perustaa vastauksensa joko CRL:ään tai johon muuhun tietoon. Kuva 4 näyttää OCSP:n toiminnan.

Varmenteiden hallintaprotokollat II



Kuva: OCSP

Varmenteiden hallintaprotokollat III

- Nyt asiakkaan ei tarvitse hakea koko CRL:ää kerralla, vaan kyselyt kohdistuvat yksittäisiin varmenteihin.
- Toisaalta asiakas ei voi valmistella CRL:n käyttöä eräajona. Jotta toiminta olisi taloudellisesti mahdollista, CA:n tulee laskuttaa OCSP-kyselyistä. Tästä syystä kyselyt tulee allekirjoittaa identifioimista varten.
- OCSP:llä on monia hyviä puolia CRL:ään verrattuna, mutta myös ongelmia. Tärkein ongelma on vastausten laatu, joka voi olla "not-revoked" (= "good" in OCSP), "revoked" ja "unknown".
- "Not-revoked" ei välttämättä merkitse samaa kuin OK. "Unknown" voi merkitä mitä tahansa väliltä "varmennetta ei ole luotu" ja "varmenne on luotu, mutta sille ei löytynyt CLR:ää".

Varmenteiden hallintaprotokollat IV

- Osittain tämä epämääräisyys johtuu alkuperäisestä CRL-mekanismista, sillä CRL voi ainoastaan antaa negatiivisen tiedon. Sekä OCSP että CRL kysyvät väärän kysymyksen "onko tämä varmenne peruutettu", kun niiden pitäisi kysyä "onko tämä varmenne voimassa".
- Tätä on yritetty oikaista toisissa protokollissa kuten SCVP (Simple Certificate Validation Protocol) ja DVCS (Data Validation and Certification Server Protocols). Harjoitustehtävänä on tutustua näihin hieman tarkemmin.
- Lisäksi on paljon muita protokollia, joiden tavoitteena on antaa tietoa varmenteiden statuksesta: ICAP (Integrated CA Services Protocol), RCSP (Real-Time Certificate Status Protocol), WebCAP (Web-based Certificate Access Protocol), PARP (Peter's Active Revocation Protocol), OpenCDP (Open CRL Distribution Process), DCS (Directory Supported Certificate Status Options).

Varmenneketjujen ongelmia I

- Varmenteiden tarkistuksissa täytyy kulkea polku lehdestä hierarkiassa ylemmällä olevaan CA:han. Tämä voi johtaa ongelmiin, kun CA:t varmentavat toisiaan.
- Voi olla esimerkiksi useita polkuja lehdestä CA:han, ja polut voivat sisältää silmukoita. Pahimmassa tapauksessa varmenteen semantiikka voi muuttua silmukan eri kierroksilla.
- *Silta-CA:t* välttävät nämä ongelmat jossain määrin lisäämällä yhden superjuuren, joka liittää kaksi tai useampia CA:ita yhteen.
- Monissa nykyisissä ohjelmistoissa on se käytäntö, että luotetaan kaikkiin varmenneviranomaisiin. Tämä mahdollistaa väärinkäytökset, sillä on helppoa perustaa uusi CA.

Varmenneketjujen ongelmia II

- Esimerkiksi selaimissa on yli sata varmenneviranomaista, joiden poistaminen vaatisi yli 600 hiiren klikkausta. Monet näistä viranomaisista ovat tuntemattomia. Ne saattavat käyttää lyhyitä julkisia avaimia ja niiden varmenteiden elinaika voi olla 40 vuotta. Saattaa myös olla, että alkuperäinen yritys on myynyt toiminnan kolmannelle osapuolelle.
- Käytössä on myös järjestelmiä, joissa varmenteen ja peruutuslistan haku yhdistetään. Voidaan myös joissain tilanteissa luopua jopa varmenteista, jos voidaan hakea julkisen avaimen kopio joltain luotettavalta osapuolelta. Tarkemmat analyysit ja esimerkit sivuutetaan.