Datan turvallisuus, yleistä tietojen luokittelusta I

Organisaation tiedot on aiheellista luokitella. Olkoon OY yhtiö, joka valmistaa elektronisia tuotteita ja harjoittaa tuotekehitystä. OY:n tiedot voitaisiin jakaa luokkiin seuraavsti:

- *Julkinen tieto* on avoinna kaikille. Tämä luokka sisältää tuotteiden määrittelyt, hintatiedot, markkinointilehtisiä yms. mikä edesauttaa OY:tä myynnissä ilman, että tuotesalaisuuksia paljastetaan.
- *Olemassaolevien tuotteiden kehityksessä tarvittava tieto* on sisäistä tietoa. Se on avoinna sekä yhtiön lakimiehille ja virkailijoille että kehittäjille. Muilta se pidetään salassa.
- *Uusien tuotteiden kehitystyössä tarvittava tieto* on vain kehittäjien saatavissa.
- *Yhtiötieto* sisältää sopimustietoja ja tietoa yhtiön toiminnoista (kuten toimista, jotka vaikuttavat pörssiosakkeisiin). Vain virkailijat ja lakimiehet pääsevät käsiksi tähän tietoon.

Datan turvallisuus, yleistä tietojen luokittelusta II

- *Asiakastieto* on asiakkaisiin liittyvää tietoa. Muun muassa luottokorttitiedot kuuluvat tähän luokkaan. Yhtiö suojelee tätä tietoa yhtä vahvasti kuin omaa tietoaan.

Datan luokittelu liittyy yleensä kiinteästi henkilökunnan luokitteluun. On siis määriteltävä henkilöryhmät ja sen jälkeen säännöt, mikä ryhmä pääsee lukemaan tai muuttamaan mitäkin tietoja. On myös mahdollista asettaa lisäehtoja erityisen tärkeille tai sensitiivisille muutoksille. Esimerkiksi voidaan vaatia, että tietyn muutoksen voi tehdä vain kahden henkilön ryhmä niin, että kummankin osasuoritus vaaditaan, ennenkuin muutos astuu voimaan.
Luottamuksellinen tieto täytyy usein salata. Salauksen lisäksi täytyy valvoa, ettei tietoa muuteta vahingossa tai laittomasti. Tarvitaan siis *eheystarkistuksia*.

Valtionhallinnon asiakirjojen luokittelu I

Suomen valtionhallinnossa asiakirjat jaetaan julkisiin ja salassa pidettäviin asiakirjoihin. Jälkimmäiset jaetaan vielä turvaluokiteltaviin ja muihin tietoihin tai asiakirjoihin. Muut tiedot ovat esimerkiksi henkilötietoja tai liikesalaisuuksia, joita ei voida kätevästi luokitella. Turvaluokiteltavat tiedot ovat joko luottamuksellisia, salaisia tai erittäin salaisia. Näitä luokkia luonnehditaan seuraavasti:

I turvaluokan asiakirja sisältää äärimmäisen arkaluonteista, salassa pidettävää tietoa. Tällainen asiakirja tulostetaan paperille, jonka poikki kulkee punainen vinoviiva ja sen jokaisen sivun ylälaitaan sijoitetaan leima tai merkintä "Erittäin salainen". Leiman väri on punainen. Tällaisen asiakirjan vastaanottajana on aina henkilö/henkilöt, eivätkä sitä saa ilman aineiston omistajan lupaa käsitellä muut kuin vastaanottajiksi merkityt sekä tällaisen asiakirjan tekniseen (vastaanotto, arkistointi yms.) käsittelyyn vastaanottavassa

Valtionhallinnon asiakirjojen luokittelu II

virastossa tai laitoksessa oikeutetut henkilöt. I turvaluokan asiakirjaa ei toistaiseksi saa lähettää sähköisissä tietojärjestelmissä. Manuaalilähetyksessä lähettäjän on aina varmistettava, että lähetys on saapunut vastaanottajaksi merkitylle henkilölle

II turvaluokan asiakirja sisältää erittäin arkaluonteista, salassa pidettävää tietoa. Tällainen asiakirja tulostetaan paperille, jonka poikki kulkee punainen vinoviiva ja sen ensimmäisen sivun ylälaitaan sijoitetaan leima tai merkintä "Salainen". Leiman väri on punainen. Tällainen asiakirja voidaan osoittaa henkilölle tai organisaatiolle ja sitä saavat käsitellä vain ne henkilöt, jotka on virastossa oikeutettu käsittelemään salassa pidettäviä asioita. II turvaluokan asiakirjan saa lähettää vastaanottajalle sähköisissä tietojärjestelmissä ainoastaan riittävän vahvasti salattuna.

Valtionhallinnon asiakirjojen luokittelu III

III turvaluokan asiakirja sisältää salassa pidettävää tietoa. III turvaluokan asiakirja tulostetaan normaalille paperille ja sen ensimmäisen sivun ylälaitaan sijoitetaan leima tai merkintä ”Luottamuksellinen”. Leiman väri on punainen. Tällainen asiakirja voidaan osoittaa henkilölle tai organisaatiolle ja sitä saavat käsitellä vain ne henkilöt, jotka tehtävässään tarvitsevat kyseisen asiakirjan sisältämiä tietoja. III turvaluokan asiakirjan saa lähettää vastaanottajalle sähköisissä tietojärjestelmissä riittävän vahvasti salattuna

Tiedostojen poisto

Kun tiedosto poistetaan, joko tiedoston tieto tai tiedoston nimi poistetaan. Näillä kahdella on oleellinen ero.

Määritelmä

Suora alias (direct alias, Unixissa kova linkki) on hakemistoalkio, joka osoittaa tiedostoon. Epäsuora alias (symbolinen linkki) on hakemistoalkio, joka osoittaa erityiseen tiedostoon, joka puolestaan sisältää kohdetiedoston nimen. Käyttöjärjestelmä tulkitsee epäsuoran aliaksen sijoittamalla erityistiedostossa olevan nimen epäsuoran aliaksen hakemistoalkioon.

Kaikki suorat, saman tiedoston nimeävät aliakset ovat samanarvoisia. Jokainen suora alias on saman tiedoston vaihtoehtoinen nimi.

- Tiedostojen sijainti hakemistossa vaikuttaa turvallisuuteen.
- Jos jokaisella suoralla aliaksella on eri oikeudet, tiedoston omistajan täytyy muuttaa jokaisen aliaksen pääsyoikeudet.
- Välttääkseen tämän useimmat systeemit liittävät tiedostoattribuuttien sisältämän tiedon varsinaiseen dataan, jolloin hakemistoalkiot sisältävät osoittimen tiedostoattribuuttitauluun.
- Kun käyttäjä poistaa tiedoston, hakemistoalkio poistetaan. Systeemi ylläpitää tietoa tiedostoon liittyvistä tiedostoattribuuteista ja kun näiden lukumäärä tulee nollaksi, tiedoston varaama tila vapautetaan. Toisin sanoen tiedoston tuhoaminen ei takaa, etteikö tiedostoon voisi vielä päästä käsiksi.

Esimerkki I

- A käyttää Unix-pohjaista systeemiä. Hänellä on ohjelma runasa, jonka setuid kuuluu A:lle. A haluaa tuhota tiedoston niin, ettei kukaan voi enää lukea sitä.
- Jos hän käyttää komentoa `rm runasa`, tiedostoon liittyvä hakemistoalkio tuhotaan. Jos kenelläkään ei ole suoraa aliasta (kovaa linkkiä) tiedostoon, tiedosto poistetaan systeemistä.
- B on kuitenkin tehnyt suoran aliaksen tiedostoon. A on poistanut tiedoston, mutta yksi tiedostoon liittyvä hakemistoalkio on edelleen olemassa, joten tiedostoa ei poisteta. B voi edelleen ajaa ohjelmaa `runasa`. Koska ohjelman setuid on A, ohjelma käyttää A:n, ei B:n oikeuksia.
- Unixissa A voi tuhota B:n tiedostoja vain, jos B on antanut A:lle kirjoitusoikeudet hakemistoon. Jos A haluaa, ettei kukaan toinen voi ajaa hänen ohjelmaansa, hänen täytyy ensin muuttaa oikeuksia ja vasta sitten poistaa tiedosto eli

Esimerkki II

```
chmod 000 runasa
rm runasa
```

Ensimmäinen komento poistaa kaikki oikeudet tiedostoon, mukaan lukien setuid-oikeuden. Toisen komennon jälkeen B säilyttää kylläkin hakemistoalkionsa, tiedosto on edelleen olemassa, mutta kukaan ei voi sitä käyttää. □

Kun tiedosto poistetaan, sen varaama tila palautetaan käyttämättömien lohkojen joukkoon. Kuitenkin tieto jää lohkoihin ja jos hyökkääjä kykenee lukemaan noita lohkoja, hän pääsee käsiksi luottamuksellisiin tietoihin. Sen tähden arkaluonteiset tiedot on ensin pyyhittävä pois ennen tiedoston poistamista.

Esimerkki. Monissa Windowsin ja Macintoshin systeemiohjelmissa on mekanismi, joka pyyhkii tiedoston ennen sen poistamista. Nämä mekanismit kirjoittavat tiedostoon vanhan tiedon päälle tietyn bittijonon. On mahdollista määrätä, mitä bittijonoja käytetään ja kuinka monta kertaa päälle kirjoitetaan. Joissakin Unixin rm-versioissa on sama ominaisuus. □

Epäsuorat aliakset ja tiedoston poisto I

Kolmas seikka, joka täytyy ottaa huomioon tiedostojen poistamisessa, on epärien aliasten olemassaolo. Kun tiedostoon liittyvä komento suoritetaan, vaikutus voi riippua siitä, onko komennon kohteena suora vai epäsuora alias. Pahimmassa tapauksessa käyttäjä uskoo, että tietty tieto on suojattu tai poistettu, vaikka suojaus tai poisto kohdistuukin epäsuoraan aliakseen eikä tietoon itseensä.

Esimerkki. Oletetaan, että A lisää tiedoston lukuoikeuden L:lle. Jos hakemistoalkio on suora alias, L voi lukea tiedoston. Mutta entä jos hakemistoalkio on epäsuora alias? Vastaus riipuu systeemistä. Red Hat Linux 7.1:ssä chmod komento muutti epäsuoran aliaksen osoittaman tiedoston oikeuksia, kun taas rm poistaa epäsuoran aliaksen. □

Tiedostojen kopiointi ja siirto I

Tarkastellaan kahta esimerkkiä

Esimerkki. Unixin komento

```
cp xyzzy plugh
```

kopioi ensimmäisen tiedoston toiseen. Jos plugh ei ole olemassa, komento luo sen ja asettaa uuden tiedoston oikeudet samaksi kuin xyzzy:n. Kuitenkin setuid- ja setgid-attribuutteja ei oteta huomioon. Jos plugh on jo olemassa, komento kopioi xyzzy:n myös siihen. Tämä voi olla turvallisuusongelma, jos xyzzy ei ole kaikkien luettavissa, mutta plugh on. □

Samanlaisia ongelmia voi syntyä mv-komennosta.

Esimerkki. Tarkastellaan komentoa

```
mv plugh /usr/ab/advent
```

Tiedostojen kopiointi ja siirto II

Jos hakemistoalkio sijaitsee samassa tiedostosysteemissä, suora alias poistetaan nykyisestä hakemistostaan ja viedään hakemistoon `usr/ab/advent` Muussa tapauksessa mv tekee operaatiot

```
cp plugh /usr/ab/advent/plugh
rm plugh
```

Ensimmäisessä tapauksessa plugh-tiedoston oikeudet säilyvät. Toisessa tapauksessa ne voivat muuttua, kuten edellisessä esimerkissä nähtiin. □

Tiedon eheyden käsite ja eheystekniikoita I

- On tärkeää, että tieto ei muutu tai että tietoa ei muuteta, jos sen on tarkoitus pysyä vakiona. Tiedon eheyteen joudutaan kiinnittämään huomiota erityisesti tietoliikenteessä, mutta myös muistissa olevan tiedon varmistaminen tulee joskus kysymykseen.
- Tavalliset tarkistussummat eivät riitä tietoturvatarkasteluissa, koska ne eivät suojaa tiedon tahalliselta muuttamiselta (hyökkääjä voi muuttaa myös tarkistussummaa). Tarvitaan menetelmä, joka paljastaa tahattomat ja tahallaan tehdyt muutokset.
- Usein eheyteen liitetään vielä todennus: tiedon vastaanottaja pystyy varmasti näkemään, kuka tiedon on lähettänyt. Hyvä todennus toteuttaa lisäksi ehdon, ettei vastaanottaja voi itse väärentää sanomaa ja väittää, että se on tullut toiselta osapuolelta.

Tarkastellaan seuraavia menetelmiä, joilla toteutetaan eheys ja todennus:

- salaus,
- tiivistefunktiot ja digitaalinen allekirjoitus,
- MAC-funktiot.

Tiedon salaus eheystekniikkana I

- Jos selvätekstiä sisältävä tieto salataan ja salattua tietoa muutetaan, salauksen purku tuottaa mielivaltaisen bittijonon, joka ei ole selvätekstin esitystä. Näin salauksen purku paljastaa muutoksen ja salaus toimii eheyden takaajana. Lisäksi salaus todentaa lähettäjän, jos salausavain on yhteisesti sovittu, eikä se ole vuotanut ulkopuolisille.
- Voi olla kuitenkin vaikeaa automaattisesti päätellä, onko salauksen purun lopputulos oikeanlaatuista, jos alkuperäinen tieto on esimerkiksi binäärikoodia, röntgenkuvia tms. Tällaisissa tilanteissa tietoon voidaan liittää tarkistussumma (esimerkiksi CRC) ja vasta sitten salata. Toinen mahdollisuus on strukturoida tieto siten, että struktuuri on helppo havaita automaattisesti.

Tiedon salaus eheystekniikkana II

- Salaus ei kuitenkaan estä vastaanottajaa väärentämästä sanomaa, salaamasta sitä ja väittämästä, että se on tullut toiselta. Usein tietokoneverkkosovelluksissa osapuolilla ei ole valmiina yhteisiä salaisia avaimia, vaan niistä pitää sopia aluksi. Tämä hankaloittaa salauksen käyttöä eheyden takaajana.
- Myöskään aina ei tarvita luottamuksellisuutta, vaan tieto voitaisiin lähettää selväkielisenä perille, jos vain se menisi vastaanottajalle ehyenä. Sen tähden on hyvä erottaa luottamuksellisuus ja eheys/todennus toisistaan ja käsitellä niitä eri proseduureilla.

Tiivistefunktiot I

- Tiivistefunktio (engl. hash function) on kuvaus h, joka laskee sanomasta M tarkistuskentän $h(M)$.
- M voi yleensä olla vaihtuvamittainen, sen sijaan $h(M)$ on kiinteämittainen. Normaalisti $h(M)$:n pituus on paljon lyhyempi kuin M:n.
- Tiivistefunktion h arvon laskemisessa ei tarvita salaisia avaimia. Funktio on kuitenkin yksisuuntainen eli $h^{-1}(X)$ on vaikea laskea, vaikka h ja X tunnetaan.
- Tiivistefunktioita voidaan käyttää eheyden ja todennuksen yhteydessä. Tyypillisesti sanomasta lasketaan ensin tiivistefunktiolla tiiviste. Sen jälkeen tiiviste allekirjoitetaan digitaalisesti. Lähtevä sanoma koostuu selväkielisestä sanomasta plus allekirjoituksesta.

Tiivistefunktiot II

- Vastaanottaja laskee sanomasta myös tiivisteen ja sen jälkeen verifioi allekirjoituksen. Verifiointi tarkoittaa yleensä sitä, että allekirjoituksesta lasketaan alkuperäinen tiiviste. Nyt voidaan verrata itse laskettua tiivistettä ja allekirjoituksesta saatua tiivistettä. Jos nämä ovat samoja, tieto on tullut eheänä perille ja allekirjoitus vielä vahvistaa lähettäjän. Myöskään vastaanottaja ei voi väärentää allekirjoitusta, joten lähettäjän ei tarvitse pelätä väärennöksiä.
- Mikäli luottamuksellisuutta ei tarvita, salausta yritetään välttää useista syistä:
  - Salausohjelmat ovat hitaita.
  - Piiritason AES- yms. toteutukset ovat melko tehokkaita, mutta näidenkin kustannukset tuntuvat, jos salausta tarvitaan kaikissa verkon solmuissa.
  - Salauslaitteisto on optimoitu suurten datamäärien salaukseen. Jos salattavana on pieni lohko, suurin osa ajasta menee alustukseen.

Tiivistefunktiot III

- Salausalgoritmi voi olla patentoitu kuten oli RSA:n tapauksessa. Tämä lisää kustannuksia.
- Salausalgoritmen vientiä säädellään (nykyään paljon vähemmän kuin esim. 1980-luvulla).

Tiivistefunktioilta vaadittavat ominaisuudet I

Seuraava luettelo listaa tiivistefunktioilta vaadittavat ominaisuudet:

1. $h(M)$ voidaan laskea minkä pituiselle M tahansa.
2. $h(M)$ on kiinteän pituinen.
3. $h(M)$ on helppo laskea.
4. Jos annetaan y, ei ole helppoa löytää sellaista M:ää, että $h(M) = y$. Eli h on *alkukuvaresistentti* (h has preimage resistance).
5. Jos on annettu y ja M_1, ei ole helppoa löytää sellaista $M_2 \neq M_1$, että $h(M_1) = h(M_2)$. Eli h on *injektiotyyppinen* (h has second preimage resistance).
6. On vaikeaa löytää mitään paria (M, M'), jolle $h(M) = h(M')$. Eli h on *törmäysresistentti* (collision resistance).

Käytännön tiivistefunktio toteutetaan samaan tapaan kuin symmetrinen salaus. Erona on, että lopputulosta ei tarvitse enää purkaa niinkuin salauksessa. Siten teksti jaetaan lohkoihin, lohkoja yhdistellään binäärioperaatioilla, välillä suoritetaan rekisterien sivuttaissiirtoja jne.

Tiivistefunktioita

- Tilanne tiivistefunktioiden kanssa on tällä hetkellä sekava. Kiinalainen tutkimusryhmä kehitti noin 6 vuotta sitten uudenlaisen tekniikan murtaa tiivistefunktioita, ja tämä johti siihen, että perinteellisiä tiivistefunktioita MD5, SHA-1, SHA-2, RIPEMD-160 ei pidetä enää turvallisina.
- Paraikaa ollaan kehittämässä SHA-3:ta, ja sen pitäisi tulla markkinoille ensi vuonna.
- MD5 (pituus 128 bittiä) on ehdottomasti vanhentunut, SHA-1:tä (160 b) ei myöskään ole suositeltu enää vähään aikaan. Sen sijaan SHA-2 -versiot (224, 256, 384, 512 bittiä) toiminevat vielä käytännössä jonkin aikaa.

MAC-funktiot I

- MAC-koodi (Message Authentication Code) on tiiviste, joka liitetään sanomaan, jotta vastaanottaja voisi varmistaa alkuperäisen lähettäjän. Tiivisteen laskemisessa käytetään salaista parametria.
- Ensimmäiset MAC-koodit generoitiin salauksen avulla. Tällöin vastaanottajalla ja lähettäjällä täytyi olla sama salainen avain. Salalohkot laskettiin vielä yhteen, joten lopputulos oli salalohkojen summa.
- Toinen tapa muodostaa MAC on käyttää tavallista tiivistefunktiota salaisen avaimen kanssa. Toisin sanoen sanomasta lasketaan tiiviste, mutta laskennassa käytetään myös salaista avainta pelkän sanoman lisäksi. Esimerkiksi MD5:ta ja SHA:ta on käytetty tällä tavoin MAC:in muodostamiseen.
- Sitten on vielä esimerkiksi MAA-algoritmi, joka on ISO-standardi 8731-2. Sen laskenta muistuttaa tiivisteen laskemista salaisen avaimen kera.

MAC-funktiot II

- MAC-menetelmän avulla tiedon eheys voidaan varmistaa, sillä kukaan kuin salaisen avaimen omistajat eivät pysty laskemaan tiivistettä. Toisaalta MAC ei estä vastaanottajaa väärentämästä sanomaa ja MAC arvoa. Näyttääkin siltä, että digitaalinen allekirjoitus on suosituin menetelmä, joka samalla takaa eheyden, varmistaa lähettäjän ja estää myös vastaanottajan väärennökset.

Tietokonesalaus

Ehdot symmetriselle (osapuolilla sama salainen avain) tietokonesalaukselle:

- Avain on kohtuullisen kokoinen ($<$ 250 bittiä kaupallisissa sovelluksissa, sotilaallisissa voi olla suurempikin).
- Avainta on voitava käyttää useaan kertaan.
- Salaus on nopeaa ja salauspiirit halpoja.
- Salauksella tulee olla hyvät *diffuusio-ominaisuudet* eli yksi selvätekstin bitti vaikuttaa moneen salatekstin bittiin niin, että selvätekstin tilastollinen rakenne häviää.
- Lisäksi salauksella tulee olla hyvät *sekoitusominaisuudet* eli on vaikeaa päätellä, miten salatekstin tilastollinen rakenne riippuu selvätekstin tilastollisesta rakenteesta.

Salausmenetelmien luokittelu

Symmetrinen tietokonesalaus on joko jonosalausta tai lohkosalausta.

- Jonosalaus tapahtuu merkki merkiltä: selvätekstin merkkiä muutetaan ja se ketjutetaan lähettävään salamerkkien virtaan. Tunnetuin jonosalaaja on RC4 ja sen uusi versio RC5.
- Jonosalausta käytetään tilanteissa, joissa salauksen on oltava nopeaa ja reaaliaikaista tiedon synnyn kanssa. Esimerkiksi GSM-puheluissa on käytetty jonosalausta A5/1, A5/2.
- Jonosalausta ei ole kuitenkaan pidetty täysin turvallisena, josta syystä siitä ei ole edes standardeja.
- Lohkosalauksessa selväteksti jaetaan lohkoihin (128 b, 256 b, 512 b), jokainen lohko salataan samalla salaisella avaimella ja lohkot lähetetään joko sellaisenaan tai ketjutettuna.
- Tällä hetkellä vallitseva standardi on AES.

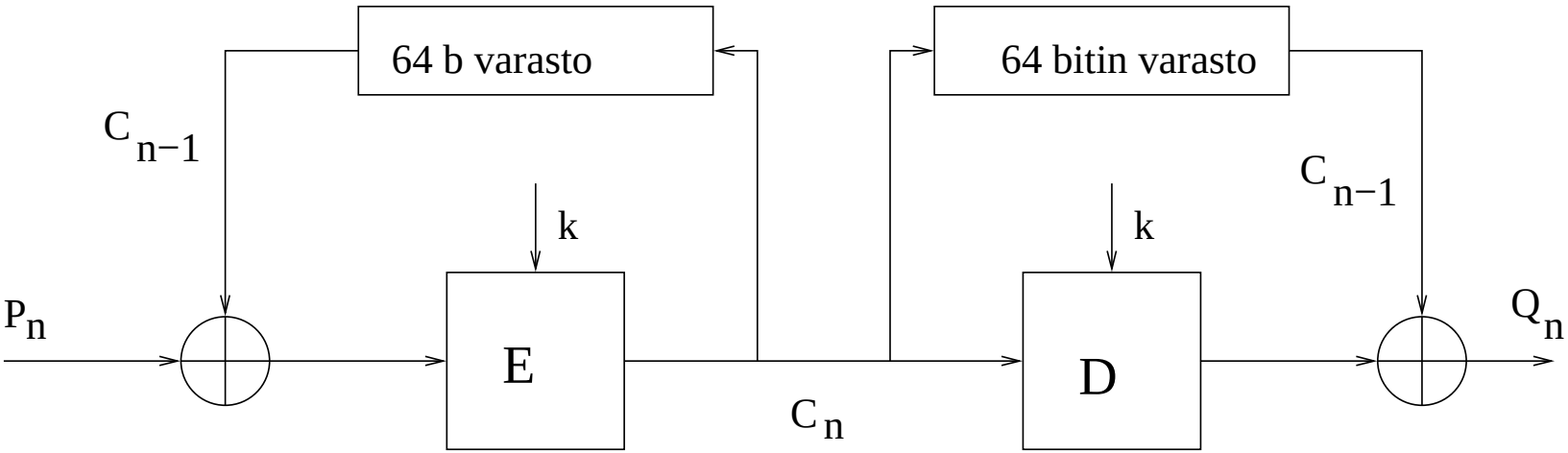
Symmetrinen salaus käytännössä: ECB I

- Yksinkertaisimillaan salausta käytetään siten, että selväteksti jaetaan esim. 128:n (AES) bitin lohkoihin, jokainen lohko salataan erikseen ja lohkot lähetetään vastaanottajalle.
- Kysymyksessä on ns. elektroninen koodikirja, ECB.
- Tällä menetelmällä on kuitenkin huonoja puolia:
  - Säännöllisyydet saattavat näkyä salatekstissä.
  - Esimerkiksi rahansiirrossa summa saattaa olla aina samalla paikalla. Suurista summista voi olla tietoa jne.
- Jotta esitetyiltä ongelmilta vältyttäisiin, pyritään salalohkot ketjuttamaan niin, että yksi salalohko vaikuttaa kaikkien muiden seuraavaksi tulevien koodaukseen.
- Näin sekoitusominaisuudet paranevat, eikä vakio-osia ole enää mahdollista paikantaa. Näitä ns. *ketjutustekniikoita* voidaan käyttää minkä tahansa symmetrisen lohkosalausmenetelmän kanssa.

Salalohkojen ketjutus I

- Salalohkojen ketjutuksessa (engl. cipher block chaining, CBC) selväteksti jaetaan lohkoiksi, joita ruvetaan salaamaan järjestyksessä.
- Nyt kuitenkin käytetään apuna yhteenlaskua modulo 2. Aina kun lohko on salattu, salattu lohko lasketaan yhteen modulo 2 seuraavan selvätekstilohkon kanssa, joka salataan vasta tämän jälkeen.
- Kaaviona salaus- ja purkuprosessi ovat seuraavan kuvan mukaisia (lohkon pituus kuvioissa 64, todellisuudessa esim. 128).

Salalohkojen ketjutus II



Kuva: CBC

Vastaavasti kaavoina:

Salalohkojen ketjutus III

$$\begin{aligned} C_n &= E_K(P_n \oplus C_{n-1}), \\ Q_n &= D_K(C_n) \oplus C_{n-1}, \\ D_K(C_n) &= P_n \oplus C_{n-1}, \\ Q_n &= P_n \oplus C_{n-1} \oplus C_{n-1} = P_n. \end{aligned}$$

- Ensimmäisen ja viimeisen lohkon käsittely vaatii erikoiskäsittelyä. Otetaan käyttöön 64 bitin *alustusmuuttuja* I, jota käytetään ensimmäisen selvälohkon salauksessa:

$$C_1 = E_K(P_1 \oplus I), \quad Q_1 = D_K(C_1) \oplus I.$$

- Yleensä alustusmuuttuja I on salainen. Viimeinen selvälohko on täydennettävä 64-bittiseksi. Tämä voidaan tehdä lisäämällä nollia tai satunnainen bittijono.

Ketjutuksessa tiedonsiirtovirheet leviävät laajemmalle kuin elektronisen koodikirjan tapauksessa. Oletetaan, että n. salalohkossa tapahtuu yhden bitin tiedonsiirtovirhe. Merkitään

- C_n virheetön salalohko,
- C'_n yhden bitin virheen sisältävä salalohko,
- Q_n selvälohko purkamisen jälkeen,
- $\tilde{Q}_n$ täysin virheellinen lohko purkamisen jälkeen,
- Q'_n yhden bitin virheen sisältävä selvälohko.

Nyt tiedonsiirtovirheen vaikutus selviä seuraavista kaavoista:

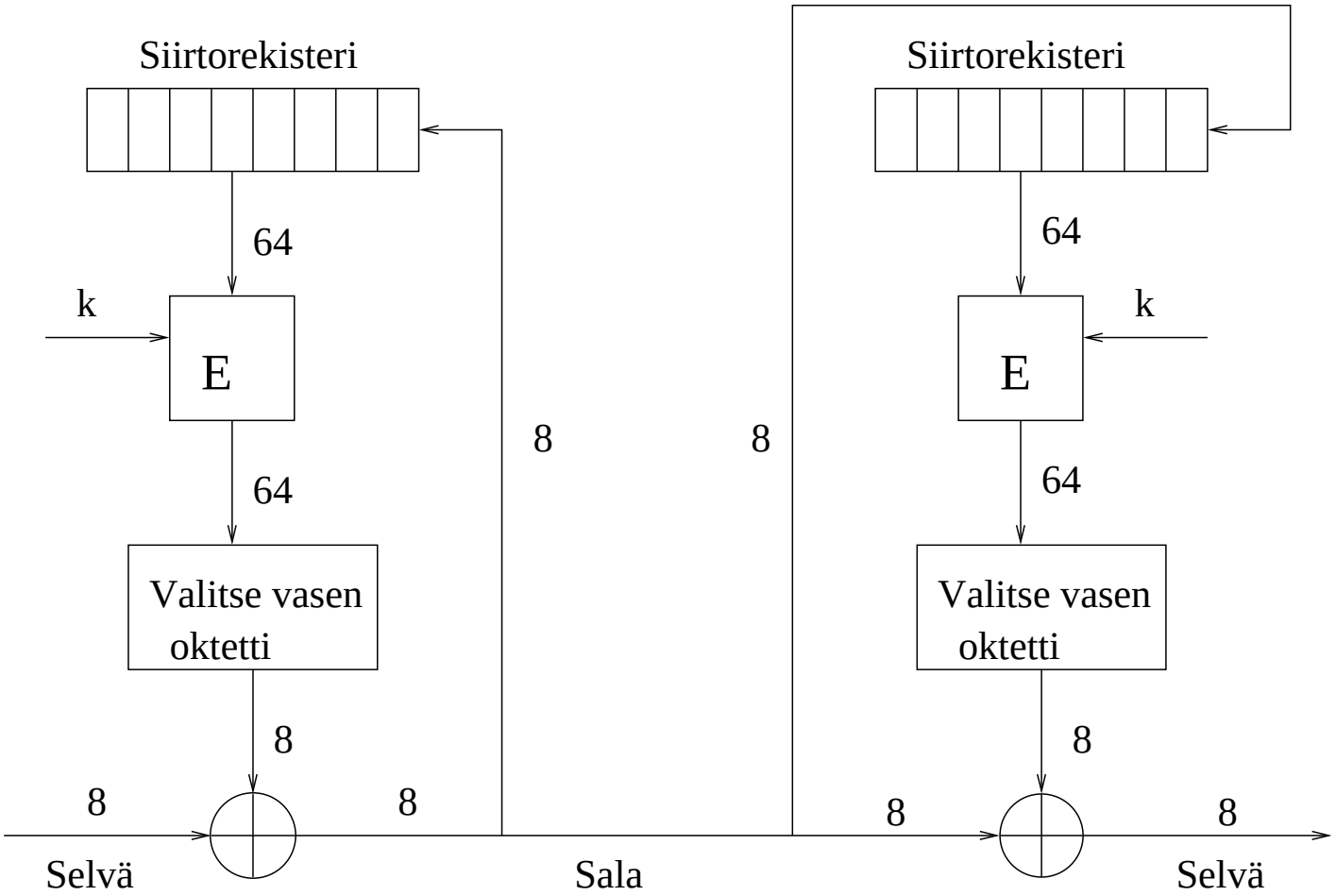
$$\begin{aligned} D_K(C'_n) \oplus C_{n-1} &= \tilde{Q}_n, \\ D_K(C_{n+1}) \oplus C'_n &= Q'_{n+1}, \\ D_K(C_{n+2}) \oplus C'_{n+1} &= Q_{n+2}. \end{aligned}$$

Eli yksi selvälohko tuhoutuu täysin ja yhdessä on yhden bitin virhe. Muut lohkot selviävät vaurioitta.

Salauksen takaisinkytkentä I

Salauksen takaisinkytkentää (engl. cipher feedback chaining, CFC) käytetään, kun salaus tapahtuu merkki merkiltä tai bitti bitiltä. Oletetaan, että merkin pituus on m bittiä. Yleensä $m = 8$. Menetelmän idea käy selville seuraavasta kaaviosta.

Salauksen takaisinkytkentä II



Kuva: CFC

Salauksen takaisinkytkentä III

Ensimmäisen oktetin kohdalla käytetään alustusmuuttujaa I, jonka pituus on sama kuin salausjärjestelmän lohkon pituus. Alustusmuuttuja sijoitetaan valmiiksi siirtorekisteriin. Se on luonnollisesti vaihdettava tarpeeksi usein. Edellisten merkintöjen lisäksi merkitään:

- S_n siirtorekisterin sisältö n. kierroksella,
- L_n on $E_K(S_n)$:n m vasenta bittiä.

Analysoidaan taas tiedonsiirtovirheen vaikutusta. Oletetaan, että n. kierroksella tapahtuu yhden bitin virhe salalohkossa. Tällöin kun $M = 8$:

- S_n ja L_n kunnossa, mutta $C'_n \oplus L_n = Q'_n$ eli selvälohkoon tulee n. kierroksella yhden bitin virhe;
- kierroksilla $n+1, \cdots, n+8$ tilanne on S'_i ja $\tilde{L}_i$, $i = n+1, \cdots, n+8$;
- Siten

$$C_i \oplus \tilde{L}_i = \tilde{Q}_i,$$

$i = n+1, \cdots, n+8$;

Salauksen takaisinkytkentä IV

- kierroksella $n + 9$ kaikki on kunnossa.

Siis virhe vaikuttaa 9 oktettiin.

Laskurimoodi I

Laskurimoodi eli CTR on saavuttanut suosiota viime aikoina, joskin se on vanha ehdotus. Salaus tapahtuu nyt muodossa

$$C_i = P_i \oplus E(K, L_i),$$

missä

- P_i on i.s selvälohko,
- L_i on laskurin arvo i. kierroksella,
- K on salainen, symmetrinen avain ja
- $\oplus$ on XOR-operaatio.

Tyypillisesti laskurilla on jokin sovittu alkuarvo, joka kasvatetaan joka kierroksella yhdellä. Mitään ketjutusta ei ole käytössä.
Menetelmällä on etuja:

- Laitteistotason tehokkuus.

Laskurimoodi II

- Ohjelmallinen tehokkuus.
- Esiprosessointi mahdollista (salaus).
- Lohkot voidaan prosessoida satunnaisessa järjestyksessä.
- CTR:n voidaan näyttää olevan ainakin yhtä vahva kuin muut ketjutusmenetelmät, jotka ovat olleet esillä.
- Tarvitaan vain salauksen toteutus, ei purun toteutusta.