Lohkosalakirjoitus

- Salattava tieto jaetaan tasapituisiin lohkoihin. Kaikki lohkot salataan samalla funktiolla.
  - Lohkonpituus on yleensä 64 bittiä.
  - Liian lyhyestä lohkosta on mahdollista kerätä sanakirja.
  - Liian pitkän lohkon sekoittaminen vaatii liikaa vaivaa.
  - 2:n potenssi, puolilohko 32 bittiä yleinen sananpituus tietokoneissa.
- Salausfunktio on mahdollisimman epälineaarinen, satunnaisen näköinen bijektio lohkoilta toisille.

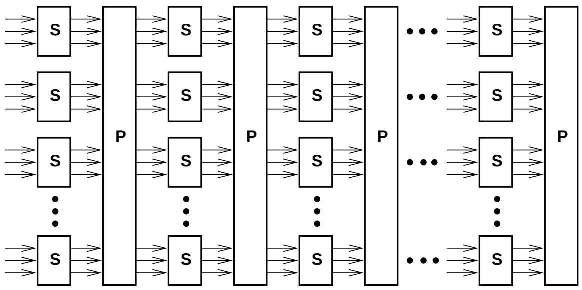
  $$E_K : 2^{64} \longrightarrow 2^{64}.$$

  (Bijektio joukolta itselleen = permutaatio. Kaikki mahdolliset salausfunktiot muodostavat permutaatioryhmän.)
- Satunnaisesti valittu permutaatio 2^{64} alkion taulukossa olisi paras, muttei mahdollinen. (monikobittinen avain?)

  Ongelma: miten esittää mahdollisimman hyviä salausfunktioita mahdollisimman kompaktissa ja nopeasti laskettavassa muodossa?

  Ratkaisu: iteroidaan vuoronperään konfuusiota ja diffuusiota.

- Verkko substituutioita ja permutaatioita (*product cipher*)
  1. Substituutiossa korvataan lohko bittejä toisilla $\longrightarrow$ konfuusio.
  2. Permutaatiossa sekoitetaan eri lohkojen bitit keskenään $\longrightarrow$ diffuusio.



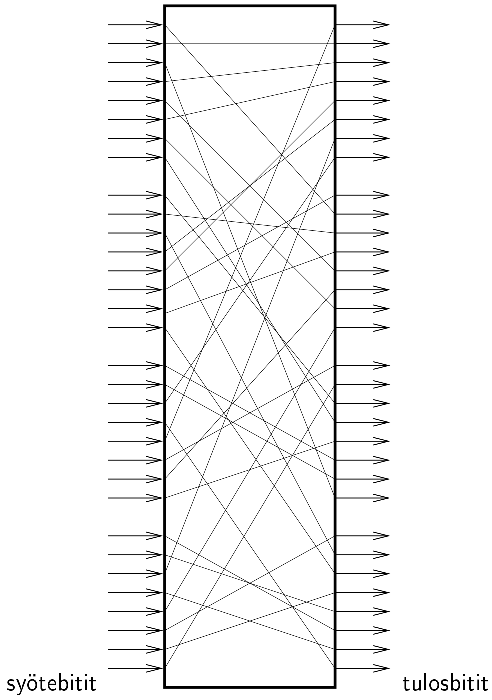
- Algoritmi on julkinen, mutta substituutioihin (ja permutaatioihin) vaikutetaan salaisella avaimella.
- Jos substituutiot ovat bijektiivisiä, on koko algoritmi bijektiivinen ja käänteisfunktio D helposti lastettavissa.

Substituutio

lohko	S(lohko)
0000	0110
0001	1011
0010	1111
0011	0010
0100	0010
0101	0101
0110	1110
0111	0000
1000	1100
1001	1010
1010	0100
1011	1001
1100	0011
1101	0111
1110	1101
1111	1000

- Taulukkototeutus tehokas ohjelmallisesti.
- Substituution ei tarvitse olla bijektio.
- $S(lohko)$ ei aina saman pituinen kuin $lohko$.
- Hyviä substituutioita voidaan tuottaa arpomalla tai esim. π:n desimaaleista.

Permutaatio



- Permutaatio toteutetaan tehokkaasti raudalla johtoja kytkemällä. Ohjelmallisesti se on hidas.

Feistel-salakirjoitus

- Konfuusio ja diffuusio kätketään sekoitusfunktion f sisään.
- Sekoitusfunktio riippuu joka kierroksella eri avaimesta.

$$\begin{array}{l} L_0, R_0 = P \\ \left.\begin{array}{l} L_i = R_{i-1} \\ R_i = L_{i-1} \oplus f(R_{i-1}, K_i) \end{array}\right\} i = 1, \ldots, n \\ C = L_n, R_n \end{array}$$

- Käänteisfunktio helppo konstruoida. f:llä ei tarvitse olla käänteisfunktiota tai sitä ei tarvitse tietää.

$$\begin{aligned} &L_0, R_0 = P \\ &L_1 = R_0 \\ &R_1 = L_0 \oplus f(R_0, K_1) \\ &L_2 = R_1 \\ &R_2 = L_1 \oplus f(R_1, K_2) \\ &L_3 = R_2 \\ &R_3 = L_2 \oplus f(R_2, K_3) \\ &C = L_3, R_3 \end{aligned}$$

Feistel-salakirjoitus

$$\boxed{\begin{array}{l} x_{-1}, x_0 = P \\ x_i = x_{i-2} \oplus f(x_{i-1}, K_i), \quad i = 1, \ldots, n \\ C = x_{n-1}, x_n \end{array}}$$

- Salaus

$$\begin{aligned} &x_{-1}, x_0 = P \\ &x_1 = x_{-1} \oplus f(x_0, K_1) \\ &x_2 = x_0 \oplus f(x_1, K_2) \\ &x_3 = x_1 \oplus f(x_2, K_3) \\ &x_4 = x_2 \oplus f(x_3, K_4) \\ &C = x_3, x_4 \end{aligned}$$

- Avaus

$$\begin{aligned} &P = x_{-1}, x_0 \\ &x_{-1} = x_1 \oplus f(x_0, K_1) \\ &x_0 = x_2 \oplus f(x_1, K_2) \\ &x_1 = x_3 \oplus f(x_2, K_3) \\ &x_2 = x_4 \oplus f(x_3, K_4) \\ &x_3, x_4 = C \end{aligned}$$

Salausfunktioiden algebrallinen rakenne

- Jos salausfunktiot E_k muodostavat **ryhmän**, salaaminen peräkkäin kahdella avaimella K_1, K_2 vastaa aina salaamista jollakin yhdellä avaimella K_3:

  $$E_{K_2}(E_{K_1}(P)) = E_{K_3}(P).$$

  Tällöin moninkertainen salaus ei ole sen turvallisempaa kuin yksinkertainen.
- Algebralliset säännönmukaisuudet eivät koskaan ole hyväksi, edes likimääräisinä.
- Joskus pienellä poikkeamalla säännöllisyydestä saadaan suuri parannus turvallisuuteen — yleensä häviävän pieni tilastollinen säännönmukaisuuskin on turmiollista.

Yksinkertaiset relaatiot avainten välillä

$$E_K(P) = C \ \Rightarrow\ E_{f(K)}(g(P, K)) = h(C, K).$$

- Pienentävät efektiivistä avaimen pituutta.

  Esim. DESin komplementaatio-ominaisuus:

  $$E_K(P) = C \ \Rightarrow\ E_{K'}(P') = C'.$$

  Tuplaa tunnetun selväkielen määrän ja siten helpottaa joitain hyökkäyksiä kertoimella 2. Ei-toivottu, mutta siedettävä ominaisuus, sillä avainpituudesta menetetään 1 bitti.

Tasavahva avainavaruus

- Kaikkien avainten tulisi olla yhtä hyviä.
- Monissa oikeissa salakirjoitusmenetelmissä on kuitenkin heikkoja avaimia, joita käytettäessä salaus ei toimi tarkoitetulla tavalla. Pahimmassa tapauksessa $E_K(P) = P$.

  Pieni määrä heikkoja avaimia ei haittaa, jos avain valitaan satunnaisesti.
- Salakirjoitujärjestelmän suunnittelija saattaa tahallaan järjestää esim. 99.9% avaimista heikoiksi.

  Siksi suunnitteluperiaatteet tulisi julkistaa.

Data Encryption Standard (DES)

- laajalti käytetty, erityisesti pankkimaailmassa
- **ei nykyisin turvallinen**
- lähtökohta kryptoanalyysin tutkimukselle ja algoritmien kehittämiselle
- lohkosalausalgoritmi
- 64 bitin lohkot
- 56/64 bitin avain **liian lyhyt**
- suunniteltu rautatoteutusta varten
- 16 iteraation Feistel

$$\boxed{\begin{array}{l} x_{-1}, x_0 = P \qquad \text{(32+32=64 bittiä)} \\ x_i = x_{i-2} \oplus f_i(x_{i-1}),\ i = 1, \ldots, 16 \\ C = x_{15}, x_{16} \end{array}}$$

- Sekoitusfunktio $f_i = f(x_{i-1}, K_i)$ riippuu eri kierroksilla eri aliavaimista.
- Salaaminen ja avaaminen tapahtuvat samalla algoritmilla; vain aliavainten järjestys vaihtuu.
- Alku- ja loppupermutaatio eivät vaikuta algoritmin turvallisuuteen.

Sekoitusfunktio f_i

- ekspansiopermutaatio
  - laajentaa 32-bittisen datalohkon 48-bittiseksi
  - diffuusio: viereiset 4/6 bitin lohkot riippuvaisiksi toisistaan
  - tulokselle XOR 48-bittisen aliavaimen kanssa
- S-boksit
  - konfuusio: epälineaarinen substituutio
  - 8 kpl, kuhunkin sisään 6 bittiä ja ulos 4 bittiä
  - pienentää 48-bittisen datalohkon 32-bittiseksi
  - algoritmin turvallisuuden kannalta keskeinen osa
  - ohjelmassa toteutus taulukkona
  - niin suuria kuin 70-luvun alussa mahdollista toteuttaa yhdellä piirillä
- P-boksi
  - permutaatio
  - diffuusio: sekoittaa databitit ennen seuraavaa kierrosta

Aliavaimet (*key schedule*)

- Kaikilla 16 kierroksella käytettään eri aliavainta.
- Aliavaimeen valitaan 48 bittiä avaimen 56:sta.
- Turvallisuuden kannalta on tärkeää, että aliavaimet valitaan hiukan epäsäännöllisesti.

Implementaatio

- suunniteltu toteutettavaksi raudalla
- permutaatiot vaikeita toteuttaa ohjelmallisesti
- nopeimmat kaupalliset DES-piirit 1 Gbittiä/s (Digital 1992)
- 166 MHz Pentium 3.5 Mtavua/s

DES-algoritmin turvallisuus

- 56-bittinen avain liian lyhyt
  - brute force-hyökkäys mahdollinen
  - Rinnakkaisprosessorijärjestelmiä avainten läpikäymiseen on ainakin suurvaltojen tiedusteluelimillä.
  - avainta lyhennetty tahallaan
- komplementaatio-ominaisuus
  - komplementtiavaimella salakirjoituskin komplementti

$$E_K(P) = C$$
$$E_{K'}(P') = C'$$

  - todellinen avaimenpituus 55 bittiä joidenkin hyökkäysten kannalta.
- pieni joukko heikkoja avaimia
  - 4 avainta, joilla kaikki aliavaimet identtisiä (*weak keys*)
  - 48 avainta, joilla vain 4 erilaista aliavainta (*possibly weak keys*)
  - 6 avainparia, jotka tuottavat saman salausfunktion (*semiweak keys*)

Data Encryption Standard (DES)

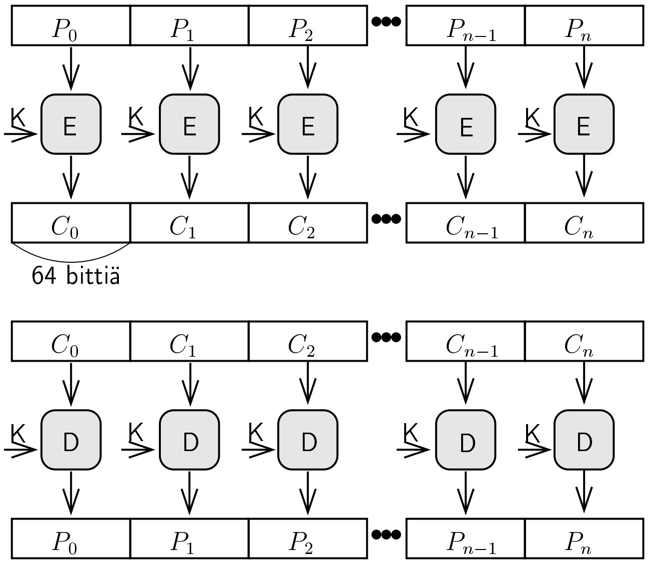
- DES ei ole ryhmä.
- S-boksien suunnittelu
  - algoritmin suunnittelun keskeinen tehtävä
  - turhan pieniä
  - optimoitu differentiaalista kryptoanalyysia vastaan
  - huomattavasti parempia kuin satunnaiset
  - kätkettyjä takaportteja?
- Kierrosten määrä on valittu niin, että parhaat tunnetut ratkaisumenetelmät yhtä hyviä kuin brute force.

Lohkosalakirjoituksen käyttömoodit

- Sanomat pitempiä kuin 64 bittiä

Electronic Code Book (ECB)

- Sanoma jaetaan 64 bitin (8 tavun) lohkoihin ja salataan lohko kerrallaan.

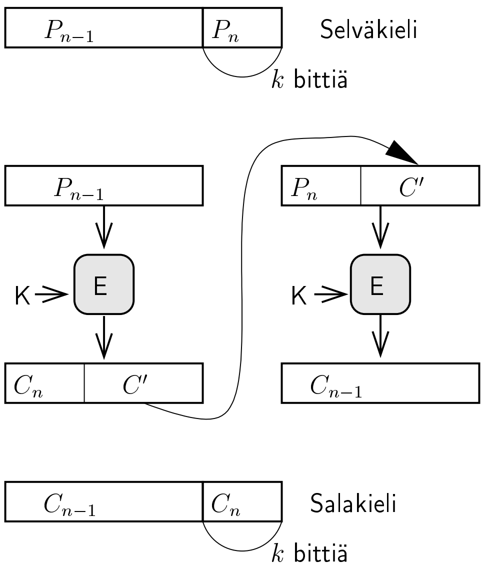


Jos sanoman pituus ei ole jaollinen 8 tavulla:

- **Täyttäminen**
  Lisätään sanoman loppuun 1 ja täytetään viimeinen lohko nollilla. Bittitasolla

  $10001001\ 11011101\ 01101011\ 01011010$
  $01110010\ 10110011\ 00000000\ 00000000$

  tai tavutasolla $112, 62, 1, 0, 0, 0, 0, 0.$
- **Ciphertext stealing**



Salakielinen teksti on saman pituinen kuin selväkielinen.

ECB-moodi

- Sopii hajasaantitiedostoille.
- **Samanlaiset lohkot havaitaan salakielessä.** Vakava ongelma, koska tietokonedata usein määrämuotoista.
- **Salaus ei suojaa eheyttä!**

  Lohkojen järjestystä voidaan muuttaa huomaamatta tai lohkoja voidaan siirtää sanomasta toiseen. Esim.

Nimi			Tehtävänimike		Palkka
C_0	C_1	C_2	C_3	C_4	C_5

Matti Manageri	Pääjohtaja	80 000.00
Risto Rehti	Ikkunoiva tukihenkilö	8 000.00

Matti Manageri	Pääjohtaja	80 000.00
Risto Rehti	Ikkunoiva tukihenkilö	80 000.00

  Aina pitää huolehtia, että yhteen kuuluvat osat pysyvät yhdessä (esim. pankkikortin tunnusluku liittyy tiettyyn korttiin).
- ECB-moodia käytetään lyhyille sanomille ($\leq$ 64 bittiä).

Cipher Block Chaining (CBC)

- Samanlaisten lohkojen tunnistaminen estetään liittämällä peräkkäiset lohkot liitetään toisiinsa ketjuksi.
  - Selväkielinen lohko XORataan edellisen salakielisen lohkon kanssa ennen salausta.

$$C_i = E_k(P_i \oplus C_{i-1})$$
$$P_i = C_{i-1} \oplus D_k(C_i)$$

  - Ensimmäinen lohko XORataan mielivaltaisen **alustusvektorin IV** (*initialization vector*) kanssa.

$$C_0 = E_k(P_0 \oplus IV)$$
$$P_0 = IV \oplus D_k(C_0)$$

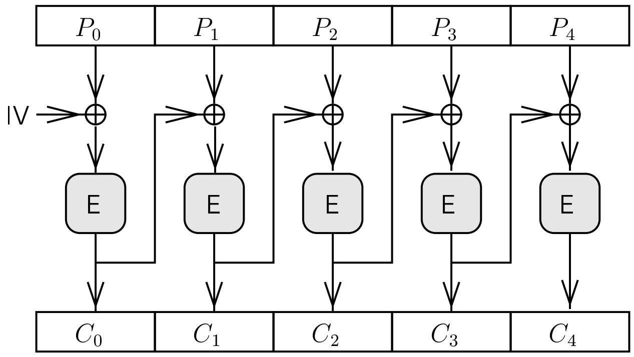
  - Alustusvektori lähetetään ennen salakielisiä lohkoja.

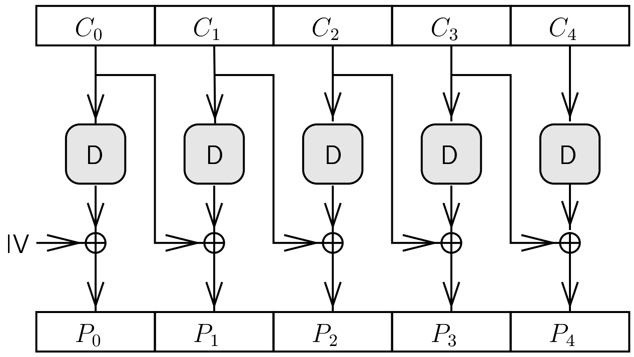
$$IV, C_0, C_1, C_2, C_3, \ldots$$

- Alustuvektoriksi käy satunnaisluku, sanoman numero tai aikaleima.

  Alustusvektorin pitää olla joka sanomalla eri. Muuten samanlaiset sanoman alut paljastuvat.
- Virhe yhdessä salakielisessä lohkossa sekoittaa kaksi selväkielistä lohkoa.

CBC-moodi





- CBC-moodi sopii käytettäväksi lähes aina.

  Suorasaantitiedostossakin CBC on parempi kuin ECB. Levyltä luettavien lohkojen määrä tosin kaksinkertaistuu.

- **Salaus (CBC-modissakaan) ei suojaa eheyttä!**
  - Jos sanoman loppuosa vaihdetaan, muuttuu vain yksi lohko roskaksi.

Nimi		Tehtävänimike			Palkka
C_0	C_1	C_2	C_3	C_4	C_5

Matti Manageri	Pääjohtaja	80 000.00
Risto Rehti	Ikkunoiva tukihenkilö	8 000.00

Matti Manageri	Pääjohtaja	80 000.00
Risto Rehti	Ikkunoiva tukihe7?Lo$a?'	80 000.00

  - Muuttamalla salakielisen lohkon C_i bittejä voidaan kääntää halutut bitit selväkielisessä lohkossa C_{i+1}. Vain lohko C_i muuttuu roskaksi.
  - Muuttamalla IV:tä voidaan kääntää ensimmäisen lohkon bittejä.
- Historiallisissa protokollissa CBC-salausta käytetään myös eheyden suojana. Siitä aiheutuu lähes aina ongelmia.

  Eheys on syytä varmistaa aina erikseen siihen tarkoitetuilla menetelmillä. IV kannattaa usein sisällyttää eheystarkistukseen.

k-bit Cipher-Feedback Mode (CFB)

- Keino pienentää salattavan lohkon pituutta. Käyttö esim. iteraktiivisten yhteyksien salauksessa.
- Esim. 8 bitin CFB:
  1. Salataan 64 bitin siirtorekisterin sisältö.
  2. XORataan tuloksen 8 eniten merkitsevää bittiä selväkielen tavun kanssa. Saadaan yksi salakielen tavu.
  3. Siirretään siirtorekisteriä 8 bittiä vasempaan ja laitetaan salakielinen tavu vähiten merkitsevien bittien paikalle.

$$C_i = P_i \oplus E_K(C_{i-1}, C_{i-2}, C_{i-3}, C_{i-4}, C_{i-5}, C_{i-6}, C_{i-7}, C_{i-8}),$$

  missä kukin C_j on 8 bittiä pitkä.
- Siirtorekisteri alustetaan **joka kerta eri luvulla** (*initialization vector* IV) , jonka ei tarvitse olla salainen.

$$C_{-8}, C_{-7}, C_{-6}, C_{-5}, C_{-4}, C_{-3}, C_{-2}, C_{-1} = IV.$$

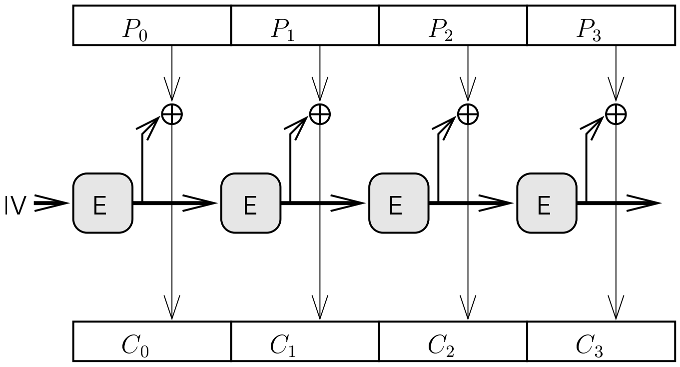
  Huom! CBC-moodissa ei mielellään käytetä samaa IV:tä kahta kertaa samalla avaimella. CFB-moodissa IV:n on pakko olla uniikki tai sanoman sisältö paljastuu.

Output-Feedback Mode (OFB)

- Synkroninen bittivirtasalaus (*synhronous stream cipher*)

  Tuotetaan 64-bittisiä pseudosatunnaislukuja salausalgoritmilla ja XORataan ne selväkielisen sanoman kanssa

$$S_i = E_K(S_{i-1})$$
$$C_i = P_i \oplus S_i$$
$$P_i = C_i \oplus S_i$$



- Pseudosatunnaislukugeneraattori alustetaan joka sanomalle eri arvolla, jonka ei tarvitse olla salainen.

$$S_{-1} = IV.$$

  Siis samoja pseudosatunaislukuja ei saa käyttää kahta kertaa! (vert. *two-time-pad*)

OFB-moodi

- Bitin kääntyminen salakielisessä sanomassa kääntää vain vastaavan bitin selväkielisessä sanomassa. Virhe ei leviä.
- Ongelma: Jos bittejä katoaa tai tulee lisää, lähettäjän ja vastaanottajan generaattorit eivät enää synkronoidu. (CBC- ja CFB-moodit synkronoituvat itsestään.)
- Suojaako salaus OFB-moodissa sanoman eheyttä yhtään?

Laskurimoodi (*Counter Mode*)

- Salataan lohkoalgoritmilla numerot $0, 1, 2, 3, \ldots$ ja XORataan tulos selväkielen kanssa.

$$C_i = P_i \oplus E_k(i)$$
$$P_i = C_i \oplus E_k(i)$$

- Sopii hajasaantitiedostoille, sillä salausta tai avausta ei tarvitse tehdä järjestyksessä.

  i voi olla esim. levylohkon numero.