Luku 1

Johdanto

Koodaus merkitsee yleisesti tiedon esittämistä toisessa muodossa. Koodit jaetaan käyttötarkoituksen mukaan kolmeen ryhmään:

Lähdekoodaus

- pelkkä tiedon esitys tarpeen mukaisessa muodossa
- Morse-aakkoset (jotain seuraavanlaista):

$$A = \cdot -$$
$$B = - \cdot \cdot \cdot$$
$$SOS = \cdot \cdot \cdot - - - \cdot \cdot \cdot$$

- tietokoneissa käytettävä ASCII-koodi:

$$A = 65 = 01000001$$
$$B = 66 = 01000010$$
$$C = 67 = 01000011$$

Tässä on itse asiassa käytetty kolmea koodausta:
- tietokoneen virtapiirien fyysiset tilat on koodattu biteiksi 0 ja 1
- kirjaimet A,B,C, ... on koodattu 10-järjestelmän kokonaisluvuiksi 65, 66, 67, ...
- desimaalijärjestelmän kokonaisluvut 0-255 on koodattu binääriluvuiksi eli 2-kantaiseen lukujärjestelmään eli kahdeksan bitin tavuiksi

tiedon pakkaus

- pikakirjoitus
- pakkausohjelmat (WinZip, PkArc, LhArc, Zip, DblSpace, ...)

Kanavakoodaus

- tiedon käsittelyssä (tiedonsiirto) tapahtuvien virheiden havaitseminen ja korjaaminen

- tietokoneiden pariteetintarkistus
- matkapuhelimet
- satelliitiyhteydet (Voyager ym.)
- CD-soittimet

Kryptografia, salakirjoitus

- tiedon salassapito

- armeijan viestiyhteydet
- tietokoneiden käyttäjätunnukset
- pankkiyhteydet

Yleistä

Kurssilla havainnollistetaan ongelmakenttää, esitetään esimerkkejä tiedon salaamisesta, katsotaan lähdekoodauksen perusteita, luodaan lyhyt katsaus tiedonsiirtokanavien matemaattisiin malleihin ja tutustutaan virheitä korjaavien koodien algebralliseen teoriaan.

Ongelmakenttä

Kuvitellaan, että tämä teksti pitäisi saada siirrettyä johonkin kauempana olevaan tietokoneeseen. Käy se niinkin, että tämä paperi lähetetään pintapostissa ja perillä vastaanottava henkilökunta kirjoittaa tekstin kyseiseen koneeseen. Se vain kestää viikkoja, kun taas tietokoneverkkoja käyttäen koko asia vie ainoastaan muutaman minuutin. Elektroniikka ei kuitenkaan tunne kirjaimia, joten elektronisessa tiedon käsittelyssä ensimmäinen ongelma on esittää teksti sähköisessä muodossa. Miten se teknisesti suoritetaan, on ongelma, joka ei kuulu tähän kurssiin, mutta ongelman matemaattinen puoli kuuluu aihepiiriin. Koska virran kulku (tai jännitteen vaihtelu, magneettisuus yms.) on esitettävissä bitteinä 0 ja 1 (0 = virta ei kulje, matala jännite, magnetisoimaton; 1 = virta kulkee, korkea jännite, magnetisoitu), ongelma on matemaattisesti tekstin esittäminen binäärimuodossa. Tämä on lähdekoodausta.

Kun (binäärimuotoista) tietoa siirretään paikasta toiseen, tarvitaan väline, jonka avulla siirto tapahtuu. Välineenä voi olla esimerkiksi puhelinlinja, radioaallot tai mikron levyke. Kaikista näistä käytetään yhteisnimitystä tiedonsiirtokanava. Heikkotasoinen materiaali kanavalaitteistossa sekä voimakkaat kanavan ulkopuoliset sähkö- ja magneettikentät aiheuttavat kanavassa siirrettävään tietoon mahdollisesti virheitä

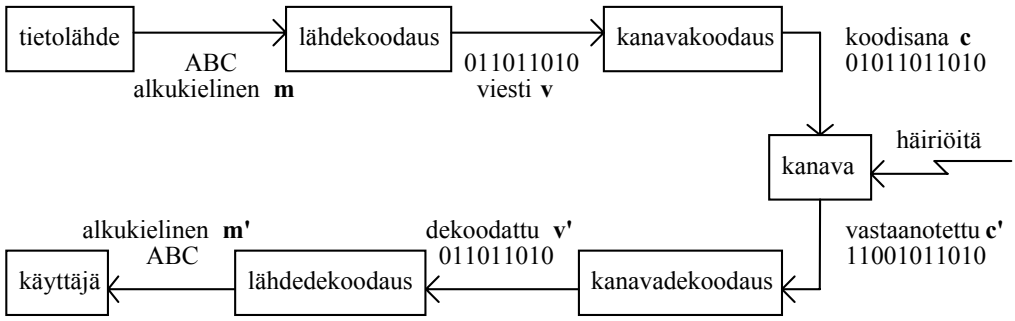
(esim. levykkeen vialliset sektorit, televisiokuvan heikko laatu), mikä merkitsee sitä, että kanavasta vastaanotettu tieto ei olekaan sama kuin sinne lähetetty. Esimerkiksi, jos kanavaan lähetetään merkki A = 65 = 01000001 ja jos bitti 4 muuttuu kanavassa siirron aikana 0:sta 1:ksi, vastaanottaja saakin merkin 01010001 = 81 = Q. Ihminen havaitsee normaalista tekstistä tällaisen pikku virheen (SANA → SQNA) helposti, mutta kun vastaanottajana onkin lukutaidoton ja ymmärtämätön kone, sillä ei ole mitään keinoa tietää, että vastaanotettu merkki Q on virheellinen. Ongelmaan löytyy ainakin periaatteessa ratkaisu: lisätään lähetettävään tietoon A = 01000001 tarkistusbittejä, jotka mahdollistavat kanavassa tapahtuvien virheiden automaattisen havaitsemisen ja korjaamisen. Esimerkiksi lisätään lähetettävään tietoon ennen siirtoa yksi bitti 0 tai 1 niin, että 1:sten määrä on parillinen: 01000001 → 001000001. Jos tiedon siirron aikana yksi bitti muuttuu, 001000001 → 001010001, vastaanottaja saa bittijonon 001010001, jossa 1:sten määrä onkin pariton, ja tietää, että ainakin yksi bitti on virheellinen. Pariteetintarkistuksen lisääminen ei vielä mahdollista virheen paikantamista eikä korjaamista (101101010, mikä väärin ?), mutta se on yksinkertaisin esimerkki virheitä korjaavien koodien (kanavakoodaus) perusajatuksesta: lisätään lähetettävään tietoon tarkistustietoa.

Esimerkki 1 *Suomalaisten henkilötunnus koostuu kolmesta osasta: syntymäajasta, henkilökohtaisesta yksilönumerosta, joka miehillä on pariton ja naisilla parillinen sekä tarkistusmerkistä, joka määräytyy alkuosan jakojäännöksen mod 31 perusteella. Lisäksi välimerkki on +, - tai A sen mukaan, onko syntynyt 1800-, 1900 tai 2000-luvulla. Esimerkiksi 131052-308T on 13.10.1952 syntyneen naisen tunnus (308 on parillinen). T on tarkistusmerkki, joka on saatu jakamalla luku 131052308 luvulla 31, jolloin jakojäännös on 25. Kun jakojäännös on 0 - 9 tarkistusmerkki on suoraan tämä jakojäännös. Suurempien jakojäännösten osalta jatketaan kirjaimilla 10 → A, 11 → B jne. Tietyt helposti sekaantuvat kirjaimet (kuten G, O,...) jätetään pois.*

Siis alkukielinen teksti m ($m = ABC$) koodataan lähdekoodauksella kanavan vaatimaan esitysmuotoon viestiksi v (v = 011011010), joka taas koodataan koodisanaksi c (lisäämällä tarkistusbittejä, c = 01011011010). Koodisana c lähetetään kanavaan. Jos kanavasta vastaanotettu sana r on eri kuin lähetetty sana c, on tapahtunut tiedonsiirtovirheitä. Vastaanottavassa päässä on dekooderi, joka tulkitsee vastaanotetun sanan r viestiksi v'. Mikäli v' on sama kuin koodisanana c lähetetty viesti v, on virhe saatu korjattua, muussa tapauksessa tiedonsiirtovirheet ovat aiheuttaneet myös dekoodausvirheen. Tarpeen vaatiessa dekoodattu viesti v' muunnetaan myös (binäärimuodosta) alkukieliseen (teksti)muotoon.

Yksi informaatioteorian perustuloksista sanoo, että koodien avulla tiedonsiirto saadaan miten luotettavaksi tahansa, kunhan tiedonsiirtokanavaa ei ylikuormiteta.

Seuraavassa vielä kaaviokuva tilanteesta.



Tiedonsiirtoon liittyy aina myös turvallisuusriski: esimerkiksi vieraan valtion vakoojat iskevät kanavaan kiinni laitteensa, rosvot hakkeroivat pankkien tieokoneita jne. Tämä tilanne aiheuttaa vaatimuksia myös tiedon salausmahdollisuuteen, pelkkä binääriesitys tai tunnettujen virheitä korjaavien koodien käyttö ei vielä estä ketään urkkimasta selville salaiseksi tarkoitettua tietoa. Tiedon salaus on huomattavasti muuta koodausteoriaa vanhempaa perua ja sillä on oma matemaattinen teoriansa.

Peruskäsitteitä

Aakkosto on äärellinen, epätyhjä joukko (esim. $\{A, B, C, ..., Z\}$ tai $\{0, 1\}$), sen alkiot ovat kirjaimia. Kirjaimista muodostettu äärellinen jono (esim. ACB tai 01001) on *sana*. Sanan pituus on siinä olevien kirjainten määrä (esim. sanan $a = ACBAC$ pituus on 5 ja sanan $b = 0100$ pituus on 4). Aakkoston $\mathcal{A}$ kaikkien sanojen joukosta käytetään merkintää $\mathcal{A}^*$ (esim. kun $\mathcal{A} = \{a, b, c, ..., z\}$, niin $\mathcal{A}^* = \{a, b, ab, ac, cb, ai, ...\}$).

Koodaus f aakkostosta $\mathcal{L}$ aakkostoon $\mathcal{K}$ on injektio $f : \mathcal{L}^* \to \mathcal{K}^*$. Aakkosto $\mathcal{L}$ on *lähde-* tai *viestiaakkosto* ja sen sanat $v \in \mathcal{L}^*$ ovat *sanomia* tai *viestejä*. Aakkosto $\mathcal{K}$ on *koodiaakkosto*. Viestien $v \in \mathcal{L}^*$ kuvat $c = f(v) \in \mathcal{K}^*$ ovat *koodisanoja* ja kaikkien koodisanojen joukko $\mathcal{C} = f(\mathcal{L}^*) = \{f(v) : v \in \mathcal{L}^*\} \subseteq \mathcal{K}^*$ on koodi. Koodauksen injektiivisyys ($f(v) = f(v') \Leftrightarrow v = v'$) on luonnollinen vaatimus: eri viestit koodataan eri koodisanoiksi. Tällöin koodaus voidaan myös purkaa: jos $c = f(v)$, niin $v = f^{-1}(v)$. Kuvausta $f^{-1} : \mathcal{C} \to \mathcal{L}^*$ nimitetään koodausta f vastaavaksi dekoodaukseksi.

Yllä oleva määrittely on hyvin yleinen tuoden esille sen, että tavoitteena ei ole pelkästään kirjaimien koodaus vaan merkkijonojen, jotka voivat olla miten pitkiä tahansa. Kurssissa rajoitutaan kuitenkin *morfismeihin* eli koodauksiin f, joissa jokainen sana $v = v_1v_2v_3...v_n$ koodataan kirjaimittain:

$$f(v) = f(v_1v_2v_3...v_n) = f(v_1)f(v_2)f(v_3)...f(v_n) = c_1c_2c_3...c_n,$$

missä $c_1 = f(v_1)$, $c_2 = f(v_2)$ jne.ovat kirjaimia v_1, v_2, v_3,...,v_n vastaavat koodisanat. Tällöin nimitetään yksinkertaisemmin rajoitettua

kuvausta $f : \mathcal{L}^* \to \mathcal{K}^*$ koodaukseksi ja joukkoa $\mathcal{C} = f(\mathcal{L}^*) \subseteq \mathcal{K}^*$ koodiksi. Mikäli kaikkien kirjainten koodisanat ovat yhtä pitkiä, koodi on *kiinteän pituinen*, muulloin se on *vaihtelevan pituinen.*

Esimerkki 2 *Kun lähdeaakkosto on* $\mathcal{L} = \{X, Y, Z\}$*, koodiaakkosto on* $\mathcal{K} = \{0, 1\}$ *ja kirjaimia* X, Y, Z *vastaavat koodisanat ovat*

$$x = f(X) = 0, y = f(Y) = 10, z = f(Z) = 11,$$

niin viesti $v = XYZX$ *koodataan bittijonoksi*

$$c = f(XYZX) = f(X)f(Y)f(Z)f(X) = xyzx = 010110$$

Koodi $\mathcal{C} = \{x, y, z\} = \{0, 10, 11\}$ *on vaihtelevan pituinen. Sanan* $r = 1110010$ *dekoodaus suoritetaan seuraavasti: koska* r*:n ensimmäinen bitti on 1 ja 1:llä alkavat koodisanat ovat* $y = 10$ *ja* $z = 11$*, niin* r*:n alkuosa on joko* y *tai* z*, ja koska* r*:n toinenkin bitti on 1, niin* r*:n alkuosa on* $11 = z$ *ja näin ollen* $r = z10010$*. Vastaavasti kaksi seuraavaa bittiä ovat* $10 = y$ *ja siis* $r = xy010$*. Seuraava on* $0 = x$*, siis* $r = zyx10$*, ja lopussa on vielä* $10 = y$*, joten* $r = zyxy$*. Sana* r *on nyt jaettu yksikäsitteisesti koodisanoiksi:* $r = zyxy$*, mitä vastaava viesti on* $ZYXY$*.*

Esimerkki 3 *Jos kirjainten koodaukseksi yritettäisiin valita*

$$X \to 0 = x, Y \to 1 = y, Z \to 01 = z,$$

sanojen XYX *ja* ZX *koodaus antaisikin saman bittijonon:*

$$XYX \to xyx = 010, ZX \to zx = 010$$

Huomaa myös, että esimerkiksi vastaanotettua sanaa $r = 00101$ *ei voisi enää jakaa yksikäsitteisesti koodisanojen* x, y, z *katenaatioksi, vaan vaihtoehtoja olisi useampia:*

$$r = xxyxy,\ r = xxyz,\ r = xzxy,\ r = xzz$$

Näin valiten ei siis saada määritelmän mukaista koodausta, jolla voitaisiin koodata kaikki sanat.

Edeltävä esimerkki osoittaa, miksi koodauksen $f : \mathcal{L}^* \to \mathcal{K}^*$ on oltava injektio. Esimerkin yksikäsitteisyysongelma on ominaista vain vaihtelevan pituisille koodeille. Kiinteän pituisille koodeille ei vastaanotetun sanan jakaminen yksikäsitteisesti koodisanoiksi tuota ongelmia: jos jokaista kirjainta vastaavan koodisanan pituus on sama n, riittää jakaa vastaanotettu sana n:n pituisiin osiin ja tulkita.

Itse asiassa esimerkin 3 ongelma johtuu siitä, että $x = 0$ on sanan $z = 01$ etuliite eli $z = x1$. Mikäli lähdeaakkoston $\mathcal{L}$ kirjaimia vastaavat koodisanat valitaan siten, että mikään niistä ei ole minkään toisen koodisanan etuliite, niin saadaan määritelmässä vaadittu injektio $\mathcal{L}^* \to \mathcal{K}^*$. Tällainen valinta on tehty esimerkissä 2, jossa yksi koodisanoista on yhden bitin pituinen ja kaksi muuta koodisanaa ovat kahden bitin pituisia. Aivan varmasti myöskään samanpituisista sanoista yksikään ei ole minkään toisen etuliite.

Harjoitustehtäviä

1. Esitä esimerkki siitä, miksi

$$A \to 1 \quad E \to 00 \quad I \to 010 \quad S \to 011 \quad T \to 01$$

ei ole koodi.

Luku 2

Perustuloksia

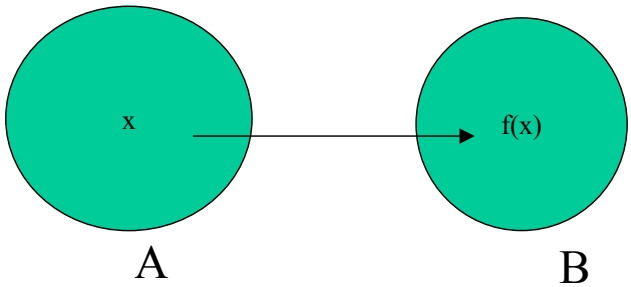
Olkoon A joukko. Merkitään $x \in A$ tarkoittamaan sitä, että x on joukon A alkio eli x kuuluu joukkoon A. Olkoon B toinen joukko. Merkitään $A \subseteq B$, kun halutaan sanoa, että A on B:n osajoukko eli alijoukko. Mikäli $A = B$ ei ole mahdollinen, voidaan myös merkitä $A \subset B$. Jos halutaan ilmaista, että x ei kuulu joukkoon A tai A ei ole B:n osajoukko voidaan kirjoittaa vastaavasti $x \notin A$ ja $A \not\subseteq B$. Olkoon f kuvaus joukosta A joukkoon B. Merkitään $f : A \to B$. Sen mukaan, miten f kuvaa A alkioita B:hen, annetaan f::lle erilaisia nimityksiä: f on *injektio*, jos f ei kuvaa kahta pistettä samaan pisteeseen eli

$$\text{jos} \quad f(x_1) = f(x_2), \quad \text{niin} \quad x_1 = x_2.$$

f on *surjektio*, jos koko B esiintyy kuvana eli

aina kun $y \in B$ niin on olemassa sellainen $x \in A$ että $f(x) = y$

Jos f on sekä injektio että surjektio, niin sanotaan, että se on *bijektio*. Joukkojen A ja B välinen bijektio antaa täydellisen alkioittaisen vastaavuuden näiden joukkojen välillä.



Lukuteoriaa

Sanotaan, että kokonaisluku a on *jaollinen* kokonaisluvulla b, jos on olemassa sellainen kokonaisluku c, että $a = bc$. Esimerkiksi. 3 jakaa luvun 15, koska $15 = 3 \times 5$.

Kokonaislukujen $a_1, a_2, \ldots, a_k$ *suurimmalla yhteisellä tekijällä* (syt) tarkoitetaan sellaista lukua d, joka jakaa kaikki ko. luvut ja on suurin mahdollinen tämän ehdon toteuttavista. Yleensä merkitään

$$d = syt(a_1, a_2, \ldots, a_k).$$

Esimerkki 4 $syt(45, 9, 21) = 3$.

Kahden luvun a ja b syt voidaan laskea seuraavalla Eukleideen algoritmilla: Olkoon $a > b$. Jaa a b:llä

$$a = q_1 b + r_1.$$

Jatka jakamalla b r_1:llä.

$$b = q_2 r_1 + r_2.$$

Jatka edelleen valitsemalla edellinen jakaja uudeksi jaettavaksi ja jakojäännös uudeksi jakajaksi.

$$\begin{gathered} r_1 = q_3 r_2 + r_3 \\ \ldots \\ r_n = q_{n+2} r_{n+1} + r_{n+2} \\ \ldots \end{gathered}$$

Lopulta joudutaan tilanteeseen, jossa jako menee tasan, koska r_i-luvut kaiken aikaa pienenevät. Tällöin

$$r_k = q_{k+2} r_{k+1}.$$

Nyt $r_{k+1} =$ syt(a, b).

Esimerkki 5 *Lasketaan* syt$(2279, 989)$

$$2279 = 2 \times 989 + 301$$

$$989 = 3 \times 301 + 86$$

$$301 = 3 \times 86 + 43$$

$$86 = 2 \times 43$$

Täten syt$(2279, 989) = 43$.

Alkuluvuksi nimitetään sellaista positiivista kokonaislukua, joka on jaoton ja $\neq 1$. Tällaisia ovat esim. 2, 3, 5, 7, 11, 13, 17,... Alkulukuja on äärettömän paljon. Tällä hetkellä (lokakuu 2003) suurin tunnettu alkuluku on

$$2232007 \cdot 2^{1490605} - 1$$

Luvussa on 448724 numeroa. Suuria alkulukuja etsitään yleensä tietyntyyppisistä luvuista, jotka ovat alkulukuja suurella todennäköisyydellä. Tällaisia ovat esim. $2^n - 1$ tyyppiset luvut. "Arvokkaampia" kryptografian kannalta kuitenkin ovat pienemmät "ei-mitään tunnettua tyyppiä" olevat alkuluvut. Alkulukutaulukoita muodostetaan ns. *Eratostheneen seulalla.* Se perustuu siihen ajatukseen, että mikäli annettu luku n on jaollinen $n = n_1 n_2$, niin ainakin toinen luvuista n_1 tai n_2 on pienempi kuin $\sqrt{n}$. Jos siis etsitään kaikki n:ää pienemmät alkuluvut, niin riittää poistaa kaikki $\sqrt{n}$:ää pienempien lukujen monikerrat.

Esimerkki 6

~~1~~	2	3	~~4~~	5	~~6~~	7	~~8~~	~~9~~	~~10~~
11	~~12~~	13	~~14~~	~~15~~	~~16~~	17	~~18~~	19	~~20~~
~~21~~	~~22~~	23	~~24~~	~~25~~	~~26~~	~~27~~	~~28~~	29	~~30~~
31	~~32~~	~~33~~	~~34~~	~~35~~	~~36~~	37	~~38~~	~~39~~	~~40~~
41	~~42~~	43	~~44~~	~~45~~	~~46~~	47	~~48~~	~~49~~	~~50~~
~~51~~	~~52~~	53	~~54~~	~~55~~	~~56~~	~~57~~	~~58~~	59	~~60~~
61	~~62~~	~~63~~	~~64~~	~~65~~	~~66~~	67	~~68~~	~~69~~	~~70~~
71	~~72~~	73	~~74~~	~~75~~	~~76~~	~~77~~	~~78~~	79	~~80~~
~~81~~	~~82~~	83	~~84~~	~~85~~	~~86~~	~~87~~	~~88~~	89	~~90~~
~~91~~	~~92~~	~~93~~	~~94~~	~~95~~	~~96~~	97	~~98~~	~~99~~	~~100~~

Taulukkoon jäljelle jääneet ovat alkulukuja. Suurten alkulukujen generointiin tämä metodi ei kuitenkaan sovellu, koska se vaatiin paljon muistia.

Jokainen luku voidaan jakaa yksikäsitteisesti alkutekijöiden potenssien tuloksi. Siis

$$n = p_1^{e_1} p_2^{e_2} \cdots p_k^{e_k}.$$

Jos luvuista a ja b on käytettävissä edellisen kaltaiset alkutekijäesitykset, niin syt:n haku on erittäin yksinkertaista. Poimitaan vain molemmista esityksistä yhteiset alkuluvut ja vastaavasti pienimmät eksponentit.

Esimerkki 7

$$a = 3^4 5^2 7^0 11^1 \text{ ja } b = 3^2 5^1 7^3 11^0$$

Tällöin

$$\text{syt}(a, b) = 3^2 5^1 7^0 11^0 = 45.$$

Mikäli hajoitelmaa ei ole valmiina, ei sitä yleensä kannata tehdä, vaan Eukleideen algoritmi on merkittävästi nopeampi. Luvun hajoittaminen alkutekijöihin on erittäin työläs tehtävä, kun kyseessä on vähänkin isompi luku. Probleeman ratkaiseminen perustuu nimittäin pitkälti jaollisuuskokeiluihin luvun neliöjuurta pienemmillä luvuilla. "Ison" luvun (numeroita esim. 200) tekijöihin jako on nykyisin käytännössä mahdotonta.

Eulerin $\varphi(n)$-funktio ilmoittaa paljonko on olemassa sellaisia n:ää pienempiä lukuja, joiden syt n:n kanssa on 1. Esimerkiksi $\varphi(10) = 4$, sillä edellä mainitun kaltaisia lukuja ovat 1, 3, 7, 9 (4 kpl). $\varphi(n)$:n arvo saadaan seuraavalla kaavalla:

$$\varphi(n) = p_1^{e_1-1}(p_1-1)p_2^{e_2-1}(p_2-1)\cdots p_k^{e_k-1}(p_k-1).$$

Esimerkki 8

$$\begin{aligned}\varphi(480200) &= \varphi(2^3 5^2 7^4) \\ &= 2^2 \times (2-1) \times 5^1 \times (5-1) \times 7^3 \times (7-1) = 164640\end{aligned}$$

Kongruenssi

Kun tutkitaan lukujen välisiä jaollisuusominaisuuksia, saa jakolaskun jakojäännös keskeisen merkityksen. Voidaanhan sanoa, että a on b:llä jaollinen, jos $a = k \times b + c$ ja $c = 0$. Määritellään, että luvut a ja b ovat *kongruentteja* $\bmod n$, jos $a - b$ on n:llä jaollinen. Merkitään

$$a \equiv b \mod n$$

Esimerkki 9

$$23 \equiv 15 \mod 8$$

$$2 \equiv 17 \mod 5$$

Käytännössä tämän käsitteen avulla pystytään laskutoimituksissa usein siirtymään pienempiin lukuihin: jos $a = k \times b + c$, niin $a \equiv c \mod b$. Jotta tätä piirrettä voitaisiin tehokkaasti käyttää hyväksi, tarvitaan kuitenkin eräitä kongruenssin ominaisuuksia. Oletetaan, että

$$a \equiv b \bmod n \text{ ja } c \equiv d \bmod n$$

Silloin

$$\begin{gathered} a + c \equiv b + d \bmod n \\ a - c \equiv b - d \bmod n \\ ac \equiv bd \bmod n \\ a^k \equiv b^k \bmod n \text{ kun } k = 1, 2, 3, \ldots \end{gathered}$$

Esimerkki 10 $4 \equiv 10 \mod 6$ *ja* $-3 \equiv 9 \mod 6$. *Täten* $4 \times (-3) \equiv 10 \times 9 \mod 6$ *eli* $-12 \equiv 90 \mod 6$.

Esimerkki 11 *Mikä viikonpäivä oli 4.12.2003. Tiedämme kalenteristä, että 6.10.2003 on maanantai, lokakuussa on 31 päivää ja marraskuussa 30. Katsotaan mikä on kysytyn päivän järjestysnumero laskettuna maanantaista 6.10.2003. Lokakuun alusta laskien saadaan:* $31+30+4 = 65$. *Järjestysnumero maanantaista 6.10.2003 on* $65-6 = 59 \equiv 3 \mod 7$ *(7 = viikonpäivien määrä). Kysytty viikon päivä on "3. päivä maanantaista eteenpäin" eli torstai.*

Esimerkki 12 *Mihin numeroon päättyy luku*

$$257^5 \times 234567 \times 786 - 546^3?$$

Lasketaan kaikki mod 10.

$$7^5 \times 7 \times 6 - 6^3 \equiv 16807 \times 42 - 216 \equiv 7 \times 2 - 6 = 8 \mod 10$$

Täten luku päättyy kahdeksikkoon.

Huomaa, että eksponenttia ei voi pienentää vähentämällä siitä modulia. Tässä asiassa tulee kuitenkin avuksi

Lause 13 *Eulerin lause.Jos*

$$\mathrm{syt}(a, m) = 1, \ \textit{niin } a^{\varphi(m)} \equiv 1 \mod m.$$

Esimerkki 14 $\varphi(9) = \varphi(3^2) = 3 \cdot (3-1) = 6$ *ja*

$$5^6 \equiv 1 \mod 9$$

$$7^{10} = 7^6 \times 7^4 \equiv 1 \times 7^4 \equiv 7^4 \mod 9$$

$$\begin{aligned} 322^{100} &\equiv 7^{100} \bmod 9 \qquad (\textit{sillä } 322 \equiv 7 \bmod 9) \\ &\equiv 7^4 \bmod 9 \qquad (\textit{sillä } 100 \equiv 4 \bmod 6) \\ &\equiv 2401 \bmod 9 \\ &\equiv 7 \bmod 9 \end{aligned}$$

Esimerkki 15 *Lasketaan* $5^{122} \mod 17$. *Ensiksi* $\varphi(17) = 16$. *Sievennetään eksponenttia* $122 \equiv 10 \mod 16$. *Täten*

$$5^{122} \equiv 5^{10} \equiv 9 \mod 17$$

Lineaarialgebraa

Aikaisemmin olemme tutustuneet vektoriavaruuksiin $\mathbb{R}^2$ ja $\mathbb{R}^3$, joiden alkiot ovat (a, b)- ja (a, b, c)-muotoisia reaalilukuvektoreita. Näillä on tutut laskuoperaatiot, yhteenlasku ja reaaliluvulla kertominen. Lisäksi näille vektoreille on määritelty mm. pistetulo, jolla voidaan tutkia kohtisuoruutta. Samoin näitä vektoreita voidaan kertoa matriiseilla, kunhan tietyt dimensioyhteensopivuudet ovat kunnossa.

Vastaavan kaltainen vektoriavaruus voidaan määritellä yleisemminkin. Kuitenkin kiinnostavaa on lähinnä sellainen tapaus, jossa ns. kerroinkunta on $\mathbb{Z}_2 = \{0, 1\}$. Tällöin esim. $\mathbb{Z}_2^5$ tarkoittaa 5-pituisia vektoreita, joissa komponentit ovat jäännösluokkasysteemin $\mathbb{Z}_2$ alkioita 0 ja 1 ja kaikki laskutoimitukset suoritetaan $\bmod 2$. Vektorit ovat muotoa $(x_1, x_2, x_3, x_4, x_5)$, missä x_i:t ovat nollia ja ykkösiä. Koodausteoriassa näitä vektoreita useimmiten merkitään yksinkertaisesti bittijonoina $x_1x_2x_3x_4x_5$

Aliavaruudeksi kutsutaan vektoriavaruuden "itsenäistä" alijoukkoa, ts. sellaista alijoukkoa, joka on *suljettu* yhteenlaskun ja kerroinkunnan alkiolla kertomisen suhteen. Siis suoritettaessa näitä operaatioita ei jouduta ulos ko. alijoukosta.

Esimerkki 16 *$\mathbb{R}^2$:n alijoukko $\{(x, x) | x \in \mathbb{R}\}$ on aliavaruus. Samoin on $\mathbb{R}^2$ itse ja $\{(0, 0)\}$. Muita $\mathbb{R}^2$:n aliavaruuksia ovat kaikki origon kautta kulkevat suorat.*

Esimerkki 17 *Joukot*

$$\{(0,0,0,0,0), (1,1,1,1,1)\},$$

$$\{(0,0,0,0,0), (1,1,0,0,0), (0,0,1,1,0), (1,1,1,1,0)\}$$

ja

$$\{(0,0,0,0,0)\}$$

ovat $\mathbb{Z}_2^5$:n aliavaruuksia.

Esimerkki 18 *$\mathbb{R}^3$:n aliavaruuksia ovat yhden alkion avaruus $\{(0, 0, 0)\}$, kaikki origon kautta kulkevat suorat, origon kautta kulkevat tasot ja $\mathbb{R}^3$ itse.*

Vektoriavaruus sisältää aina nollavektorin. Jos vektorit v ja u kuuluvat siihen, niin näiden mukana aina kuuluvat myös $-v$, $v + v$, $u + v$ ja yleensäkin kaikki v:stä ja u:sta yhteenlaskemalla ja kerroinkunnan alkioilla kertomalla muodostuvat vektorit.

Esimerkki 19 *Oletetaan, että $\mathbb{Z}_2^4$:n aliavaruus U sisältää vektorit $(1, 0, 0, 0)$ ja $(0, 0, 1, 0)$. Tällöin U:hun kuuluvat myös vektorit $(0, 0, 0, 0)$ ja $(1, 0, 1, 0)$. Mahdollisesti U sisältää muitakin vektoreita, mutta ei välttämättä, koska mainitut neljä muodostavat jo erään aliavaruuden.*

Vektoriavaruuden U alkiot $\{u_1, u_2, \ldots, u_n\}$ muodostavat *kannan*, jos kaikki U:n vektorit voidaan esittää yksikäsitteisesti muodossa

$$c_1u_1 + c_2u_2 + \cdots + c_nu_n,$$

missä c_i:t käyvät läpi kerroinkunnan alkiot.

Esimerkki 20 $\mathbb{R}^3$*:n luonnollinen kanta on*

$$\{(1,0,0),(0,1,0),(0,0,1)\}.$$

Esimerkki 21 $\mathbb{Z}_2^3$*:n luonnollinen kanta on*

$$\{(1,0,0),(0,1,0),(0,0,1)\}.$$

Esimerkki 22 $\mathbb{Z}_2^5$*:n aliavaruuden*

$$\{(0,0,0,0,0),(1,1,0,0,0),(0,0,1,1,0),(1,1,1,1,0)\}$$

kanta on esim.

$$\{(1,1,0,0,0),(0,0,1,1,0)\}.$$

Kannaksi kelpaa myös

$$\{(1,1,0,0,0),(1,1,1,1,0)\}$$

ja

$$\{(0,0,1,1,0),(1,1,1,1,0)\}.$$

Mikään kanta ei voi sisältää nollavektoria. Kaikki kantavektorit ovat ko. vektoriavaruuden vektoreita. Kanta ei yleensä ole yksikäsitteinen, vaan kuten eo. esimerkistä näkyi, se voidaan usein valita monella tavalla. Kannassa on kuitenkin aina sama määrä vektoreita.

Esimerkki 23 $\mathbb{R}^3$*:lle voidaan myös valita esim. kanta* $\{(1,1,1),(1,2,3),(1,4,9)\}$*. Käytännössä yksinkertaisinta on kuitenkin tukeutua luonnolliseen kantaan.*

Vektoreita v ja u sanotaan *ortogonaalisiksi* eli kohtisuoriksi, jos niiden pistetulo on 0. Kun etsitään annetun vektoriavaruuden U vektoreiden joukosta kaikki ne vektorit, jotka ovat kohtisuorassa kaikkia U:n aliavaruuden V vektoreita vastaan saadaan V:n *ortogonaalikomplementti* $V^\perp$

Esimerkki 24 $\mathbb{R}^2$*:n vektorit* $(1,1)$ *ja* $(1,-1)$ *ovat toisiaan vastaan kohtisuorassa. Yleisemmin kaikki* $\mathbb{R}^2$*:n aliavaruuksien*

$$U = \{(x,x)|x \in \mathbb{R}\}$$

ja

$$V = \{(x,-x)|x \in \mathbb{R}\}$$

vektorit ovat parittain kohtisuorassa. Ne muodostavat toistensa ortogonaalikomplementit eli $U = V^\perp$ *ja* $V = U^\perp$*. Geometrisesti kyseessä on kaksi toisiaan vastaan kohtisuorassa olevaa suoraa.*

Reaalisen vektoriavaruuden ja sen ortogonaalikomplementin kannat täydentävät toisiaan siinä mielessä, että ne yhdessä muodostavat koko avaruuden kannan. Äärellisen kerroinkunnan kohdalla tilanne on kuitenkin toisenlainen.

Esimerkki 25 *$\mathbb{Z}_2^5$:ssä on aliavaruus $U = \{(0,0,0,0,0),(1,0,1,1,0)\}$. Vektori (x,y,z,u,v) on kohtisuorassa vektoria $(1,0,1,1,0)$ vastaan tarkalleen silloin, kun $x+z+u \equiv 0 \mod 2$ ts. $x=z=u=0$ tai kaksi komponenteista on ykkösiä ja yksi on nolla. Alkiot y ja v saavat olla mitä tahansa. Täten*

$$\begin{aligned} U^\perp = \{&(0,0,0,0,0),(1,0,1,0,0),(0,0,1,1,0),(1,0,0,1,0),(1,1,1,0,0),\\ &(0,1,1,1,0),(1,1,0,1,0),(1,0,1,0,1),(0,0,1,1,1),(1,0,0,1,1),\\ &(1,1,1,0,1),(0,1,1,1,1),(1,1,0,1,1),\} \end{aligned}$$

Esimerkki 26 *$\mathbb{Z}_2^5$:ssä on aliavaruus $U = \{(0,0,0,0,0),(1,0,1,0,0)\}$. Tilanne on siinä mielessä erilainen, että nyt myös U:n vektorit ovat kohtisuorassa itseään vastaan. U:n ortogonaalikomplementtiin kuuluvat kaikki ne vektorit (x,y,z,u,v), joille $x+z \equiv 0 \mod 2$ ts. $x=z$.*

Harjoitustehtäviä

1. Jaa alkutekijöiden tuloksi luku $n = 19404$ ja laske sitten $\varphi(n)$.
2. Laske Eukleideen algoritmilla $\text{syt}(1023, 2071)$.
3. Laske paperilla vaiheittain

$$367 \cdot 723 + 52 \mod 13$$

$$367^{14} \mod 13$$

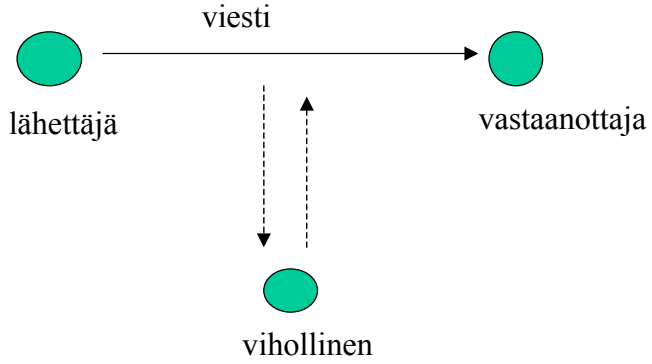
4. Laske lukujen $1,2,3,4,5,6,7,8,9,10$ potenssit $1,...,10 \mod 11$, kunnes ko. potenssi $\equiv 1 \mod 11$. Huomaa, että Eulerin lauseen nojalla, kyseeseen tuleva eksponentti on tällöin korkeintaan 10. Huomaa lisäksi, että ko. eksponentti jakaa luvun 10. Mistähän tämä mahtaa johtua?

Luku 3

Tiedon salaus

Korvaussysteemit

Salakirjoitustarve syntyy, kun luottamuksellista tekstiä tai muuta materiaalia lähetetään sellaisissa olosuhteissa, joissa se voi joutua vääriin käsiin.



Vanhimpia salakirjoituksia lienee *Caesarin korvaussysteemi*, jossa aakkostoa yksinkertaisesti siirretään esimerkiksi kolmella kirjaimella: A → D, B → E, C → F jne. eli

lähdeaakkosto: $ABCDEFGHIJKLMNOPQRSTUVXYZÅÄÖ$
↓ ↓
koodiaakkosto: $DEFGHIJKLMNOPQRSTUVXYZÅÄÖABC$

Esimerkiksi viestistä $SELVÄKIELINEN$ tulee koodisana $VHOZB$ $NLHOLQHQ$. Tämän kryptotekstin $VHOZBNLHOLQHQ$ dekoodaus (dekryptaus) takaisin selväkieliseksi on ehkä askarruttanut Caesarin

aikalaisia, mutta tämän korvaussysteemin murskaaminen on lasten leikkiä, pahimmassa tapauksessa joutuu kokeiluissa kiertämään aakkoston vain kerran, kuten seuraava esimerkki osoittaa.

Esimerkki 27 *Salakirjoituksen UPQTFDSFU selvittäminen koodauksena käytettyä korvausta tuntematta tehdään yksinkertaisesti kokeilemalla läpi kaikki mahdollisuudet:*

UPQTFDSFU	*ÖXYÄMKÅMÖ*	*GBCFTRETG*	*NIJMÄZLÄN*
VQRUGETGV	*AYZÖNLÄNA*	*HCDGUSFUH*	*OJKNÖMÖO*
XRSVHFUHX	*BZÅ AOMÖOB*	*IDEHVTGVI*	*PKLOAÄNAP*
YSTXIGVIY	*CÅÄBPNAPC*	*JEFIXUHXJ*	*QLMPBÖOBQ*
ZTUYJHXJZ	*DÄÖCQOBQD*	*KFGJYVIYK*	*RMNQCAPCR*
ÅUVZKIYK	*EÖADRPCRE*	*LGHKZXJZL*	*SNORDBQDS*
ÄVXLJZLÄ	*FABESQDSF*	*MHILYKM*	*TOPSECRET*

Käytetyn aakkoston (Suomi, Englanti, Saksa, Latina, ...) arvaaminen (eli eri aakkostojen kokeilu) teettää hieman lisätöitä.

Hieman paremman salakirjoituksen saa, kun käyttää aakkoston siirron sijasta epäsäännollistä korvausta (kunhan se on injektio). Silloin salakuuntelijat eivät pysty murskaamaan salakirjoitusta aivan yhtä yksinkertaisesti kuin Caesarin systeemiä.

Esimerkki 28 *Tässä esimerkkinä erään ohjelman englannin kielisestä README.TXT-tiedostosta satunnaisesti valittu rivi koodattuna (kryptattuna) salakirjoitukseksi satunnaisella korvauksella. Ennen koodausta pienet kirjaimet on muutettu isoiksi ja vain kirjaimet A-Z on kryptattu. SPSEJCAJXSLPZMPHPAIBZSSPMJTJSJSELXXIBKVLTJBZM*

Tämänkään salakirjoituksen selvittäminen koodausta tuntematta ei tuota suurempia ongelmia. Kirjaimittain tapahtuva korvaus nimittäin säilyttää kielen ominaisuudet. Salakirjoituksesta havaitsee pian ainakin kaksoiskirjaimet SS ja XX, joita voi käyttää hyödyksi kryptauksen murskaamisessa (taas, että yms. sanoja) . Samoin merkkiyhdistelmät SPS, PHP, $JTJSJS$ ovat sellaisia, joiden avulla saatetaan helposti arvata joitakin sanoja tai niiden osia ($JTJSJS$ selväkielisenä sisältää sekä vokaaleja että konsonantteja tai numeroita). Salakirjoituksesta löytyy edelleen kirjainparit SP, SE, ZM ja IB kahteen kertaan sekä ainakin yksi käännetty kirjainpari MP, PM ja myös kolmen kirjaimen kombinaatiot SEJ, JSE. Kaikki tällaiset havainnot helpottavat salakuuntelijan murskaustyötä, mutta ilman hyvää arvausta ne eivät kuitenkaan vielä paljasta, millaisella kirjainkorvauksella kryptoteksti on koodattu. Jollakin keinolla on löydettävä ensin muutaman kirjaimen korvaus, sen jälkeen tulkitseminen helpottuu sanottavasti. Menetelmä tähän on yksinkertainen tilastollinen analyysi. Jokaisessa kielessä on sille ominainen kirjainten esiintymisjakauma, joka myös säilyy koodattaessa tekstiä salakirjoitukseksi yksinkertaisella kirjainkorvauksella.

Esimerkin tiedostossa on kaikkiaan 26.000 merkkiä ja kirjainten $A-Z$ prosenttiosuudet ovat

E	8.881	R	4.963	C	2.788	F	1.843	W	0.818
O	5.731	A	4.632	U	2.723	Y	1.674	K	0.459
T	5.650	I	4.581	M	2.622	G	1.323	X	0.293
S	5.129	N	4.045	L	2.572	B	1.126	Z	0.039
		D	3.401	P	2.457	V	1.076	Q	0.027
				H	2.279			J	0.004

Tämän mukaan kryptotekstissä $SPSEJCAJXSLP...$ useimmin esiintyvä merkki on todennäköisesti E (englannin kielen yleisin kirjain), toiseksi useimmin esiintyvä on ilmeisesti joko O tai T jne. Eipä muuta kuin kokeilemaan, miten salakirjoituksesta saisi järkevän näköistä tekstiä. Lasketaan ensimmäiseksi salakirjoituksessa olevien merkkien määrät.

S	7	L	3	T	2	V	1
J	6	M	3	A	2	K	1
P	5	B	3	I	2	H	1
		Z	3	E	2	C	1
		X	3				

Hyvältä näyttää: kun näitä esiintymistiheyksiä verrataan edelliseen taulukkoon, saa heti käsityksen, että S, J, P ovat ilmeisesti E, O, T (ehkä myös S, R tulevat kysymykseen), mahdollisesti jopa samassa järjestyksessä, ja L, M, B, Z, X taas ovat todennäköisesti S, R, A, I, N, mutta myös D ja jopa C, U ovat mahdollisia ehdokkaita. Katsotaan:

$S = E, J = O, P = T$
eteEOeCAoXeLtZMtHtAIBZeetMoT$oeoe$ELXXIBKVLToBZM

Eipä olekaan, $OEOE$ on kirjainyhdistelmä, jota tuskin englannista löytyy edes välilyönnillä. Järjestys $S = O, J = E$ ei ole sen parempi ($JSJS \rightarrow EOEO$). Eivät myöskään $J = T, S = E, P = O$ ($SPS \rightarrow EOE$) tai $J = T, S = O, P = E$ ($SPS \rightarrow OEO$) näytä yhtään paremmalta, joten yritetään arvausta $S = T$ kummankin vaihtoehdon $J = O, P = E$ ja $J = E$, $P = O$ kera:

$S = T, J = O, P = E$
tetEOeCAoXtLeZMeHeAIBZtteMoT$otot$ELXXIBKVLToBZM

$S = T, J = E, P = O$
totEEeCAeXtLoZMoHoAIBZttoMeT$etet$ELXXIBKVLTeBZM

Jälkimmäisen kokeilun alku $TOTE$ viittaa voimakkaasti selväkieliseen tekstiin 'To the ...', joten lisätään arvauksiin $E = H$, mikä sopii esiintymistiheytensä puolestakin mainiosti:

$S = T, J = E, P = O, E = H$
$tothe$CAeXtLoZMoHoAIBZttoMeT$eteth$LXXIBKVLTeBZM

Tietäen tiedoston sisältävän englanninkielistä mikrosanastoa silmiin pistää osa $E \quad ETE$, joka saattaisi hyvinkin olla $DELETE$. Kokeillaan $M = D, T = L$ (molemmat sopivat esiintymistiheydeltään):

$S = T, J = E, P = O, E = H, M = D, T = L$
$to the$CAeXtLoZdoHoAIBZ$ttodeleteth$LXXIBKVLleBZd

Delete sanan jälkeen näyttäisi sopivan $THIS$, siis $L = I, X = S$ (frekvenssit OK):

$S = T, J = E, P = O, E = H, M = D, T = L, L = I, X = S$
$to the$CA$estio$ZdoHoAIBZ$ttodeletethiss$IBKVileBZd

Kun muistaa, että kryptotekstin merkit B, Z ovat todennäköisesti kirjaimia $R, A, N(S, I)$, huomaa lopussa olevan yhdelmän BZM, mistä voi jo arvata sanan AND. Siispä kokeillaan $B = A, Z = N$

$S = T, J = E, P = O, E = H, M = D, T = L, L = I, X = S,$
$B = A, Z = N$
$to the$CA$estiondo$HoAI$anttodeletethiss$IaKVileand$

Nyt näyttää lupaavalta: "To The Question 'Do You Want To Delete' ...".

$S = T, J = E, P = O, E = HM = D, T = L, L = I, X = S,$
$C = Q, A = U, Z = N, H = Y, I = W, B = A, K = P, V = F.$
$tothequestiondoyouwanttodeletethisswapfileand$

README-tiedostosta löytyy teksti

At the "Permanent swap file is corrupt" screen, type Y in response to the question "Do you want to delete this swap file?", and then press ENTER.

Kirjaimittain tapahtuvan korvaamisen yleistyksenä voi tietenkin suorittaa kryptauksen korvaamalla kirjainpareja tai -kolmikkoja. Voidaan myös sovitulla tavalla vaihtaa kirjaintenjärjestystä, kirjoittaa ylimääräisiä merkkejä tms. Se ei oleellisesti kuitenkaan paranna tietosuojaa. Esimerkin mukainen tilastollinen analyysi kielessä esiintyvien kirjainparien suhteellisista frekvensseistä toimii vielä tehokkaasti. Oleellinen parannus saadaan vasta, kun korvataan todella pitkiä ja vaihtelevan pituisia merkkijonoja pitkillä ja vaihtelevan pituisilla merkkijonoilla.

Caesar-järjestelmän modifikaatioita ovat esim. kirjainten muunlaiset permutaatiot sekä "turhien" kirjaimien lisääminen "merkitsevien" välin. Seuraavaa avainsanan ohjaamaa järjestelmää käytettiin paljon keskiajalla. Olkoon avainsana $VARIS$, jonka kirjainten järjestysluvut ovat 21, 1, 17, 9, 18. Tekstin 1. kirjainta siirretään nyt 21 askelta eteenpäin, 2. kirjainta 1 askel, 3. kirjainta 17 askelta jne. Teksti $TURUSSA\ TUULEE$ saa näin muodon $OVIDKNB\ KDMHFX$. Tämän järjestelmän käyttöä nopeuttaa ns. Vigenére-taulu, 25 kertaa 25 matriisi, jossa vaaka- ja pystyrivit on indeksoitu kirjaimin aakkosjärjestyksessä. Kirjaimen A indeksoima vaakarivi on $BC...ÖA$, kirjaimen B indeksoima $CD...ÖAB$ jne. Salakirjoitus saadaan nyt lukemalla tekstiä vaakarivi-indekseistä ja avainsanaa pystyrivi-indekseistä.

Cardano keksi vielä merkittävän muutoksen: hän käytti tekstiä itseään avainsanana edellämainitussa mielessä. Avainsanan käyttö seuraavalla tavalla on sitä vastoin jo oleellisesti toisenlaista. Avainsana $VARIS$ (jossa kaikki kirjaimet ovat erilaisia) kirjoitetaan kirjainneliön alkuun ja sen jälkeen muut kirjaimet aakkosjärjestyksessä:

$$\begin{array}{ccccc} V & A & R & I & S \\ B & C & D & E & F \\ G & H & J & K & L \\ M & N & O & P & T \\ U & X & Y & Ä & Ö \end{array}$$

Teksti $TURUSSA\ TUULEE$ jaetaan kahden kirjaimen osiin pitäen täytekirjaimen X avulla huoli siitä, ettei mihinkään osaan tule kahta kirjainta:

$TU\ RU\ SX\ XS\ AT\ UX\ XU\ LE\ EX.$

Kukin osa kryptataan nyt seuraavan periaatteen mukaisesti. Tarkastellaan osaa TU ja etsitään neliöstä kirjaimet T ja U. Ne ovat sellaisen suorakulmion kärkinä, jonka muina kärkinä ovat (TU-järjestys säilyttäen) M ja $Ö$. TU:n kryptaus on nyt $MÖ$. Jos kirjaimet ovat samalla vaaka- tai pystyrivillä (kuten U ja X esimerkissämme), siirrytään ao. rivillä yksi askel eteenpäin (mahdollisesti syklisesti). $TURUSSA\ TUULEE$ saa näin muodon:

$MÖ\ VY\ AÖ\ ÖA\ SN\ XY\ UÖ\ KF\ CÄ.$

Tällaisen avainsananeliön käyttö määrää erään salakirjoitusjärjestelmän eli kryptosysteemin: niin pian kuin avain tunnetaan, voidaan kryptata ja dekryptata. Purkamisessa avainta tuntematta on oleellista saada tarpeeksi iso näyte salakirjoitusta. Kun tällainen on saatu, on vain ajan kysymys, että se saadaan avattua. Ratkaiseva merkitys on siis sillä, kuluuko tähän (parhaita tietokoneita käyttäen) muutamia sekunteja vai miljardi vuotta. Kehitys kulkeekin tämän johdosta suuntaan, jossa etsitään matemaattisesti vaikeita ongelmia ja sovelletaan niitä tietosuojan kehittämiseen. Toisena pyrkimyksenä on tehdä salakirjoituksesta ns. julkista. Toisin sanoen kirjoittamiseen ja avaamiseen tarvitaan eri avain. Tällöin voidaan periaatteessa päästä siihen tilanteeseen, että kuka tahansa pystyy salakirjoittamaan, mutta vain valitut lukemaan tätä tekstiä. Tämän luonteisia tarpeita on helppo nähdä esim. pankkitietojen välitystilanteissa.

Mainittakoon vielä, että salakirjoitustarkoituksiin on kehitetty salakirjoituskoneita. Ensimmäisen tällaisen keksi Thomas Jefferson 1790-luvulla. Ruotsalainen Boris Hagel keräsi 1930-luvulla melkoisen omaisuuden valmistamalla kirjoituskoneen näköisiä laitteita, jotka käänsivät selvää tekstiä salakielelle ja päivastoin. Toisen maailmansodan aikaisiin tiedustelu- ja vakoilutoimiin liittyvät olennaisesti erilaiset salakirjoituskoneet. Salakirjoituskone muuttaa aakkoston korvaamisjärjestelmää tiheästi usein jo yhden kirjaimen tehtyään. Tämä riippuu koneen mekaanisesta tai sähköisestä rakenteesta. Käyttämällä perätysten lukuisia erilaisia hammaspyöriä tai sähköisiä kytkentöjä voidaan korvausjär-

jestelmien luku saada hyvinkin suureksi. Viestittäjien kesken on sovittava koneen alkuasento tai ilmoitettava se esimerkiksi avainsanan avulla. Salakirjoituskoneita olivat mm. C-36 or M-209 Converter, Enigma, Sigaba, Red ja Purple.

C-36

Enigma

Seuraava mutkikkaampi salakirjoitusjärjestelmän on kehittänyt Hill. Yksinkertaisuuden vuoksi kerrotaan siitä esimerkin avulla:

Ensin numeroidaan aakkosto $A = 0, B = 1, \ldots, Z = 25$.

A	B	C	D	E	F	G	H	I	J	K	L	M	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13
O	P	Q	R	S	T	U	V	W	X	Y	Z		
14	15	16	17	18	19	20	21	22	23	24	25		

Seuraavassa kaikki laskutoimitukset tehdään mod 26. Tämä tarkoittaa sitä, että luku korvataan jakojäännöksellä, 26:lla jaon jälkeen.Esimerkiksi $54 \equiv 2 \bmod 26$ (”54 on kongruentti 2:n kanssa modulo 26”). Valitaan kokonaisluku d. Se on kryptausmatriisin dimensio ja toisaalta salakirjoitukseen liittyvä lohkopituus. Olkoon $d = 3$ ja kryptausmatriisi

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 3 & 2 \\ 4 & 5 & 11 \end{bmatrix}$$

Käänteismatriisi mod 26 on

$$M^{-1} = \begin{bmatrix} 25 & 15 & 7 \\ 20 & 17 & 8 \\ 22 & 1 & 1 \end{bmatrix}$$

Jaetaan alkuperäinen teksti 3-pituisiin lohkoihin:

$$DOES\ IT\ MAKE\ SENSE \rightarrow DOE\ SIT\ MAK\ ESE\ NSE$$

Lukuina

$$\begin{bmatrix} D \\ O \\ E \end{bmatrix} = \begin{bmatrix} 3 \\ 14 \\ 4 \end{bmatrix}, \begin{bmatrix} S \\ I \\ T \end{bmatrix} = \begin{bmatrix} 18 \\ 8 \\ 19 \end{bmatrix}, \begin{bmatrix} M \\ A \\ K \end{bmatrix} = \begin{bmatrix} 13 \\ 0 \\ 10 \end{bmatrix},$$
$$\begin{bmatrix} E \\ S \\ E \end{bmatrix} = \begin{bmatrix} 4 \\ 18 \\ 4 \end{bmatrix}, \begin{bmatrix} N \\ S \\ E \end{bmatrix} = \begin{bmatrix} 13 \\ 18 \\ 4 \end{bmatrix}$$

Kryptataan DOE

$$M \begin{bmatrix} D \\ O \\ E \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 0 & 3 & 2 \\ 4 & 5 & 11 \end{bmatrix} \begin{bmatrix} 3 \\ 14 \\ 4 \end{bmatrix}$$
$$= \begin{bmatrix} 43 \\ 50 \\ 126 \end{bmatrix} \equiv \begin{bmatrix} 17 \\ 24 \\ 22 \end{bmatrix} \bmod 26 = \begin{bmatrix} R \\ Y \\ W \end{bmatrix}$$

Samoin jatko

$$\begin{bmatrix} S \\ I \\ T \end{bmatrix} \to \begin{bmatrix} 13 \\ 10 \\ 9 \end{bmatrix} = \begin{bmatrix} N \\ K \\ J \end{bmatrix}, \begin{bmatrix} M \\ A \\ K \end{bmatrix} \to \begin{bmatrix} 17 \\ 20 \\ 6 \end{bmatrix} = \begin{bmatrix} R \\ U \\ G \end{bmatrix}$$

$$\begin{bmatrix} E \\ S \\ E \end{bmatrix} \to \begin{bmatrix} 0 \\ 10 \\ 20 \end{bmatrix} = \begin{bmatrix} A \\ K \\ U \end{bmatrix}, \begin{bmatrix} N \\ S \\ E \end{bmatrix} \to \begin{bmatrix} 9 \\ 10 \\ 4 \end{bmatrix} = \begin{bmatrix} J \\ K \\ E \end{bmatrix}$$

Koko teksti on nyt

$$RYWNKJRUGAKUJKE$$

Salakirjoituksen murskaaminen on aikaisempia oleellisesti vaikeampaa, koska aakkoston suuruus ei ole oikeastaan 26 vaan 26^3. Nyt ei työskennellä 26 erilaisen merkin vaan 3-pituisten lohkojen kanssa. Jos tunnet matriisin M voit laskea käänteismatriisin M^{-1} ja käyttää sitä salakirjoituksen purkuun. Esimerkiksi

$$M^{-1} \begin{bmatrix} R \\ Y \\ W \end{bmatrix} = \begin{bmatrix} 25 & 15 & 7 \\ 20 & 17 & 8 \\ 22 & 1 & 1 \end{bmatrix} \begin{bmatrix} 17 \\ 24 \\ 22 \end{bmatrix}$$
$$= \begin{bmatrix} 939 \\ 924 \\ 420 \end{bmatrix} \equiv \begin{bmatrix} 3 \\ 14 \\ 4 \end{bmatrix} \bmod 26 = \begin{bmatrix} D \\ O \\ E \end{bmatrix}$$

jne.

Erikoismerkkien (□♮∇⊤◊♯♣♢ ♡♠□■♦▲Ⓢ...) käyttö ei oleellisesti hyödytö salakirjoituksessa, koska sinänsä ei se tuo mitään uutta, on vain keksittävä merkkien vastaavuus aivan kuten Caesarin johdannaisissa.

DES

Parhaana klassisena salakirjoitusjärjestelmänä pidetään Yhdysvalloissa 1970-luvun alussa keksittyä DES–järjestelmää (Data Encryptation Standard). Menetelmässä käytetään tehokkaasti hyväksi automaattisen tietojenkäsittelyn mahdollisuuksia. Idea standardista on vallankumouksellinen: aikaisemminhan pyrittiin salaamaan kaikki mahdolliset yksityiskohdat. DES-systeemi on julkaistu yksityiskohtia myöten. Tietysti DES:in käytössä on aina salainen avain. DESiä ei enää pidetä riittävän varmana kriittisen tiedon salaamiseen. Sen variantti 3DES on varmempi, mutta vastaan tulee suorituskykyongelmat. Uusin versio on AES (Advanced Encryption Standard), jossa avain pituus voi olla 128, 192 tai 256.

DES:n avaimeksi valitaan 56 merkkiä pitkä bittijono. Mahdollisia avaimia on siis 72058 biljoonaa. Tähän lisätään pariteetintarkistusmerkki kahdeksan bitin välein. Siis paikkoihin 8, 16, ..., 64. Merkki valitaan siten, että kahdeksan pituinen pätkä on aina pariton.

Seuraavaksi tämä 64 merkin jono järjestetään seuraavalla permutaatiolla:

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36

63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Tällöin muodostuu kaksi lohkoa C_0 ja D_0, molemmissa 28 bittiä. C_0:n ensimmäiset bitit ovat 57. , 49. ja 41. avaimen biteistä. Vastaavasti D_0:n viimeiset ovat avaimen 20., 12. ja 4. bitti.

Nyt muodostetaan lisää blokkeja C_1,C_2...,C_{15} ja $D_1, D_2, ..., D_{15}$. Blokit C_n ja D_n saadaan C_{n-1}:stä ja D_{n-1}:stä siirtämällä bittejä syklisesti (siirrossa "yli menevät" viedään loppuun) 1 tai kaksi askelta vasemmalle seuraavan taulukon mukaisesti:

n:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
siirtoja:	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Seuraavaksi määritellään 16 erilaista 48 alkion järjestettyä joukkoa K_n $n = 1, ..., 16$, jotka koostuvat avaimen biteistä. K_n koostuu seuraavista

C_n :n ja D_n:n biteistä:

14	17	11	24	1	5
3	28	15	6	21	10
23	19	12	4	26	8
16	7	27	20	13	2
41	52	31	37	47	55
30	40	51	45	33	48
44	49	39	56	34	53
46	42	50	36	29	32

Tämä kaikki oli tähän asti pelkästään valmistelua. Tehtiin avaimesta salakirjoituksessa tarvittavia työkaluja.

Seuraavaksi selvitetään, miten varsinainen salakirjoitus tapahtuu.

Salakirjoitettava teksti jaetaan 64 bitin jonoihin. Tällainen 64-pituinen blokki w järjestetään ensiksi seuraavan alkupermutoinnin avulla

58	50	42	34	26	18	10	2
60	52	44	36	28	20	12	4
62	54	46	38	30	22	14	6
64	56	48	40	32	24	16	8
57	49	41	33	25	17	9	1
59	51	43	35	27	19	11	3
61	53	45	37	29	21	13	5
63	55	47	39	31	23	15	7

Täten muokattu blokki w' alkaa alkuperäisen blokin bitistä 58 ja jatkuu bitillä 50. Jaetaan w' kahteen osaan L_0R_0 (L = left, R = right), jotka molemmat sisältävät 32 bittiä. Määritellään edelleen blokit L_n ja R_n $n = 0, 1, 2, ..., 16$. Tämä tehdään määrittelyillä

$$L_n = R_{n-1}$$
$$R_n = L_{n-1} + f(R_{n-1}, K_n)$$

missä + on bitti bitiltä suoritettu yhteenlasku mod 2 ja funktio f määritellään myöhemmin. Salakirjoitus saadaan nyt soveltamalla alkupermutoinnin käänteispermutointia blokkiin $L_{16}R_{16}$.

Salakirjoituksen avaaminen tapahtuu tekemällä kaikki toisinpäin. Esimerkiksi yhtälöillä

$$R_{n-1} = L_n$$
$$L_{n-1} = R_n + f(L_n, K_n)$$

laskeudutaan takaisin muotoon L_0R_0, josta alkupermutoinnin käänteispermutoinnilla päästään takaisin alkuperäiseen tekstiin. (Tietenkään ei alkuvalmisteluja, jotka koskivat K_n-blokkien rakentamista, tarvitse palauttaa.)

Selvitetään vielä, minkälainen funktio on f. Se yhdistää ensin 32-bittiset blokit

R_{n-1} ja K_n tai L_n ja K_n 48-bittiseksi blokiksi seuraavasti: Ensimmäisestä muuttujasta R_{n-1} tai L_n muodostetaan 48-bittinen seuraavalla järjestyksellä ja toistoilla

32	1	2	3	4	5
4	5	6	7	8	9
8	9	10	11	12	13
12	13	14	15	16	17
16	17	18	19	20	21
20	21	22	23	24	25
24	25	26	27	28	29
28	29	30	31	32	1

Näin muodostuneeseen blokkiin lisätään sitten K_n bitti bitiltä mod 2. Summablokki jaetaan 6-bittisiin osablokkeihin $B = B_1B_2...B_8$. Näitä muutetaan vastaavasti seuraavien taulukkojen $S_1, S_2, ..., S_8$ avulla.
S_1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

S_2

15	1	8	14	6	11	3	4	9	7	2	13	12	0	5	10
3	13	4	7	15	2	8	14	12	0	1	10	6	9	11	5
0	14	7	11	10	4	13	1	5	8	12	6	9	3	2	15
13	8	10	1	3	15	4	2	11	6	7	12	0	5	14	9

S_3

10	0	9	14	6	3	15	5	1	13	12	7	11	4	2	8
13	7	0	9	3	4	6	10	2	8	5	14	12	11	15	1
13	6	4	9	8	15	3	0	11	1	2	12	5	10	14	7
1	10	13	0	6	9	8	7	4	15	14	3	11	5	2	12

S_4

7	13	14	3	0	6	9	10	1	2	8	5	11	12	4	15
13	8	11	5	6	15	0	3	4	7	2	12	1	10	14	9
10	6	9	0	12	11	7	13	15	1	3	14	5	2	8	4
3	15	0	6	10	1	13	8	9	4	5	11	12	7	2	14

S_5

2	12	4	1	7	10	11	6	8	5	3	15	13	0	14	9
14	11	2	12	4	7	13	1	5	0	15	10	3	9	8	6
4	2	1	11	10	13	7	8	15	9	12	5	6	3	0	14
11	8	12	7	1	14	2	13	6	15	0	9	10	4	5	3

S_6

12	1	10	15	9	2	6	8	0	13	3	4	14	7	5	11
10	15	4	2	7	12	9	5	6	1	13	14	0	11	3	8
9	14	15	5	2	8	12	3	7	0	4	10	1	13	11	6
4	3	2	12	9	5	15	10	11	14	1	7	6	0	8	13

S_7

4	11	2	14	15	0	8	13	3	12	9	7	5	10	6	1
13	0	11	7	4	9	1	10	14	3	5	12	2	15	8	6
1	4	11	13	12	3	7	14	10	15	6	8	0	5	9	2
6	11	13	8	1	4	10	7	9	5	0	15	14	2	3	12

S_8

13	2	8	4	6	15	11	1	10	9	3	14	5	0	12	7
1	15	13	8	10	3	7	4	12	5	6	11	0	14	9	2
7	11	4	1	9	12	14	2	0	6	10	13	15	3	5	8
2	1	14	7	4	10	8	13	15	12	9	0	3	5	6	11

Edellä olevia taulukoita käytetään seuraavasti: Taulukolla S_k, operoidaan lohkoon B_k. Olettakaamme, että B_7 on esim. 110010. Ensimmäisestä ja viimeisestä bitistä muodostuu luku $x = 10$ eli 2 kymmenjärjestelmässä. Tämä luku on aina välillä 0:sta 3:een. Samalla tavoin 4 keskimmäistä bittiä muodostavat luvun $b = 1001$ eli 9 kymmenjärjestelmässä. Tämä on puolestaan aina välillä 0:sta 15:een. S_7:n sarakkeet numeroidaan 0:sta 15:een ja vaakarivit 0:sta 3:een. Nyt poimitaan kaavion paikasta $(2, 9)$ alkio15, jonka binääriesityksellä 1111 korvataan B_7. Syntyneeseen uuteen lohkojonoon $B' = B'_1B'_2...B'_8$ sovelletaan permutaatiota

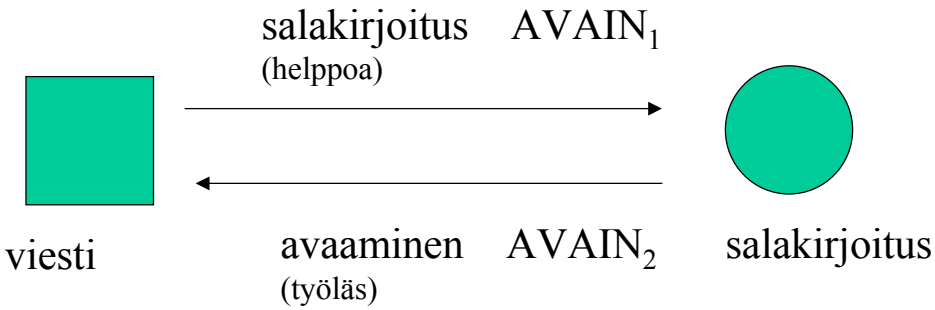
16	7	20	21
29	12	28	17
1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

DES on epäilemättä paperilla luettuna sekavan ja pitkän näköinen, mutta se soveltuu erittäin hyvin tietokoneella operoitavaksi, koska kaikki suoritetut operaatiot ovat helposti ohjelmoitavissa ja nopeita suorittaa. Ideana on siis lohkoa, yhdistää, muuttaa ja sekoittaa tekstiä vastaavaa bittikoodia riittävän siinä määrin, että minkäänlaiset arvaamiset eivät voi johtaa menestykseen.

RSA

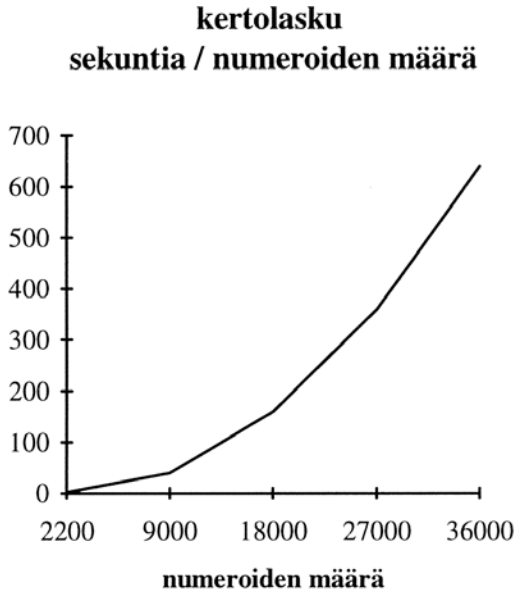
RSA-salakirjoituksen nimi tulee matemaatikkojen *Rivest, Shamir* ja *Adleman* nimien alkukirjaimista. Se on laajimmin käytetty julkisen salakirjoituksen muoto. Sitä käytetään mm. GSM-matkapuhelinten datan sekoituksessa. Kryptaus perustuu melko yksinkertaiseen peruslukuteorian tulokseen ns. *Eulerin lauseeseen*.

Julkisen salakirjoituksen idea on siinä, että jotkut probleemat ovat yksinkertaisempia toiseen kuin päinvastaiseen suuntaa. - Ajattele esim. puhelinnumeron etsimistä puhelinluettelosta nimen perusteella verrattuna siihen, jos pitäisikin löytää nimi numeron perusteella. Tämän kaltaiisella ajatuksella voidaan salakirjoitus julkistaa (vastaa numeron etsimistä nimen perusteella). Samalla kuitenkin salakirjoitetun tekstin tulkitseminen pysyy salaisena (vastaa nimen etsimista numeron perusteella).



RSA-kryptokoodin matemaattinen perusta on laskennallinen kompleksisuus. Kahden k-numeroisen luvun (luvut $n = 10^k$) yhteenlasku on lineaarinen numeroiden määrän k suhteen eli suoritusaika T^+ on kertaluokkaa $T^+ = O(k) = O(\log n)$. Vastaavasti kerto- ja jakolaskut koulualgoritmeilla ovat neliöllisiä eli suoritusaika on $T^* = O(k^2) = O((\log n)^2)$. Sen sijaan kokonaisluvun n tekijöihinjako on numeroiden määrän k suhteen eksponentiaalinen. Yksinkertaisin algoritmi on tarkistaa n:n jaollisuus kaikilla kokonaisluvuilla $\leq \sqrt{n}$, kunnes tekijät ovat löytyneet tai voidaan todeta, ettei n:llä ole tekijöitä. Tämän algoritmin kompleksisuus on kertaluokkaa $T = O(\sqrt{n}) = O(n^{1/2}) = O(10^{(\log n)/2}) = O(10^{k/2})$. Parhaimmillakin algoritmeilla saadaan eks-

ponentti $k/2$ vain hieman pienemmäksi.



Käytännössä tämä kompleksisuus merkitsee, että 1000-numeroisilla luvuilla voi laskea omalla kotimikrollaan, mutta 1000-numeroisen luvun tekijöihinjakoon ei nykyisten superkoneiden kapasiteetti enää riitä. Tavallinen kotimikro suoriutuu kahden yli 147.000-numeroisen luvun yhteenlaskusta $0.06 - 0.16$ sekunnissa. Kertolaskuun kuluneista ajoista oheinen kaavio. Pienet alkuluvut seuloutuvat nopeasti: kaikki alkuluvut $\leq$ 175.000 löytyivät tehdyssä testissä 5.7 sekunnissa. Kun 1000-numeroisen luvun jaollisuuden näillä korkeintaan 5-numeroisilla alkuvuilla saa tarkistettua minuutissa, se ei suinkaan tarkoita, että saman luvun saisi jaettua täydellisesti tekijöihin kohtuullisessa ajassa. Täydellinen tekijöihinjakohan vaatii jaollisuustarkistuksen myös 500-numeroisilla luvuilla. Näitä taas on lukumääräisesti 10500. Kun vuodessa on ainoastaan $60 \cdot 60 \cdot 24 \cdot 360 = 31.104.000 < 10^8$ sekuntia, tekijöihin jako on käytännössä mahdoton. (Edellä kuvatut laskelmat kotimikrolla ovat muutaman vuoden vanhoja. ja kaiken aikaa vanhenevia. Täten lukuja ei ole syytä ymmärtää absoluuttisina vaan suuntaa antavina.)

Kesällä 2002 tuli julkisuuteen tieto, jonka mukaan alkulukutarkistus on saatu toimimaan polynomiaalisesti (Agrawal, M.; Kayal, N.; and Saxena, N. "Primes in P.") aikaisemman eksponentiaalisen sijaan. Tämä sellaisenaan ei uhkaa mitenkään RSA:ta, mutta luo uhkakuvia siitä, että tulevaisuudessa voi tekijöihinjakokin oleellisesti nopeutua. Käytännössä se aiheuttaa paineita uusien salausmenetelmien kehittämiseen.

RSA-kryptokoodi.

Valitaan kaksi (isoa) alkulukua p ja q, lasketaan niiden tulo $n = pq$ sekä valitaan kokonaisluvut e ja d siten, että

$$ed \equiv 1(\mathrm{mod}(p-1)(q-1))$$

eli $ed = k(p-1)(q-1)+1$ jollekin kokonaisluvulle k.

Merkitään $\mathbb{Z}_n$:llä kokonaislukuja mod n. Toisin sanoen samaistetaan ne kokonaisluvut, jotka ovat kongruentteja mod n. Edustajistona $\mathbb{Z}_n$:lle voidaan ajatella lukuja $0, 1, 2, ..., n-1$ tai mitä tahansa n:n peräkkäisen kokonaisluvun joukkoa. Kryptokoodin koodauksena (kryptaus) on funktio

$$f : Z_n \to Z_n, \quad f(x) = x^e (= x^e \mod n)$$

ja dekoodauksena funktio

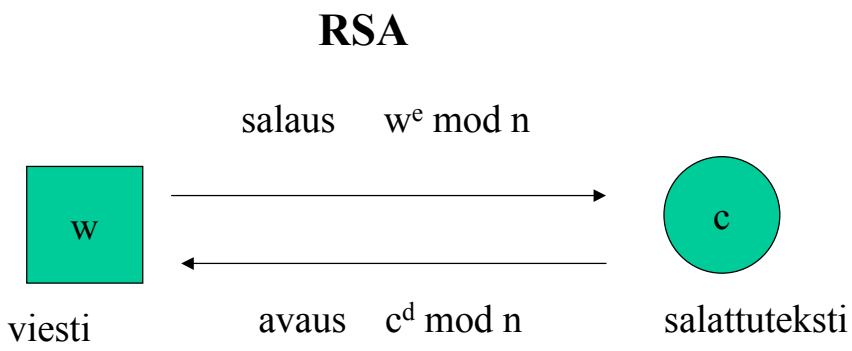
$$g : Z_n \to Z_n, \ g(y) = y^d (= y^d \mod n)$$

Luvut e ja d voidaan valita valitsemalla ensin e (kokeillen) siten, että

$$\text{syt}(e, (p-1)(q-1)) = 1$$

ja ratkaisemalla sen jälkeen d kongruenssista

$$ez \equiv 1 (\text{mod}(p-1)(q-1))$$



RSA on julkinen kryptokoodi, mikä merkitsee, että sen koodaus eli n ja e julkistetaan kaikkien (asianomaisten) käytettäväksi, mutta dekoodaus eli d pidetään salaisena ja samoin tietenkin luvut p, q ja $\varphi(n)$. Se, että koodaus voidaan julkistaa, perustuu johdantona esitettyyn tekijöihinjaon kompleksisuuteen: alkulukujen p ja q laskeminen suuresta luvusta n on käytännössä mahdoton eikä luvun e julkistaminenkaan helpota tekijöihinjakoa (paitsi erityistapauksissa).

Esimerkki 29 *Valitaan alkuluvut* $p = 31$ *ja* $q = 37$, *jolloin* $n = pq = 31 \cdot 37 = 1147$ *ja* $\varphi(n) = \varphi(pq) = (p-1)(q-1) = 30 \cdot 36 = 1080$. *Koska* $1080 + 1 = 23 \cdot 47$, *voidaan valita* $e = 23$ *ja* $d = 47$. *Julkistetaan* $n = 1147$ *ja* $e = 23$, *mutta pidetään* $p = 31, q = 37, \varphi(n) = 1080$ *ja* $d = 47$ *salaisina.*

Jokainen voi lukuja $n = 1147$ ja $e = 23$ käyttäen koodata ilman tehokasta laskintakin esimerkiksi luvun 7 (oletetaan kirjaimet numeroiduksi $A = 1, B = 2, C = 3, ...$):

$$\begin{aligned}
7^{23} &= 7^{16+4+2+1} \\
&= 7^1 7^2 7^4 7^{16} \\
&= 7^1 7^2 \left(7^2\right)^2 \left(7^4\right)^4 \\
&= 7 \cdot 49 \cdot 49^2 \cdot \left(49^2\right)^4 \\
&= 343 \cdot 2401 \cdot 2401^4 \\
&\equiv 343 \cdot 107 \cdot 107^4 && \mod 1147 \\
&= 36701 \cdot 107^2 \cdot 107^2 \\
&\equiv -3 \cdot 11449 \cdot 11449 && \mod 1147 \\
&\equiv -3 \cdot (-21) \cdot (-21) && \mod 1147 \\
&= -1323 \\
&\equiv 971 && \mod 1147
\end{aligned}$$

Luku 7 (kirjain $G = 7$) on näin kryptattu luvuksi 971 eli $f(7) = 7^e = 7^{23} = 971 \mod 1147$. Dekryptauksen g pitää pystyä laskemaan luvusta 971 alkuperäinen 7. Lasketaan, että $g(971) = 971^d = 971^{47} = 7 \mod 1147$:

$$\begin{aligned}
971^{47} &\equiv (-176)^{32+8+4+2+1} && \mod 1147 \\
&= -176 \cdot (-176)^2 \cdot (-176)^4 \cdot (-176)^8 \cdot (-176)^{32} \\
&= -176 \cdot 30976 \cdot (30976)^2 \cdot (30976)^4 \cdot (30976)^{16} \\
&\equiv -176 \cdot 7 \cdot 7^2 \cdot 7^4 \cdot 7^{16} && \mod 1147 \\
&= -176 \cdot 7^{23} \\
&\equiv -176 \cdot 971 && \mod 1147 \\
&\equiv -176 \cdot (-176) && \mod 1147 \\
&\equiv 7 && \mod 1147
\end{aligned}$$

Esimerkissä olisi luvuille e ja d voinut valita myös muita arvoja, esimerkiksi $e = 23 + 1080 = 1103$ ja $d = 47$, sillä $(1080 + 23) \cdot 47 \equiv 1 (\mod 1080)$. Yhtä hyvin voitaisiin valita $e = 7$ ja $d = 463$, sillä $7 \cdot 463 = 3241 = 3 \cdot 1080 + 1 \equiv 1 (\mod 1080)$.

Jotta kysymyksessä olisi todella koodi, on koodauksen $f(x) = x^e (\mod n)$ ja dekoodauksen $g(y) = y^d (\mod n)$ oltava toistensa käänteiskuvauksia eli olipa x mikä tahansa luku. Tässä asiassa onnistutaan Eulerin lauseen avulla.

Lause 30 *Jos p ja q ovat alkulukuja sekä $n = pq$, niin*

$$x^{(p-1)(q-1)+1} \equiv x (\mod n) \text{ kaikille kokonaisluvuille } x$$

Todistus. Koska $\varphi(n) = \varphi(pq) = (p-1)(q-1)$, niin Eulerin lauseen perusteella

$$x^{(p-1)(q-1)+1} = x^{\varphi(n)+1} \equiv x (\mod n) \text{ aina, kun } \operatorname{syt}(x, n) = 1$$

Tämä pätee myös muulloin. Jos esimerkiksi $x = rp^a$, missä $\operatorname{syt}(r, n) = 1$, niin

$$r^{\varphi(n)+1} \equiv r \bmod n$$

$$x^{\varphi(n)+1} = (rp^a)^{\varphi(n)+1} = (r)^{\varphi(n)+1} p^a \left(p^a\right)^{(p-1)(q-1)} \equiv rp^a \left(p^{a(p-1)}\right)^{(q-1)} \bmod n$$

Koska p ja siis myös $p^{a(p-1)}$ on q:lla jaoton ja koska $\varphi(q) = q - 1$, niin edelleen

$$\left(p^{a(p-1)}\right)^{(q-1)} \equiv 1 \bmod n$$

eli

$$\left(p^{a(p-1)}\right)^{(q-1)} = tq + 1$$

jollekin kokonaisluvulle t. Mutta tällöinhän

$$x^{\varphi(n)+1} \equiv rp^a \left(p^{a(p-1)}\right)^{(q-1)} = rp^a \left(tq + 1\right) = rp^a tq + rp^a \equiv rp^a = x \bmod n$$

kuten haluttiin osoittaakin.

Koska RSA-kryptokoodissa $ed = k(p-1)(q-1) + 1 = k\varphi(n) + 1$, niin Lauseen perusteella myös

$$(x^e)^d = x^{ed} = x^{k\varphi(n)+1} = \left(x^{\varphi(n)+1}\right)^k x^{1-k} \equiv x^k x^{1-k} \equiv x \bmod n$$

Esimerkki 31 *$p = 3, q = 13, n = pq = 39, \varphi(n) = (p-1)(q-1) = 24, x = 6 = 2p$. Osoitetaan laskemalla, että $x^{\varphi(n)+1} \equiv x \bmod n$*

$$\begin{aligned} 6^{25} &= 6 \cdot 6^8 \cdot 6^{16} = 6 \cdot 36^4 \cdot 36^8 \equiv 6 \cdot (-3)^4 \cdot (-3)^8 && \bmod 39 \\ &= 6 \cdot 81 \cdot 81^2 \equiv 6 \cdot 3 \cdot 3 && \bmod 39 \\ &= 6 \cdot 27 \equiv 162 = 4 \cdot 39 + 6 \equiv 6 && \bmod 39 \end{aligned}$$

Esimerkki 32 *Kryptataan teksti $SALAINEN$. Numeroidaan kirjaimet: $A = 01, B = 02, ..., Ö = 28$ (huomaa etunollat). Sen jälkeen kryptataan RSA-koodilla $n = 1147, e = 23$.*

$SALAINEN$	$\rightarrow$	$19, 01, 12, 01, 09, 14, 05, 14$	*kirjainten numerointi*
	$\rightarrow$	1901120109140514	*pilkut pois*
	$\rightarrow$	$190, 112, 010, 914, 051, 400$	*ryhmään $\mathbb{Z}_{1147}$ ylim 0:*
	$\rightarrow$	$190^{23}, 112^{23}, 10^{23}, 914^{23}, 51^{23}, 400^{23}$	*RSA-koodaus*
	$\rightarrow$	$1056, 0815, 1099, 0306, 0267, 0361$	$\bmod 1147$
	$\rightarrow$	105608151099030602670361	*pilkut pois*

Mikäli tämän haluaa esittää kirjaimina, joutuu kuitenkin jättämään numeroita:

$$\begin{aligned} &105608151099030602670361 \rightarrow \\ &10, 56, 08, 15, 10, 99, 03, 06, 02, 67, 03, 61 \rightarrow \\ &J56HOJ99CFB67C61 \end{aligned}$$

Katsotaan vielä tämän salakirjoituksen $J56HOJ99CFB67C61$ dekoodaus:

$J56HOJ99CFB67C61$	
$\rightarrow 10, 56, 08, 15, 10, 99, 03, 06, 02, 67, 03, 61$	*kirjainten numerointi*
$\rightarrow 105608151099030602670361$	*pilkut pois*
$\rightarrow 1056, 0815, 1099, 0306, 0267, 0361$	*ryhmään $\mathbb{Z}_{1147}$*
$\rightarrow 1056^{47}, 815^{47}, 1099^{47}, 306^{47}, 267^{47}, 361^{47}$	*RSA-dekoodaus*
$\rightarrow 190, 112, 010, 914, 051, 400$	$\bmod 1147$
$\rightarrow 190112010914051400$	*pilkut pois*
$\rightarrow 19, 01, 12, 01, 09, 14, 05, 14, 00$	*pilkut toisin*
$\rightarrow SALAINEN$	*ja kirjaimiksi*

Käytetyt kaksi erilaista pilkutusta (19,01,12 → 190,112) sotkevat kirjainten väliset rajat ja lisäksi tekstiin tulee 'roskaa' (190,112 → 1056,0815):

	$SALAINEN$	
kirjainten korvaus	$QHYHBZWZ$	sanan rakenne näkyy
RSA	$J56HOJ99CFB67C61$	rakenne muuttunut

Esimerkiksi A näkyy korvatessa molemmilla kerroilla H:na.

Esimerkki 33 *Salakuuntelijoina olemme siepanneet edellisestä esimerkistä koodiluvun $z = 1099$ ja jostain olemme saanet tietoomme myös modulin $n = 1147$, mutta muuta emme tiedäkään (e on julkinen vain asianomaisille). Yritämme murtaa koodin laskemalla $\varphi(n)$:n arvon.*

Ratkaistaan p ja q yhtälöistä

$$n = pq \qquad \varphi(n) = (p-1)(q-1)$$

Saadaan

$$\varphi(n) = (p-1)(q-1) = pq - p - q + 1 = n - p - q + 1 = n - p - n/p + 1$$

eli

$$p^2 - (n + 1 - \varphi(n))p + n = 0$$

mistä

$$p = \frac{1}{2}\left((n + 1 - \varphi(n)) \pm \sqrt{(n + 1 - \varphi(n))^2 - 4n}\right)$$

Tarvitsee vain pystyä laskemaan se oikea $\varphi(n)$:n arvo, jolla ratkaisukaava antaa kokonaislukuratkaisun p, joka on myös n:n tekijä. Kaikkia lukuja $\varphi(n) = 1, ..., n$ ei tarvitse kokeilla, sillä edellä olevista yhtälöistä saadaan arviot

$$\varphi(n) = n - (p + n/p) + 1 \leq n - (\sqrt{n} + n/\sqrt{n}) + 1 = n - 2 + 1 = 1080.2$$

koska $2 \leq p \leq \sqrt{n}$ (oletetaan $p < q$)

$$\varphi(n) = n - (p + n/p) + 1 \geq n - (2 + n/2) + 1 = n/2 - 1 = 572.5$$

Tämä jättää vielä 508 mahdollista $\varphi(n)$:n arvoa kokeiltavaksi.

Tiedetään että $\text{syt}(z, n) = \text{syt}(1099, 1147) = 1$ ja siis $z^{\varphi(n)} \equiv 1 (\text{mod}\, n)$. Niinpä ryhdytään laskemaan:

$$\begin{array}{lll} z = 1099 & z^2 = 10 & z^3 = 667 \\ z^4 = 100 & z^5 = 935 & z^6 = 1000 \\ z^7 = 174 & z^8 = 824 & z^9 = 593 \\ z^{10} = 211 & z^{11} = 195 & z^{12} = 963 \\ z^{13} = 803 & z^{14} = 454 & z^{15} = 1 \end{array}$$

Sanotaan, että luvun $z = 1099$ kertaluku $\bmod n$ on 15. Alkion kertaluku jakaa aina luvun $\varphi(n)$. Täten tiedämme myös, että $\varphi(n)$ on 15:llä jaollinen. Tämä vähentää huomattavasti kokeiltavia $\varphi(n)$:n arvoja: välillä $573 - 1080$ olevista kokonaisluvuista 15:sta jaollisia ovat $585, 600, 615, ..., 1065, 1080$ eli 34 kappaletta. Näistä esimerkiksi $\varphi(n) = 585$ antaisi

$$p^2 - (1148 - 585)p + 1147 = 0, \quad p^2 - 563p + 1147 = 0$$

$$p = \frac{1}{2}\left(563 \pm \sqrt{563^2 - 4 \cdot 1147}\right) = \frac{1}{2}\left(563 \pm \sqrt{312\,381}\right) = \frac{1}{2}\left(563 \pm 558.91\right)$$

mistä kumpikaan ratkaisu ei ole kokonaisluku. Tarkistapa, että arvolla $\varphi(n) = 1080$ saadaan oikeat ratkaisut $p = 31, q = 37$.

Esimerkki tuo esille tekijöihinjaolle vaihtoehtoisen tavan RSA-kryptokoodin murtamiseen. Laskennallinen kompleksisuus on kuitenkin edelleen ratkaiseva tekijä RSA:n luotettavuudessa: jos n on suuri (200-numeroinen), niin myös lukujen kertaluvut ovat suuria ja niiden laskeminen sekä kaikkien mahdollisten $\varphi(n)$-arvojen kokeileminen esimerkin mukaisesti on käytännössä mahdotonta.

Tulevaisuuden näkökulmia

Seuraavassa esitetään joitakin julkisen salakirjoituksen käyttömahdollisuuksista.

Supervalloilla on eräissä äänestyksissä, mm. YK:n turvallisuusneuvostossa veto-oikeus. Toisin sanoen päätökseksi tulee "ei", jos supervalta on käyttänyt oikeuttaan, vaikka enemmistö olisikin päätöslauselman hyväksymisen kannalla.

Ajatelkaamme nyt seuraavaa tilannetta. Kaikilla on mahdollisuus äänestää "jaa" tai "ei", mutta supervallat voivat myös äänestää "jaa jaa" tai "ei ei". Jos näitä kaksoisääniä ei käytetä, ratkaisee enemmistö. Muutoin ratkaisevat kaksoisäänet. Tasatuloksen syntyessä on päätös "jaa". Voidaanko tälläinen äänestys suorittaa niin, ettei kukaan tiedä, kuinka toiset ovat äänestäneet? Jos esimerkiksi päätökseksi on julistettu "ei", yksinkertaisen "ei"-äänen käyttäjä ei tietäisi, onko syynä enemmistön kanta vai supervallan kaksoisääni. Samoin kaksoisäänen antaja ei tietäisi olisiko sama lopputulos syntynyt myös enemmistöpäätöksellä.

Tietenkin on mahdollista rakentaa äänestyskone, joka antaa päätöksen esitettyjen sääntöjen mukaan. Todella tärkeissä kysymyksissä kukaan ei kuitenkaan luottaisi tähän koneeseen. Tästä on esimerkkinä se, että tällaista suhteellisen yksinkertaista laitetta ei ole vielä rakennettu. Julkisen salakirjoituksen avulla voidaan kuvatunlainen äänestys suorittaa salaisena lippuäänestyksenä, vieläpä ilman puolueettomia äänten-laskijoita. Äänestysohjeesta tosin tulee melko monimutkainen. Tämä on

esimerkki yleisestä ongelmasta: osapuolet tuntevat oman salaisuutensa ja haluavat vaihtaa osan siitä, mutta pitää loput omana tietonaan.

Seuraavalla esimerkillä ei ole sanottavaa merkitystä käytännössä, mutta itse julkisen salakirjoituksen soveltamisen kannalta se on kuvaava. Kysymys on siitä miten voidaan puhelimessa heittää luotettavasti kruunaa ja klaavaa eli suorittaa arvonta.

Aviopuolisot ovat eronneet ja asuvat varsin etäällä toisistaan. He haluavat ratkaista kumpi saa heidän vanhan autonsa. Ongelmana on, ettei kumpikaan luota toiseen eikä käytettävissä satu olemaan puolueetonta tuomaria. Kruunan ja klaavan heittäminen puhelimessa käy päinsä julkisen salakirjoituksen avulla ja kumpikin saa halutessaan selville toisen mahdollisen petosyrityksen.

Viimeinen esimerkki on samaa tyyppiä kuin salainen äänestys. Kaksi naista haluaa tietää, kumpi on vanhempi, mutta he eivät halua paljastaa omaa ikäänsä toiselle. Keskustelun alkaessa kumpikaan ei tiedä mitään toisen iästä. Keskustelun päättyessä molemmat tietävät, kumpi on vanhempi, mutta eivät ole saaneet muuta tietoa toisen iästä.

Salakirjoitus selkäreppumenetelmällä.

Tämä esimerkki antaa RSA:n tyyppisen salakirjoitussysteemin. Kyseessä on lähinnä teoreettisen tietojenkäsittelyn parissa tutkitun ns. selkäreppu-probleeman sovellus. Ongelman epämatemaattinen formulointi voisi olla esim., millä tavalla pakata annetut tavarat kovaan matkalaukkuun, kun tiedetään, että ne sopivasti asetettuna sopivat sinne eikä jää yhtään tyhjää tilaa. Tavallaan ollaan siis tekemisissä palapelin ratkaisemisen kanssa.

Merkitään aakkoset esim. seuraavasti A on 00000, B 00001, C 00011 jne., ts. 5-pituisena binäärimuodossa. Salakirjoituksen muodostamiseksi selväkielestä tarvitaan kokonaislukujen jono, esimerkiksi

$$t' = (43, 129, 215, 473, 903, 302, 561, 1165, 697, 1523)$$

Selvätekstiä salakirjoitetaan 2 kirjainta eli 10 bittiä kerralla. Jos nämä kirjaimet ovat esim. L ja M, merkitään

$$LM = 0101101100$$

Nämä bitit pannaan nyt vastaamaan t'-jonon kymmentä lukua alusta lähtien ja lasketaan ne t':n luvut yhteen, joiden kohdalla on 1.

$$129 + 473 + 903 + 561 + 1165 = 3231$$

Saatu summa on salakirjoitettu versio. Salakirjoituksen tulkitsemiseksi pitää jonosta t' löytää sellaiset luvut, joiden summa on 3231.

Esimerkin salakirjoituksen tulkinta on helppoa kokeilemalla, koska jono ja luvut ovat pieniä. Jos t':ssa olisi esim. 200 lukua, jolloin selvätekstiä kirjoitettaisiin 40 merkkiä kerrallaan, ei kokeileminen enää onnistuisi edes tehokkailla tietokoneilla.

Jos Matti haluaa, että hänelle lähetetään viestejä salakirjoitettuna selkäreppumenetelmällä, hän voi huoletta julkistaa jonon t' ja kirjainten biteiksimuuttamistavan. Jonon on tietenkin oltava riittävän pitkä, ettei viestejä voi kokeilemalla tulkita. Nyt kuka tahansa voi kirjoittaa Matille, mutta vain Matti pystyy lukemaan viestit. Lähtökohta on, että vain Matti tietää, että jono t' on muodostettu tietyllä tavalla toisesta jonosta

$$t = (1, 3, 5, 11, 21, 44, 87, 175, 349, 701)$$

Tämän jonon Matti pitää omana tietonaan samoin kuin lukuparin $(37, 1590)$, jonka avulla julkinen jono t' on saatu t:stä.

Jono t on ns. superkasvava jono eli jokainen sen luku on aina suurempi kuin edellisten summa. Tämä tekee sen helpoksi tulkita kokeilemalla. Matilla on vielä luku 43, joka on sellainen, että $43 \cdot 37 \equiv 1 \bmod 1590$. Jono t' saadaan t:stä kertomalla kaikki t:n luvut 43:lla mod 1590.

Tulkitaan nyt viesti 3231. Kerrotaan ensin luku 37:llä mod 1590, jolloin saadaan 297. Tämä tulkitaan jonon t avulla: Luetaan t-jonoa oikealta vasemmalle. 175 on ensimmäinen mahdollinen, vähennetään se luvusta 297, jää 122. Suurin tätä pienempi t-jonossa on 87. Vähennetään se 122:sta. Jäljellä on 35. Tällä tavoin jatkaen löydetään vielä luvut $21, 11$ ja 3. Täten

$$297 = 3 + 11 + 21 + 87 + 175$$

Muodostetaan sitten bittijono, jossa vastaavissa kohdissa on 1. Tällainen on 0101101100 eli selväkielisenä LM.

Pokeria puhelimitse.

Miten kaksi ihmistä A ja B voivat pelata pokeria puhelimitse?

A ja B pyytävät kolmannelta henkilöltä luvun n, joka on kahden suuren alkuluvun p ja q tulo. Kolmas henkilö antaa A:lle ja B:lle myös henkilökohtaiset luvut e_A ja d_A sekä e_B ja d_B, jotka toteuttavat RSA:ssa esiintyvät kongruenssit, esim.

$$e_A d_A \equiv 1 \bmod (p-1)(q-1)$$

Sitten A ja B sopivat siitä, miten pakan 52 korttia esitetään erisuurina välin $2...n$ kokonaislukuina $k_1, k_2, ..., k_{52}$., joiden syt n:n kanssa on 1. B sekoittaa oman puhelimensa ääressä korttipakan ja ilmoittaa syntyneen korttien järjestyksen A:lle siten, että luvun k_i asemesta hän ilmoittaa luvun

$$x = k_i^{e_B} \bmod p$$

A valitsee umpimähkään viisi kuulemistaan 52:sta luvusta ja ilmoittaa ne B:lle. Nämä ovat B:n kortit. A ei niitä tiedä, koska tuntee vain luvut x. Toisaalta nämä luvut korotettuna potenssiin d_B antavat $k_i \bmod n$,

joten B tietää omat korttinsa. A valitsee umpimähkään viisi muuta korttia jäljellä olevista 47 luvusta ja ilmoittaa ne B:lle, kuitenkin siten, että luvun x asemesta han ilmoittaa luvun

$$y = x^{e_A} \bmod p$$

Kuultuaan edellisen vaiheen viisi y-lukua B ilmoittaa A:lle nämä luvut korotettuna potenssiin $d_B \bmod n$. A:n korotettua saamansa viisi lukua potenssiin $d_A \bmod n$ hän tietää korttinsa, sillä B:n aukaistua oman lukkonsa A saa tietää kortit oman lukkonsa avulla.

Tämä salakirjoitusmenetelmä perustuu siihen, että laskeminen $\bmod n$ sekoittaa alkuperäisen koodin totaalisesti. Oikeastaan ei pelaamisessa välttämättä tarvita kolmannen henkilön apua ollenkaan ja moduliksi n voidaan valita tarpeeksi suuri alkuluku. Prosessissahan ei missään vaiheessa anneta vastustajan tietoon e-lukua. Täten tämän on mahdotonta saada selville myöskään d:tä.

Edellä kuvatut esimerkkien tyyppisiä vakavammin otettavia aihepiirejä löytyy esim. pankki- ja pörssimaailmasta samoin kuin vaalijärjestelyistä. On myös helppo kuvitella tilanteita, joissa ei haluta paljastaa toiselle osapuolelle esim. uuden aseen yksityiskohtia, mutta halutaan saada toinen osapuoli vakuuttuneeksi siitä, että tällainen on rakennettavissa tai että tällainen on jo hallussa.

Harjoitustehtäviä

1. Salakirjoita teksti KOKOONKOKOKOONKOKOKOKON Hillin menetelmällä. Käytä samaa matriisia kuin edellä. Lisää loppuun tarpeellinen määrä X-kirjaimia, jotta tekstin pituus on 3:lla jaollinen

2. Decryptaa teksti HOHÄNONONONMOMURORHOHAAJOJA (Teksti on suomea. Alkuperäinen on Astrid Lindgrenin kirjasta, Mestarietsivä Kalle Blomkvist.)

3. Decryptaa teksti HEISKA LAVONEN KETI VERRAN KUOLASUORMAA SIRVEN JÄVUITSE. (Teksti suomea. Salakirjoitusmenetelmä ns. siansaksa.)

4. Decryptaa teksti KOMÄNTÄNTTI KOIVONTTI KOLLÄKYNTTI KOTKAISTARANTTI. (Teksti on suomea. Salakirjoitusmenetelmä on ns. kontinkieli.)

5. Mitä tässä lukee?

   QÄJÄGQRÄ HÄ QÄJÄNCPZGQRZ

   Teksti on suomea, käytetty Caesar-salakirjoitusta ja kirjaimisto täydennetty tyypillisillä vierailla merkeillä (kuten B, C,..., W,...,Z)

6. Monisteen tekstissä kirjainten suhteellinen jakauma on

A 8.213	N 5.633	Ä 2.905	R 1.937	B 0.537
I 8.015	O 4.969	L 2.848	J 1.710	X 0.516
T 7.527	K 4.516	U 2.339	D 1.329	Z 0.474
E 6.347		M 2.212	Y 1.180	C 0.417
S 6.142		V 2.007	H 1.152	F 0.304
			P 1.145	Ö 0.241
				G 0.212
				Q 0.141
				Å 0.064
				W 0.035

Siis mitä tärkeää tämä salakirjoitus kätkee?

IBEY YFSOKWCCGYOGHOKYCF ÖHKNEGNJOBEYEGC ECFOO
HGGR BOOUYHF CXNJJC GYHUOFEYYKGO
ECCUCCF WYGHF JNOGHGGCXCBEY GCVCFEC.

Sanojen välit ovat paikoillaan ja teksti on monisteesta.

7. Selväkielinen teksti SAUNA saa kryptattuna muodon TAKE BACK VAT OR BONDS. Millainen on käytetty kryptosysteemi?

8. Ratkaise oheinen salakirjoitus 1 (lähde: Simon Singh: Koodikirja s. 472). Teksti löytyy sähköisessä muodossa oppilaitoksen levyiltä. Teksti on suomea.

9. Ratkaise oheinen salakirjoitus 2 (lähde: Simon Singh: Koodikirja s. 473). Teksti on latinaa. Ensimmäinen kirjain on F.

10. Käytetään kirjainten numerointia

A	=	01	G	=	07	M	=	13	S	=	19	Z	=	25
B	=	02	H	=	08	N	=	14	T	=	20	Å	=	26
C	=	03	I	=	09	O	=	15	U	=	21	Ä	=	27
D	=	04	J	=	10	P	=	16	V	=	22	Ö	=	28
E	=	05	K	=	11	Q	=	17	X	=	23			
F	=	06	L	=	12	R	=	18	Y	=	24			

Valitaan RSA-systeemi, jossa $n = pq = 1147$, $e = 47$ ja $d = 23$. Laske $\varphi(n)$. Tarkista e:tä ja d:tä sitova yhtälö.

11. Kryptaus kuten eo. tehtävässä Koodaa sana SANOMA. Lohko koodattava sana 3-pituisiin ja käytä salakirjoitetussa 4-pituista lohkoa, ts. 3-pituinen lohko $\rightarrow$ 4-pituinen salakirjoitettu lohko.

12. Kryptaus kuten eo. tehtävässä. Mitä seuraavassa lukee?

1031 0248 0773 0386 0517 1046 0399 0877 1141 0529 0100

13. Käytetään kirjainten numerointia

A=00000	F=00101	K=01010	P=01111	U=10100	Å=11001
B=00001	G=00110	L=01011	Q=10000	V=10101	Ä=11010
C=00010	H=00111	M=01100	R=10001	X=10110	Ö=11011
D=00011	I=01000	N=01101	S=10010	Y=10111	
E=00100	J=01001	O=01110	T=10011	Z=11000	

(5-pituinen bin.koodi.) ja selkäreppu-salakirjoitusmenetelmää avaimilla

$$\begin{aligned} t' &= (43, 129, 215, 473, 903, 302, 561, 1165, 697, 1523) \\ t &= (1, 3, 5, 11, 21, 44, 87, 175, 349, 701) \\ & (37, 1590) \qquad 43 \end{aligned}$$

- a) Salakirjoita eo. menetelmällä sana AUTO
- b) Mitä tässä lukee 2756 516 1507 ?

14. Jono $(2, 7, 10, x, 50)$ on superkasvava. Mikä on x:n ala ja yläraja?

15. Tarkastellaan RSA-salakirjoitussysteemiä, jossa $n = p \cdot q = 47 \cdot 53$. Laske $\varphi(n)$. Etsi jokin salakirjoituksessa tarvittava pari e, d. Salakirjoita bittijono 000011011100 tulkitsemalla se ensin luvuksi yhtenä lohkona.

Elliptiset käyrät

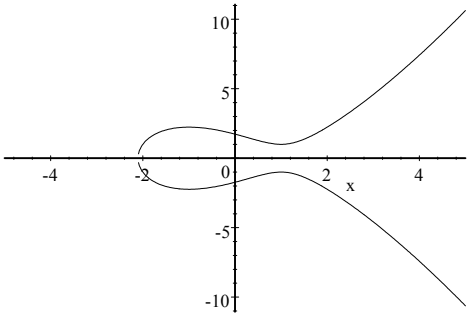
Elliptisten käyrien teoria on lähtöisin analyysistä ja lukuteoriasta. Tätä kautta niihin on myös helpointa päästä kiinni."Normaali" elliptinen käyrä on muotoa.

$$y^2 = x^3 + ax + b$$

missä a ja b ovat vakioita.Esimerkiksi käyrällä

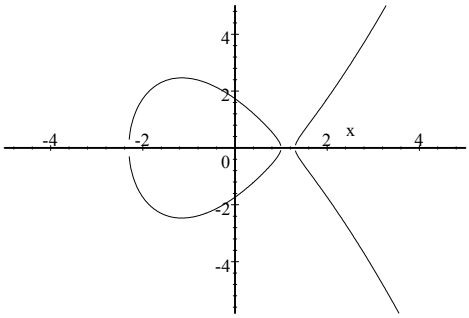
$$y^2 = x^3 - 3x + 3$$

on näille käyrille tyypillinen muoto

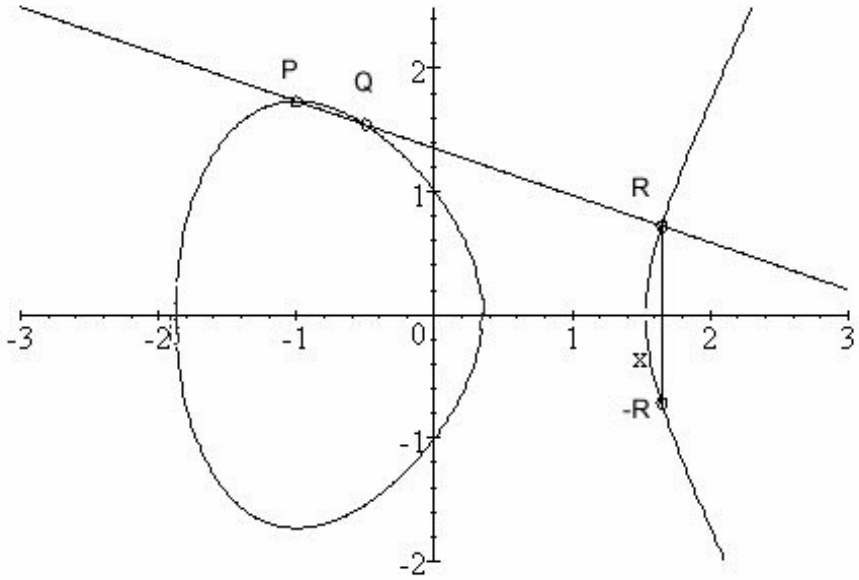


Toinen tyypillinen muoto on

$$y^2 = x^3 - 4x + 3$$



Olkoon P ja Q kaksi elliptisen käyrän $\mathcal{E}$ pistettä. Voidaan määritellä yhteenlasku $P+Q$ seuraavasti: Otetaan suora, joka kulkee pisteiden P ja Q kautta Olkoon R kolmas käyrän $\mathcal{E}$ ja suoran leikkauspiste.Tällöin määritellään $P+Q = -R$, missä $-R$ on R:n peilikuva x-akselin suhteen (eli. $-R$:llä on sama x-koordinaatti mutta y-koordinaatin merkki on muuttunut).



Määritellään $2P = P + P = -R$ ja lasketaan $-R$ kuten edellä, mutta suora muuttuu tangentiksi pisteessä P. Edelleen monikerrat nP lasketaan rekursiivisesti $nP = (n-1)\,P + P$. Määritellään vielä $-P$ pisteenä, jolla on sama x-koordinaatti kuin P:llä, mutta y-koordinaatin merkki on vaihdettu. Tämän systeemin nolla-alkio O on ääretönpiste. Nyt $\mathcal{E}$ muodostaa ns. Abelin ryhmän, joka on suljettu yhteenlaskun suhteen ja jossa kaikilla on vasta-alkio.

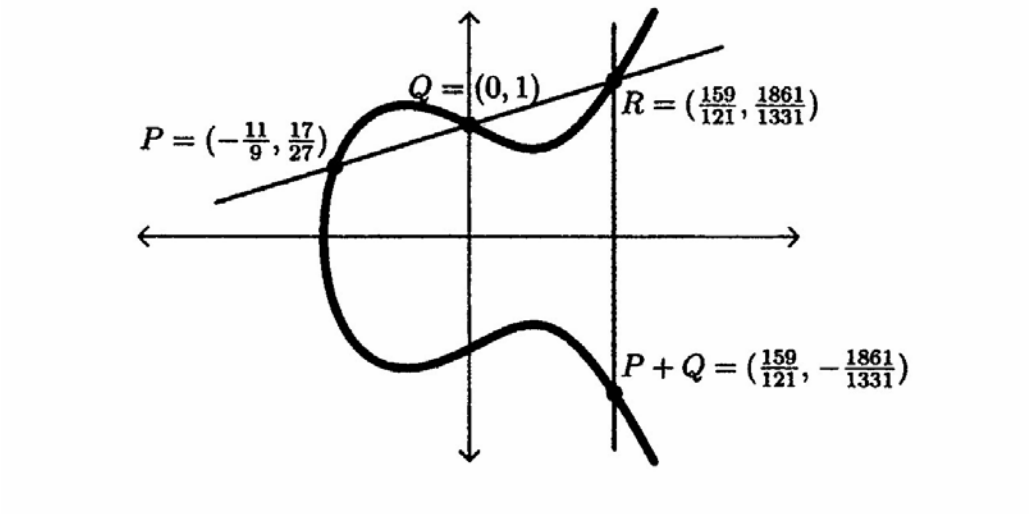
Esimerkki 34 *Kun*

$$y^2 = x^3 - x + 1$$

niin

$$\left(-\frac{11}{9}, \frac{17}{27}\right) + (0,1) = \left(\frac{159}{121}, -\frac{1861}{1331}\right)$$

Vertaa kuvaa:



Elliptisten käyrien kryptografia perustuu siihen, että vaikka tunnetaan P ja nP, on äärimmäisen vaikea ratkaista kokonaisluku n. Tätä probleemaa käytetään sellaisessa muodossa, jossa eo. käyrien analogioita tarkastellaan binäärikunnissa.

Diffie-Helmanin avaintenvaihtoprotokollan analogia. Oletetaan, että Ada ja Barry haluavat sopia avaimesta, jota myöhemmin käytetään klassillisen kryptosysteemin yhteydessä. Valitaan ensin julkisesti äärellinen kunta F_q ja siinä määritelty elliptinen käyrä $\mathcal{E}$. Mikäli Ada ja Barry pystyvät nyt määrittämään sellaisen pisteen P tältä käyrältä, jota muut eivät tiedä, voidaan klassilisen kryptosysteemin avaimeksi valita esimerkiksi tämän pisteen x-koordinaatti..

Ada and Barry valitsevat ensin julkisesti pisteen B käyrältä $\mathcal{E}$ palvelemaan eräänlaisena kantapisteenä. Avaimen generointia varten Ada valitsee satunnaisen kokonaisluvun a (q:n kertaluokkaa). Tämän hän pitää salaisena. Hän laskee pisteen aB käyrältä $\mathcal{E}$ ja julkistaa tuloksen. Barry tekee samoin: Valitsee satunnaisluvun b ja julkistaa tuloksen bB. Salainen avain on $P = abB$. He molemmat pystyvät laskemaan tämän, mutta muut eivät. Esimerkiksi Ada tuntee pisteen bB (julkinen) ja salaisen lukunsa a. Kolmas osapuoli kuitenkin tuntee vain pisteet aB ja bB. Selvittääkseen pisteen $P = abB$ on näiden ratkaista ns. diskreetin logaritmin probleema - löydettävä a tuntien B ja aB (sekä löydettävä b tuntien B ja bB) - Tällä hetkellä ei ole tiedossa nopeaa metodia tähän.

Massey-Omuran metodi (viestin salaamiseen). Oletetaan, että molemmilla (Ada ja Barry) on henkilökohtaiset salaus- ja purkuavaimet e_A, e_B (salaus) d_A, d_B (purku). Muutetaan nyt salattava viesti m käyrän $\mathcal{E}$ pisteeksi P_m. Ada lähettää viestin muodossa $e_A P_m$. Barry salaa tämän

omalla salauksellaan muotoon $e_B e_A P_m$ ja palauttaa Adalle. Tämän jälkeen Ada purkaa tästä oman salauksensa avaimellaand_A ja lähettää taas Barrylle. Viesti on nyt $e_B P_m$, tämän Barry pystyy omalla avaimellaan purkamaan. Tämä tietysti edellyttää salauksilta sitä, että järjestys voidaan vaihtaa, ts. $e_B e_A P_m = e_A e_B P_m$. Tämä ei tietenkään ole itsestään selvää, mutta onnistuu esimerkiksi elliptisiin käyriin perustuvilla salauksilla.

ElGamalin metodi (viestin salaamiseen). Valitaan julkinen B.Jokainen valitsee satunnaisen kokonaisluvun a, jonka salaa ja laskee sekä julkaisee pisteen aB. Viestin P_m lähettämiseksi Barrylle Ada valitsee satunnaisluvun k ja lähettää parin $(kB, P_m + k(a_B B))$ (missä $a_B B$ on Barryn julkinen avain). Lukeakseen viestin Barry kertoo ensimmäisen pisteen tässä parissa salaisella luvullaan a_B ja vähentää sen jälkiosasta:

$$P_m + k(a_B B) - a_B(kB) = P_m.$$

Harjoitustehtäviä

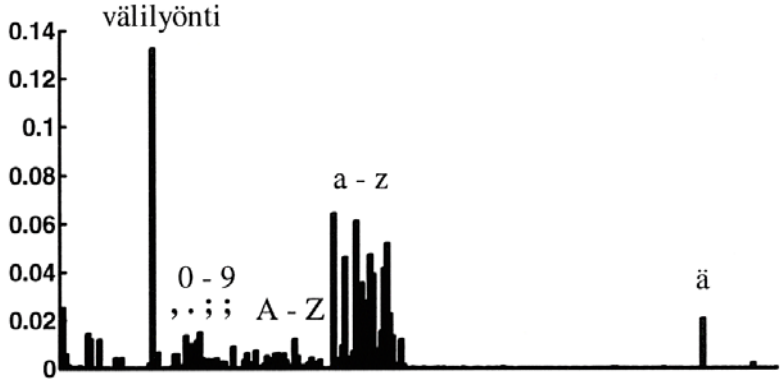
1. Olkoon elliptinen käyrä $\mathcal{E}$: $y^2 = x^3 - 36x$ ja $P = (-3, 9)$ sekä $Q = (-2, 8)$. Määritä $-P$, $-Q$, $P + Q$ ja $2P$. (Vastaus: $-P = (-3, -9)$, $-Q = (-2, -8)$, $P + Q = (6, 0)$, $2P = \left(\frac{25}{4}, -\frac{35}{8}\right)$.)

Luku 4

Lähdekoodaus

Entropia

Tietokoneissa yleisesti käytössä oleva aakkosto on 8:n bitin ASCII-aakkosto (American Standard Code for Information Interchange). Tässä aakkostossa on $2^8 = 256$ merkkiä, joista kukin on 8 bittiä pitkä eli yhden tavun mittainen. Oheisen monisteen eräässä aikaisemmassa versiossa oli 30000 merkkiä. Graafisesti jakauma oli seuraavanlainen:



Kun eri merkkien määrät lasketaan, saadaan oheinen jakauma. Välilyönti on ehdottomasti yleisin merkki, seuraavalla sijalla on 'a' ja sen jälkeen muut pienet kirjaimet. Isot kirjaimet, numerot ja välimerkit ovat huomattavasti harvemmin esiintyviä. Kaikkia ASCII-aakkoston merkkejä ei edes löydy, arviolta puolet ovat käyttämättömiä tai hyvin harvoin esiintyviä.

Merkkien erilainen esiintymistiheys mahdollistaa tiedon tiivistämisen. Kiinteän pituisen eli mainitun 8-bittisen ASCII-koodin sijasta voidaan

käyttää vaihtelevan pituista koodia, missä yleisimpien merkkien koodisanat ovat lyhyitä ja harvinaisten merkkien koodisanat taas pitkiä. Kaavion havaintoaineisto voitaisiin esimerkiksi koodata binääriseksi siten, että välilyönti olisi ehkä 1-2 bittiä, 'a', 'e', 'i' 2-3 bittiä, 'A' - 'Z' 4-7 bittiä ja käyttämättömät sekä harvinaiset merkit vaikka 16-20 bittiä. Esimerkiksi Morse-aakkosto on vaihtelevan pituinen koodi, jossa on huomioitu englantilaisen aakkoston kirjainten esiintymistiheydet.

Informaatioteorian mitta vaihtelevan pituisessa koodissa tarvittavien bittien keskiarvolle on *entropia*. Ajatellaan ensin tilannetta, missä N yhtä todennäköistä kirjainta on esitettävä binäärimuodossa, kukin kirjain yhtä monen bitin avulla. Koska kirjainten koodisanojen on oltava erit ja koska erilaisia k:n pituisia bittijonoja on 2^k kappaletta, näiden N:n kirjaimen esitykseen tarvittavien bittien määrän k (kirjainta kohti) on täytettävä ehto $2^k = N$ eli $k = \log_2 N$. Luku $\log_2 N$ tulkitaan informaatioteoriassa asiayhteyden mukaan yhden kirjaimen sisältämäksi informaatioksi tai siksi määräksi informaatiota (tässä bittejä), mikä tarvitaan kirjaimen identifioimiseksi. Kun kirjaimen informaatio kirjoitetaan muodossa

$$\log_2 N = \log_2 (1/(1/N))$$

havaitaan sen riippuvan kirjaimen todennäköisyydestä $1/N$ (kaikki N kirjainta oletettiin yhtä todennäköisiksi). Tämän yleistys on selvä: jos kirjaimen todennäköisyys on p_i, sen informaatio tai sen identifioimiseen tarvittavan informaation määrä (bittien määrä) on $\log_2 (1/p_i)$. Tämä merkitsee, että kirjaimen koodisanan pituus on logaritmisesti kääntäen verrannollinen sen todennäköisyyteen. Näin päästään tavoitteseen: yleisimpien kirjaimien koodisanat ovat lyhyitä, harvinaisten pitkiä. Entropia on näiden pituuksien odotusarvo eli kirjainten todennäköisyyksillä painotettu keskiarvo.

Määritelmä 35 *Olkoon lähdeaakkoston $\mathcal{L} = \{l_1, l_2, \ldots, l_n\}$ kirjainten jakauma (eli kirjainten esiintymistodennäköisyydet)*

$$p_1 = P(l_1), p_2 = P(l_2), \ldots, p_n = P(l_n)$$

Tällöin aakkoston (lähteen) $\mathcal{L}$ entropia $H = H(\mathcal{L})$ on

$$\begin{aligned} H &= \sum_{i=1}^{n} p_i \log \left(\frac{1}{p_i}\right) = -\sum_{i=1}^{n} p_i \log p_i \\ &= -p_1 \log p_1 - p_2 \log p_2 - \cdots - p_n \log p_n \end{aligned}$$

(Summassa oleva termi $p_i \log p_i$ määritellään 0:ksi silloin, kun $p_i = 0$.)

Entropia on laadullinen suure, yksikkönä siinä käytetyn logaritmin kantaluku. Kun tavoitteena on tiedon koodaus binäärimuotoon, logaritmin kantalukuna on 2 ja informaation sekä entropian yksikkönä

on bitti. Jonkin toisen ongelman yhteydessä tieto halutaan ehkä esittää desimaalijärjestelmän lukuina ja tälllöin logaritmin kantalukuna on tietenkin 10. Yleisesti, kun lähdeaakkosto $\mathcal{L}$ halutaan koodata koodiaakkostoon $\mathcal{K}$, missä on k kirjainta, luonnollinen valinta logaritmin kantalukuvuksi on k. Jatkossa oletammekin aina, että logaritmin kantaluku on koodiaakkoston kirjainten määrä k. Tarvittaessa tämä voidaan merkitä näkyviin alaindeksin avulla. Logaritmien muunnoskaava antaa muunnoskaavan

$$H_k(\mathcal{L}) = H_2(L)/\log_2 k$$

Edellisen havaintoaineiston entropia on 5.4 bittiä eli pienempi kuin käytetyn ASCII-koodin pituus $8 = \log_2 256 = \log_2 n$. Seuraava lause osoittaa, että tämä on yleisesti voimassa paitsi, jos lähdeaakkoston kirjaimet ovat yhtä todennäköisiä, missä tapauksessa entropia on suurimmillaan eli $H = \log n$, missä n on lähdeaakkoston kirjainten määrä.

Lause 36 *Jos aakkostossa $\mathcal{L} = \{l_1, l_2, \ldots, l_n\}$ on n kirjainta, niin*

$$0 \leq H(\mathcal{L}) \leq \log_2 n$$

Lisäksi $H(\mathcal{L}) = \log n$ tarkalleen silloin, kun aakkoston $\mathcal{L}$ kirjaimet ovat yhtä todennäköiset.

Todistus. Kirjainten l_i todennäköisyydet $p_i = P(l_i)$ ovat välillä $0 \leq p_i \leq 1$, joten niiden logaritmit ovat ei-positiivisia eli $\log p_i \leq 0$ ja siis $-p_i \log p_i \geq 0$. Koska entropia H on näiden termien $-p_i \log p_i$ summa, niin $H \geq 0$.

Toisen epäyhtälön todistamiseksi tarkastellaan erotusta $H(\mathcal{L}) - \log n$ ja osoitetaan se ei positiiviseksi käyttäen hyväksi logaritmin ominaisuuksia ja differentiaalilaskentaa. Tämä sivutetaan.

Lähdekoodausteoreema

Yksi informaatioteorian kulmakivistä on lähdekoodausteoreema. jonka mukaan aakkoston kirjaimet voidaan koodata koodisanoiksi, joiden pituuksien keskiarvo on halutun lähellä koodattavan lähdeaakkoston entropiaa. Seuraavassa on tämän tuloksen ensimmäinen approksimaatio.

Lause 37 *(Lähdekoodausteoreema). Olkoon $\mathcal{L} = \{l_1, l_2, \ldots, l_m\}$ lähdeaakkosto, $\mathcal{K} = \{k_1, k_2, \ldots, k_r\}$ koodiaakkosto sekä $f : L^* \to K^*$ koodaus. Olkoot edelleen $p_1 = P(l_1), p_2 = P(l_2), \ldots, p_m = P(l_m)$ kirjainten todennäköisyydet, $n_1, n_2, \ldots, n_m$ koodisanojen $c_1 = f(l_1), \ldots, c_m = f(l_m)$ pituudet sekä $\overline{n}$ näiden pituuksien keskiarvo eli*

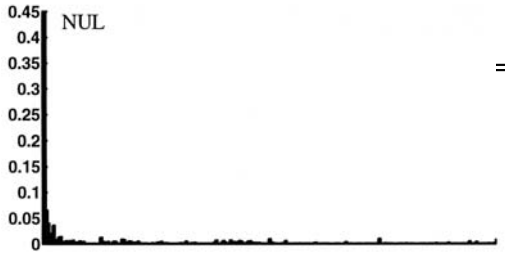
$$\overline{n} = \sum_{n=1}^{m} p_i n_i = p_1 n_1 + p_2 n_2 + \cdots + p_m n_m$$

Tällöin $\overline{n} \geq H_r(\mathcal{L})$ Kääntäen on olemassa koodi, jolle $\overline{n} < H_r(\mathcal{L}) + 1$.

Havaintoaineistossa sivulla 44 entropia on $H = 5.4$ bittiä, mikä merkitsee, että 8-bittisen ASCII-aakkoston sijasta teksti voitaisiin koodata vaihtelevan pituisella koodilla, missä koodisanojen keskimääräinen pituus olisi 5.4 bittiä. Vieressä on muutama esimerkki yksittäisten kirjainten määristä, niiden todennäköisyyksistä p_i sekä entropian ehdottamista koodisanojen pituuksista $\log_2(1/p_i)$.

merkki	**määrä**	p_i	$\log_2(1/p_i)$
välilyönti	3977	0,132	2,92
a	1931	0,064	3,92
A	184	0,006	7,42
DEL	1	0,00003	14,89

Kun mukaan otetaan tiedoston grafiikka, merkkijakautuma on täysin erilainen. Lähes puolet eräästä tähän monisteeseen liittyvästä 410 Ktavun kokoisesta tiedostosta koostuu ASCII-merkistä NUL ja kaikki ASCII-merkit löytyvät tiedostosta. Tämä heijastuu välittömästi entropiaan. Entropia on bittiä pienempi: $H = 4.415883$. Tämä merkitsee, että tiedosto voidaan koodata binäärimuotoon, missä on keskimäärin 4.4 bittiä yhtä merkkiä kohden. Kun entropiaa verrataan käytetyn ASCII-koodin pituuteen 8 bittiä, havaitaan tiedoston talletukseen tarvittavan ainoastaan 55 % ASCII-koodin vaatimasta tilasta: 410 Ktavuinen tiedosto olisi voitu kutistaa 45 % pienemmäksi eli 226 Ktavuiseksi.



merkki	p_i	$\log_2(1/p_i)$
NUL	0,4455	1,17
välilyönti	0,0120	6,38
a	0,0064	7,30
A	0,0007	1040

Kun merkkien (tavujen) sijasta koodataankin merkkipareja (sanoja), päästään yleensä pienempään entropiaan eli parempaan pakkaussuhteeseen, ja koodattaessa vielä pitempiä merkkijonoja pakkaussuhde paranee yhä edelleen. Alla olevassa taulukossa on esimerkki kirjainparien entropiasta sekä eräiden vanhempien pakkausohjelmien suorituksista.

Tiedoston koko: 699.020 tavua (682.6 Ktavua)

Entropia	**antaisi tiedostokoon**	**% alkuper.**
/ tavu: 4.205 bittiä / tavu	367.964 (358.9 Kb)	52,6
/ sana: 7.137 bittiä / sana = 3.569 bittiä / tavu	311.823 (304.5 Kb)	44,6

Pakkausohj/ver	**pakattu tiedosto**	**% alkup**	**vast entr.**
Zoo 2.01	208.031 (203.8 Kb)	29,8	2,380
PkArc 3.5	207.669 (202.8 Kb)	29,7	2,377
Diet 1.10a	146.304 (142.9 Kb)	20,9	1,674
PkZip 1.1	142.763 (139.4 Kb)	20,4	1,634
Lha 2.13	120.923 (118.1 Kb)	17,3	1,374

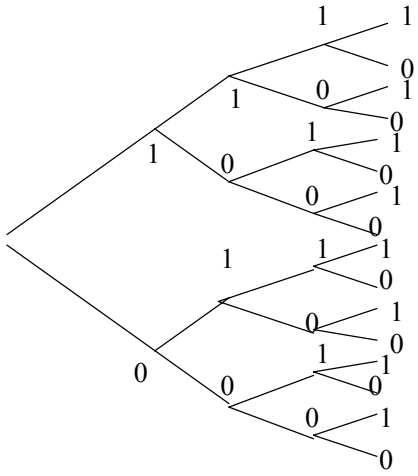
Etuliitekoodit

Lähdekoodausteoreema on tyypillinen kvalitatiivinen olemassaolotulos, joka kertoo ainoastaan keskiarvomielessä parhaan koodin koodisanojen pituuksien odotusarvon olevan lähellä lähdeaakkoston entropiaa, mutta ei kerro, miten tällainen koodi voidaan konstruoida muuten kuin satunnaisilla valinnoilla. Huomattakoon myös, että entropiassa olevat logaritmit $log_2(1/p_i)$ eivät välttämättä ole kokonaislukuja eivätkä siis kelvollisia koodisanojen pituuksiksi, vaan vähimmäistoimenpiteenä on suoritettava niiden pyöristys. Tämä kokonaislukuvaatimus on huomioitu myöhemmin esitettävässä *Kraftin epäyhtälössä*, mikä on lähdekoodausteoreeman todistuksen ydinkohta.

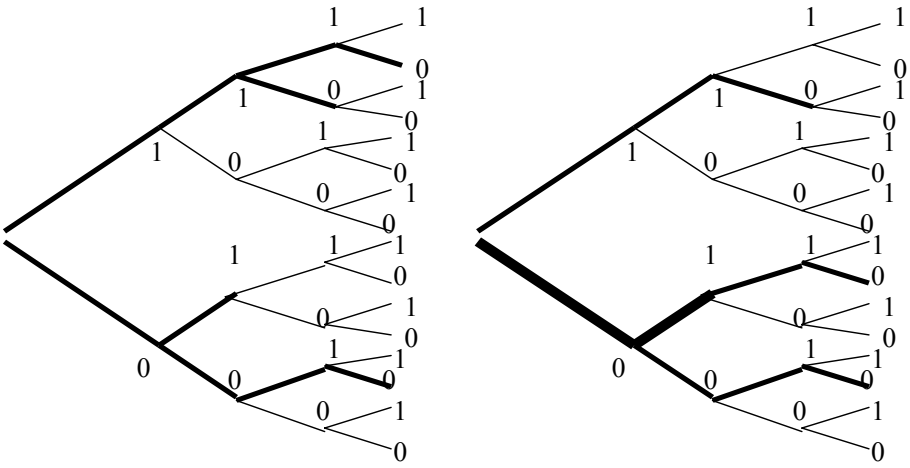
Etuliitekoodi on koodi $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$, missä yksikään koodisana c_i ei ole minkään toisen koodisanan $c_j (j \neq i)$ etuliite eli c_j ei ole muotoa $c_i a$, missä a on jokin ei-tyhjä sana. Esimerkiksi koodi $\mathcal{C}_1 = \{01, 110, 0010, 0110\}$ ei ole etuliitekoodi, sillä sen koodisana $c_1 = 01$ on koodisanan $c_4 = 0110$ etuliite: $c_4 = 0110 = c_1 10$. Koodi $\mathcal{C}_2 = \{01, 110, 0010, 1110\}$ sen sijaan on etuliitekoodi.

Etuliitekoodi sekä muutkin koodit voidaan esittää graafisesti (binääri) puuna: puun juuresta haarautuu kaksi oksaa 0 ja 1, kummankin oksan toisesta päästä haarautuu taas kaksi oksaa 0 ja 1 jne. Haarautumiskohtia nimitetään solmuiksi. Solmut, joihin päästään juuresta kulkemalla yhtä haaraa pitkin, ovat tasolla 1, solmut, joihin päästään juuresta kulkemalla kahta oksanhaaraa, ovat tasolla 2, ja niin edelleen. Ylimmän tason solmut, joista ei enää oksia haaraudu, ovat puun lehtiä. Kukin koodisana on graafisessa esityksessä koodisanan bittien määräämä oksistoreitti juuresta eteenpäin. Esimerkiksi 011001 määrää reitin 'juuri $\rightarrow 0 \rightarrow 1 \rightarrow 1 \rightarrow 0 \rightarrow 0 \rightarrow 1$'. Tässä kuljetaan kuutta oksanhaaraa pitkin, joten sanan viimeinen bitti (1) on puussa tasolla 6 (= sanan 011001 pituus). Koska i:n pituisia bittijonoja on 2^i kappaletta, tasolla

i on 2^i solmua eli yhtä paljon kuin i:n pituisia sanojakin.



Esimerkki 38 *Koodien* $\mathcal{C}_1 = \{01, 110, 0010, 0110\}$ *ja* $\mathcal{C}_2 = \{01, 110, 0010, 1110\}$ *puuesitykset ovat seuraavanlaiset (käytetyt haarat vahvistettuina):*



Lause 39 *(Kraftin epäyhtälö). Mikäli koodiaakkoston* $\mathcal{K}$ *kirjainten määrä on* k *sekä etuliitekoodin* $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ *koodisanojen pituudet ovat* $n_1, n_2, \ldots, n_m$*, niin*

$$\sum_{i=1}^{m} \frac{1}{2^{n_i}} = 2^{-n_1} + 2^{-n_2} + \cdots + 2^{-n_m} \leq 1 \tag{4.1}$$

Kääntäen, mikäli positiiviset kokonaisluvut $n_1, n_2, \ldots, n_m$ *toteuttavat tämän epäyhtälön, niin on olemassa etuliitekoodi* $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$*, jonka koodisanojen pituudet ovat* n_1*,*n_2*,...,*n_m*.*

Tulos on varsin ilmeinen, kun katselee eo binääripuita.

Esimerkki 40 $\{a = 01, b = 110, c = 0010, d = 1110\}$

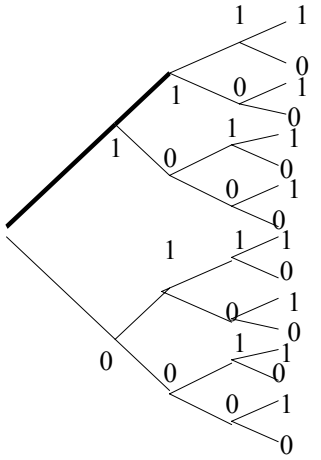
$$\frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} + \frac{1}{2^4} = \frac{1}{2}$$

Etuliitekoodi annetuin pituuksin konstruoidaan valitsemalla täydestä binääripuusta solmuja koodisanoiksi seuraavasti: Valitaan ensimmäiseksi koodisanaksi c_1 satunnaisesti tasolla n_1 oleva solmu (eli polku juuresta ko. solmuun). Tasolla n_2 olevista solmuista on kappaletta solmun c_1 alipuussa., joten tasolla n_2 on solmuja, jotka eivät ole solmun c_1 alipuussa. Valitaan satunnaisesti jokin niistä toiseksi koodisanaksi c_2. Jatketaan näin induktiivisesti.

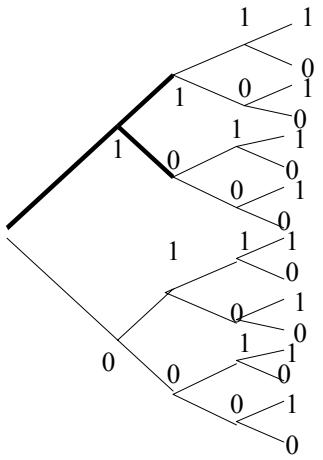
Esimerkki 41 *Konstruoidaan seuraavaksi koodi* $\{a, b, c, d, e\}$*, jonka koodipituudet ovat 2, 2, 3, 3, 4. Koska*

$$\frac{1}{2^2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^3} + \frac{1}{2^4} = \frac{13}{16} < 1$$

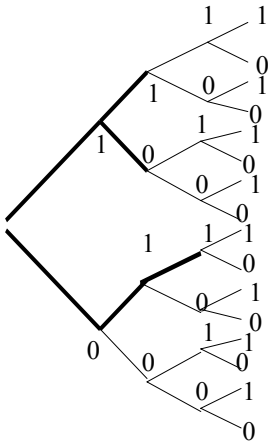
tämän pitäisi olla mahdollista



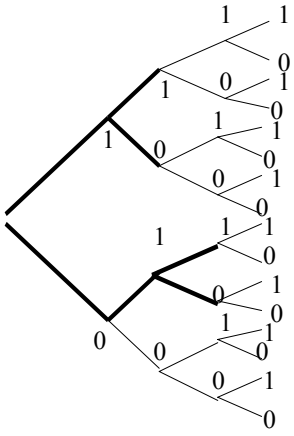
$a = 11$



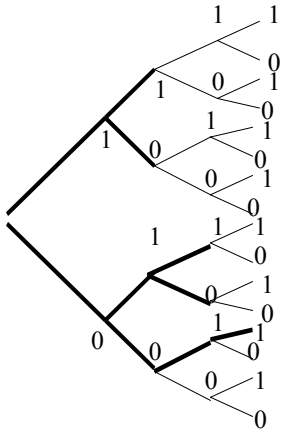
$b = 10$



$c = 011$



$d = 010$



$e = 0011$

Kraftin epäyhtälö todistettiin ainoastaan etuliitekoodeille, mutta muut koodit eivät ole sen parempia, vaan Kraftin epäyhtälö on voimassa kaikille koodeille.

Meillä on nyt menetelmä konstruoida koodisanat, kun niiden pituudet tiedetään. Ongelman lähtökohtana on kuitenkin lähdeaakkosto $\mathcal{L} = \{l_1, l_2, \ldots, l_m\}$, jolle on etsittävä sopiva koodaus. Mikäli kirjainten todennäköisyydet ovat $p_1 = P(l_1), p_2 = P(l_2), \ldots, p_m = P(l_m)$, valitaan kokonaisluvut $n_1, n_2, \ldots, n_m$ siten, että

$$\frac{1}{2^{n_i}} \leq p_i < \frac{1}{2^{n_i - 1}} \tag{4.2}$$

Kun lasketaan vasemman puoleiset epäyhtälöt puolittain yhteen, saadaan

$$\frac{1}{2^{n_1}} + \frac{1}{2^{n_2}} + \cdots + \frac{1}{2^{n_m}} \leq p_1 + p_2 + \cdots + p_m = 1$$

Näin ollen Kraftin epäyhtälö on voimassa ja voidaan konstruoida koodi $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$, jonka koodisanojen pituudet ovat juuri valitut luvut $n_1, n_2, \ldots, n_m$ Kun epäyhtälöiden (4.2) oikean puoleisista epäyhtälöistä otetaan puolittain 2-kantainen logaritmi, saadaan

$$\log p_i < -n_i + 1 \quad (i = 1, \ldots, m)$$

Kun vielä kerrotaan tässä i:s epäyhtälö puolittain p_i:llä ja lasketaan saadut epäyhtälöt yhteen, tuloksena on

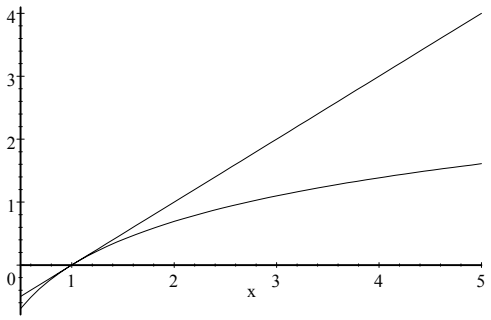
$$\begin{aligned} & p_1 \log p_1 + p_2 \log p_2 + \cdots + p_m \log p_m \\ < \; & p_1 (-n_1 + 1) + p_2 (-n_2 + 1) + \cdots + p_m (-n_m + 1) \\ = \; & -(p_1 n_1 + p_2 n_2 + \cdots + p_m n_m) + (p_1 + p_2 + \cdots + p_m) \end{aligned}$$

eli

$$\text{entropia} > \text{keskimääräinen sana pituus} - 1$$

Näin on todistettu lähdekoodausteoreeman koodin olemassaoloa koskeva osa. Ensimmäinen osa nähdään käyttäen epäyhtälöä

$$\ln x \leq x - 1$$



ja Kraftin lausetta. Nimittäin

$$\begin{aligned}
H - L(C) &= \frac{1}{\ln 2}\left(-p_1 \log p_1 - p_2 \log p_2 - \cdots - p_m \log p_m\right) \\
&\quad - \left(p_1 n_1 + p_2 n_2 + \cdots + p_m n_m\right) \\
&= \left(-p_1 \log p_1 - p_2 \log p_2 - \cdots - p_m \log p_m\right) \\
&\quad + \left(p_1 \log 2^{-n_1} + p_2 \log 2^{-n_2} + \cdots + p_m \log 2^{-n_m}\right) \\
&= p_1 \log \frac{2^{-n_1}}{p_1} + p_2 \log \frac{2^{-n_2}}{p_2} + \cdots + p_m \log \frac{2^{-n_m}}{p_m} \\
&= \frac{1}{\ln 2}\left(p_1 \ln \frac{2^{-n_1}}{p_1} + p_2 \ln \frac{2^{-n_2}}{p_2} + \cdots + p_m \ln \frac{2^{-n_m}}{p_m}\right) \\
&\leq \frac{1}{\ln 2}\left(p_1\left(\frac{2^{-n_1}}{p_1} - 1\right) + \cdots + p_m\left(\frac{2^{-n_m}}{p_m} - 1\right)\right) \\
&= \frac{1}{\ln 2}\left(\left(2^{-n_1} + 2^{-n_2} + \cdots + 2^{-n_m}\right) - \left(p_1 + p_2 + \cdots + p_m\right)\right) \\
&\leq 0
\end{aligned}$$

Esimerkki 42 *Kun aakkoston* $\mathcal{L} = \{a, i, t, e, s\}$ *kirjainten todennäköisyydet ovat*

$$p_a = 0,35, \quad p_i = 0,3, \quad p_t = 0,15, \quad p_e = 0,13, \quad p_s = 0,07$$

niin entropia on

$$\begin{aligned}
H &= -\left(0.35 \ln 0.35 + 0.3 \ln 0.3 + 0.15 \ln 0.15 + 0.13 \ln 0.13 + 0.07 \ln 0.07\right) / \ln \\
&\approx 2.1129
\end{aligned}$$

ja lähdekoodausteoreema lupaa koodin, jonka keskipituus on välillä $2.1129 \leq \overline{n} < 3.1129$. *Todistusta seuraten suoritetaan arvioinnit*

$$2^{-2} = 1/4 = 0.25 < 0.35 = p_a < 0.5 = 1/2 = 2^{-2+1}$$

$$2^{-2} = 1/4 = 0.25 < 0.3 = p_i < 0.5 = 1/2 = 2^{-2+1}$$

$$2^{-3} = 1/8 = 0.125 < 0.15 = p_t < 0.25 = 1/4 = 2^{-3+1}$$

$$2^{-3} = 1/8 = 0.125 < 0.13 = p_e < 0.25 = 1/4 = 2^{-3+1}$$

$$2^{-4} = 1/16 = 0.0625 < 0.07 = p_s < 0.125 = 1/8 = 2^{-4+1}$$

ja valitaan koodisanojen pituuksiksi $n_a = 2, n_i = 2, n_t = 3, n_e = 3, n_s = 4$. *Kirjaimia vastaaviksi koodisanoiksi* c_a, c_i, c_t, c_e, c_s *voidaan nyt valita vaikka esimerkin 41 koodisanat:*

$$a \to 11 = c_a, i \to 10 = c_i, t \to 011 = c_t, e \to 010 = c_e, s \to 0011 = c_s \tag{4.3}$$

Tämän koodin keskipituus on

$$\begin{aligned}
\overline{n} &= p_a n_a + p_i n_i + p_t n_t + p_e n_e + p_s n_s \\
&= 0,35 \cdot 2 + 0,3 \cdot 2 + 0,15 \cdot 3 + 0,13 \cdot 3 + 0,07 \cdot 4 = 2,42
\end{aligned}$$

Saatu koodi ei ole paras mahdollinen. Esimerkiksi koodissa

$$a \to 11 = c_a, i \to 10 = c_i, t \to 01 = c_t, e \to 000 = c_e, s \to 001 = c_s \tag{4.4}$$

kirjainten t ja s koodisanat ovat lyhyempiä ja koodin keskipituuskin on siis pienempi:

$$\overline{n} = 0,35 \cdot 2 + 0,3 \cdot 2 + 0,15 \cdot 2 + 0,13 \cdot 3 + 0,07 \cdot 3 = 2,2$$

Vaihtelevan pituisten koodien konstruointiin on olemassa algoritmi, joka antaa koodin, minkä keskipituus on pienin mahdollinen. Tässä tapauksessa algoritmi antaa koodin

$$a \to 1 = c_a, i \to 01 = c_i, t \to 000 = c_t, e \to 0010 = c_e, s \to 0011 = c_s \tag{4.5}$$

Edellisiin verrattuna tässä koodissa on yleisimmän kirjaimen a koodisana vain yhden pituinen, mikä on saatu mahdolliseksi kasvattamalla harvinaisempien kirjainten pituuksia. Keskipituudeksi saadaan

$$\overline{n} = 0,35 \cdot 1 + 0,3 \cdot 2 + 0,15 \cdot 3 + 0,13 \cdot 4 + 0,07 \cdot 4 = 2,2$$

Kun koodilla (4.4) on sama keskipituus, havaitaan, ettei optimaalinen koodi ole yksikäsitteinen.

Sanan 'asetaatti' koodaus osoittaa eron kiinteän pituisen, 8-bittisen ASCII-koodin sekä vaihtelevan pituisten koodien (4.3), (4.4) ja (4.5) välillä:

ASCII : 01100001011100110110010101110100011000010110000101110100000111010001101001

koodi (4.3) : 110011010011111101101101

koodi (4.4) : 11001000011111010110

koodi (4.5) : 1001100100001100000001

Tässä koodi (4.4) oli tehokkaampi kuin koodi (4.5), mutta koodattaessa teksti 'saita taitaa asiat', tilanne onkin yhden bitin verran suosiollisempi koodille (4.5). Vertaa

koodi (4.4) : 00111100111 011110011111 11001101101

koodi (4.5) : 00111010001 00010100011 10011011000

Huffman-koodit

Lähdekoodausteoreeman ja Kraftin epäyhtälön todistusten mukainen konstruointi antaa kyllä koodin, missä koodisana on sitä lyhyempi mitä todennäköisempi se on (eli mitä todennäköisempi vastaava lähdeaakkoston kirjain on), mutta se ei välttämättä anna optimaalista koodia, jonka

keskipituus on pienin mahdollinen. Optimaalinen koodi voidaan konstruoida seuraavaksi esitettävällä *Huffmanin algoritmilla*. Huomautettakoon, että annetulle lähdeaakkostolle voi olla useampia optimaalisia koodauksia.

Tarkastellaan ensin keskipituuden muutosta, kun kahden lähdeaakkoston kirjaimen a ja b koodisanat c ja c' vaihdetaan keskenään. Kun kirjainten a, b todennäköisyydet ovat p_a, p_b, niin koodisanojen c ja c' pituuksien n, n' osuudet keskipituuksissa ja ovat

$$\begin{array}{llll} \text{koodi} & a \to c & b \to c' & \overline{n} = p_a n + p_b n' \\ \text{vaihto} & a \to c' & b \to c & \overline{n}' = p_a n' + p_b n \end{array}$$

Vaihdon aiheuttama muutos on siis

$$\overline{n}' - \overline{n} = (p_a n' + p_b n) - (p_a n + p_b n') = (p_a - p_b)\,(n' - n)$$

Mikäli alkujaan on $p_a \leq p_b$ ja $n \leq n'$, niin muutos on ei-positiivinen,$n' - n \leq 0$, ja siis koodisanat voidaan vaihtaa keskenään eikä keskipituus ainakaan kasva. Näin ollen optimaalisessa koodissa koodisanojen pituudet ovat päinvastaisessa järjestyksessä kuin niiden todennäköisyydet eli sopivasti indeksoituna tilanne on seuraavanlainen:

$$\begin{array}{ll} \text{koodisanat} & c_1, c_2, \ldots, c_m \\ \text{todennäköisyydet} & p_1 \geq p_2 \geq \cdots \geq p_m \\ \text{pituudet} & n_1 \leq n_2 \leq \cdots \leq n_m \end{array}$$

Tarkastellaan seuraavaksi pisintä koodisanaa c_m Havaitaan, että jokin koodisana c_i on yhtä pitkä kuin c_m ja eroaa tästä ainoastaan viimeisessä bitissä, sillä muussa tapauksessa koodisanasta cm voidaan jättää viimeinen bitti pois ja koodi säilyy vielä etuliitekoodina sekä keskipituus saadaan pienemmäksi. Tarvittaessa voidaan vaihtaa koodisanojen järjestystä siten, että kyseinen koodisana c_i on toiseksi epätodennäköisin koodisana c_{m-1}. Näin ollen on olemassa optimaalinen koodi, jonka kaksi epätodennäköisintä koodisanaa c_{m-1} ja c_m ovat yhtä pitkät sekä eroavat vain viimeisessä bitissä.

Optimaalinen etuliitekoodi $\mathcal{C} = \{c_1, c_2, \ldots, c_{m-2}, c_{m-1}, c_m\}$ voidaan nyt konstruoida konstruoimalla ensin optimaalinen etuliitekoodi $\mathcal{C}' = \{c_1, c_2, \ldots, c_{m-2}, c\}$, jossa on yksi koodisana vähemmän ja sanan c todennäköisyys on $p_{m-1} + p_m$, sekä valitsemalla sen jälkeen $c_{m-1} = c0$ ja $c_m = c1$. Jos nimittäin sanan c pituus on n, niin sanojen $c_{m-1} = c0$ ja $c_m = c1$ pituus on $n+1$ ja koodin $\mathcal{C}$ optimaalisuus seuraa koodien $\mathcal{C}$ ja $\mathcal{C}'$ keskipituuksien n ja $\overline{n}'$ välisestä yhtälöstä

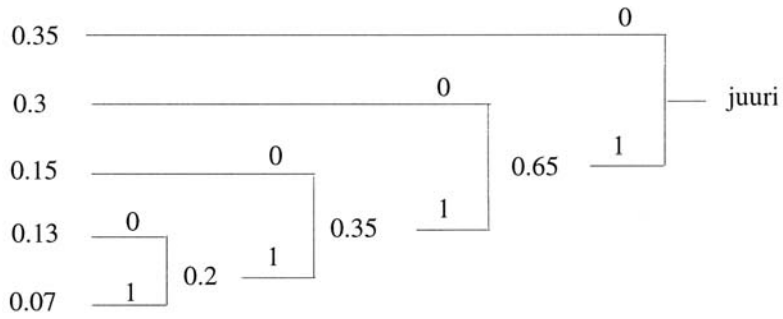
$$\begin{aligned} \overline{n} &= p_1 n_1 + p_2 n_2 + \cdots + p_{m-2} n_{m-2} + p_{m-1}\,(n+1) + p_m\,(n+1) \\ &= p_1 n_1 + p_2 n_2 + \cdots + p_{m-2} n_{m-2} + (p_{m-1} + p_m)\,n + (p_{m-1} + p_m) \\ &= \overline{n}' + (p_{m-1} + p_m) \end{aligned}$$

Tässähän $p_{m-1} + p_m$ on koodisanojen pituuksista riippumaton vakio, joten mikäli $\overline{n}'$ on pienin mahdollinen, niin samoin on myös $\overline{n}$.

Koodin $\mathcal{C}'$ voi konstruoida rekursiivisesti samalla tavalla konstruoimalla ensin koodin, jossa on yksi koodisana vähemmän. Näin päästään lopuksi kaksisanaiseen koodiin ja optimaalinen kaksisanainen koodi on tietenkin $\{0, 1\}$. Kirjanpidon summista $p_{m-1} + p_m$ sekä loppuun liitettävistä biteistä 0 ja 1 voi tehdä esimerkiksi seuraavan mallin mukaan: kirjoitetaan todennäköisyydet allekkain, piirretään kahdesta pienimmästä todennäköisyydestä oksat oikealle, ylempi 0 ja alempi 1, ja näiden oksien haarautumiskohta myös oikealle sekä kyseisen solmun arvoksi näiden kahden todennäköisyyden summa. Itse asiassa koodisanat konstruoidaan siis viimeisestä bitistä lähtien.

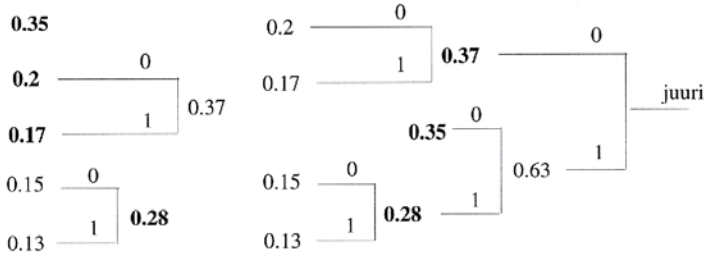
Esimerkki 43 *Olkoot kirjainten todennäköisyydet*

$$p_1 = 0.35, p_2 = 0.3, p_3 = 0.15, p_4 = 0.13, p_5 = 0.07$$



Kun puu luetaan juuresta lähtien, saadaan koodisanat 0, 10, 110, 1110 ja 1111. Tämä on oleellisesti sama koodi kuin sivun 53 koodi (4.5) $\mathcal{C} = \{1, 01, 000, 0010, 0011\}$*, oksat on vain nimetty toisella tavalla.*

Esimerkki 44 *Pieni uudelleenjärjestely on toisinaan tarpeen:*



Koodisanat ovat $00, 01, 10, 110, 111$ ja keskipituus on $\overline{n} = 2,28$.

Harjoitustehtäviä

1. Kirjainten koodaus

$$A \to 01 \quad E \to 111 \quad I \to 001 \quad S \to 10 \quad T \to 110$$

Koodaa sana "ASIAT". Miten monta bittiä lyhyempi tämä koodisana on kuin vastaava ASCII-koodilla koodattu ? Dekoodaa 100100111001.

2. Kirjainten todennäköisyydet ovat

$$p_A = 0.4 \quad p_E = 0.2 \quad p_I = 0.3 \quad p_S = 0.05 \quad p_T = 0.05$$

Määritä aakkoston entropia ja laske eo. tehtävän koodille kirjainten koodisanojen pituuksien

$$n_A \quad n_E \quad n_I \quad n_S \quad n_T$$

keskiarvo.

3. Aakkoston $L = \{a, i, t, e, s, n\}$ kirjainten todennäköisyydet ovat

$$p_a = 0.3 \quad p_i = 0.2 \quad p_t = 0.2 \quad p_e = 0.1 \quad p_s = 0.1 \quad p_n = 0.1$$

Kumpi seuraavista koodeista on parempi (tilansäästön kannalta)?

	a	i	t	e	s	n
koodi 1	11	10	011	010	001	000
koodi 2	1	01	001	0001	00001	00000

4. Symmetriselle binäärikanavalle on $p = 10^{-3}$. Laske todennäköisyys, että 6-pituisen sanan siirron aikana tapahtuu a) korkeintaan 2 virhettä b) vähintään 5 virhettä.

5. Konstruoi tekstille

KANNATTAA TARKASTAA TARKKAAN

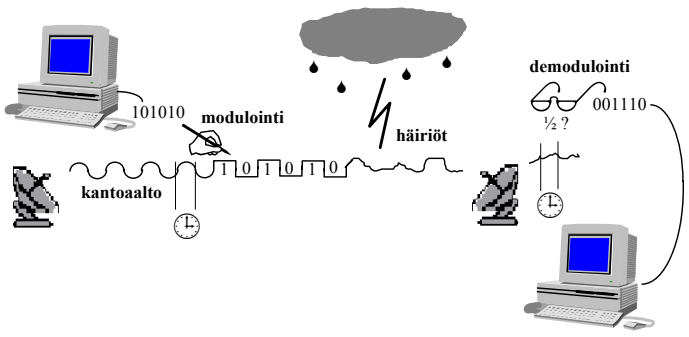
vaihtelevan pituinen koodi, jonka keskipituus eroaa tekstin binäärientropiasta vähemmän kuin 0,2.

Luku 5

Tiedonsiirtokanava

Diskreetti muistiton kanava

Tiedonsiirtokanavat ovat tekniseltä toteutukseltaan erilaisia. Tyyppiesimerkkejä sähkövirran kulkuun perustuvista kanavista ovat erilaiset tietokoneverkot ja puhelinlinjat. Radioaallot sekä valokuidut taas hyödyntävät aaltoliikettä ja talletusvälineinä käytettävissä levykkeissä sekä magneettinauhoissa tiedon tekninen esitys pohjautuu magneettisuuteen. Mutta yksinkertaisesti paperikin toimii tiedonsiirtokanavana ja samoin vanhojen äänilevyjen kaiverretut urat sekä filmin kemialliset yhdisteet. Oleellinen osa kanavan käyttöä on lähettäjän suorittama modulointi, tiedon kirjoittaminen kanavaan, sen muunto lähettäjän esitysmuodosta kanavan tekniseen muotoon, sekä vastaanottajan suorittama demodulointi, kanavan tulostuksen tulkitseminen, sen muunto kanavan teknisestä muodosta käyttäjän esitysmuotoon. Erilaiset kanavaan kohdistuvat häiriöt saattavat vääristää kanavassa siirrettävän tiedon teknistä esitysmuotoa, mahdollisesti muuttaen tiedon täysin toiseksi tai aiheuttaen vastaanottajalle tulkintavaikeuksia, jolloin kanavan tulostus ehkä tulkitaan väärin.



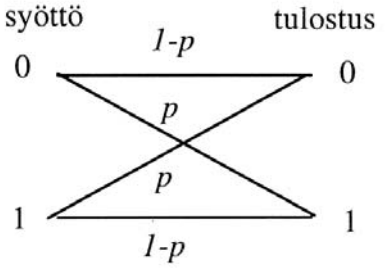
Kanavan tekniseen toteukseen liittyvä matematiikka on oma alansa. Esimerkiksi aaltomuotoisen kanavan modulointiteoriassa törmää ensimmäiseksi kompleksisiin sini-, kosini- ja eksponenttikäyriin sekä integraalimuunnoksiin. Puhtaasti algebrallisen koodausteorian kannalta modulointi ja demodulointi ovat osa kanavaa ja tiedonsiirtokanavan mallina käytetään diskreettiä muistitonta kanavaa. Diskreettisyys merkitsee - analogisen, jatkuvan kanavan vastakohtana - että kanavan syöttöaakkosto $S = \{s_1, s_2, \ldots, s_n\}$ ja tulostusaakkosto $T = \{t_1, t_2, \ldots, t_m\}$ ovat äärellisiä. Muistittomuus puolestaan merkitsee, että kanavan tulostuksen t jokainen kirjain riippuu ainoastaan syötön s vastaavasta kirjaimesta. Tämä edellyttää ensinnäkin, että modulointi ja demodulointi suoritetaan kirjaimittain, ja toiseksi, että häiriöt ovat sen verran pieniä, etteivät ne oleellisesti vaikuta kuin satunnaisiin kirjaimiin. Tyypillisesti kanavaan kohdistuvat häiriöt kestävät kauemmin kuin yhden merkin siirto, joten ne saattavat vaikuttaa useampaan kuin yhteen siirrettyyn merkkiin ja siirron aikana tapahtuvat virheet esiintyvät ryöppyinä eli useat peräkkäiset bitit saattavat olla virheellisiä. Mikäli häiriöt ovat heikkotehoisia ja kanava on teknisiltä ominaisuuksiltaan huippuluokkaa, voidaan kuitenkin olettaa virheiden olevan satunnaisia, hyvin harvoin esiintyviä. Näillä edellytyksillä häiriöt määräävät kanavan siirtotodennäköisyydet $P(t_i|s_j)$ eli ehdolliset todennäköisyydet sille, että kanavan tulostus on tulostusaakkoston kirjain t_i, kun syöttö on syöttöaakkoston kirjain s_j. Kanavan muistittomuus - tulostussanan $y = y_1 y_2 \cdots y_k$ kukin kirjain y_i riippuu ainoastaan syötön $x = x_1 x_2 \cdots x_k$ vastaavasta kirjaimesta x_i eikä lainkaan muista kirjaimista - määritellään todennäköisyyslaskennan riippumattomuuskäsitteen avulla: kanava on muistiton, jos tulostuksen y ehdollinen todennäköisyys syötön x suhteen on

$$P(y|x) = P(y_1|x_1) \cdot P(y_2|x_2) \cdots P(y_k|x_k)$$

Huomaa myös synkronointioletus, että kanava ei kadota kirjaimia eikä myöskään lisää niitä.

Oletamme itse asiassa kanavaksi oheisen kuvan esittämän symmetrisen binäärikanavan, jonka syöttö- ja tulostusaakkosto on sama binääriaakkosto $S = T = \{0, 1\}$. Kanavan siirtotodennäköisyydet ovat

$$P(1|0) = P(0|1) = p$$
$$P(0|0) = P(1|1) = 1 - p$$



eli todennäköisyydellä $1 - p$ tulostus on sama kirjain kuin syöttökin ja todennäköisyydellä p tiedonsiirron aikana on tapahtunut virhe, joka on

muuttanut syöttökirjaimen toiseksi kirjaimeksi. Hyvälle kanavalle on p hyvin pieni ja on todennäköisempää, että kanavan tulostus on oikein, kuin, että se on virheellinen. Esimerkiksi todennäköisyys, että kanavan tulostus on sana $t = 01101$, kun siihen on syötetty sana $s = 11001$, on

$$\begin{aligned} P(01101|11001) &= P(0|1)P(1|1)P(1|0)P(0|0)P(1|1) \\ &= p(1-p)p(1-p)(1-p) \\ &= p^2(1-p)^3 \end{aligned}$$

Jos tässä on $p = 10^{-6}$, niin todennäköisyys, että sanan $s = 11001$ siirron aikana on tapahtunut juuri kyseiset kaksi virhettä, bitit 1 ja 3, on siis

$$\begin{aligned} P(t|s) &= 10^{-6}(1-10^{-6})10^{-6}(1-10^{-6})(1-10^{-6}) \\ &= (10^{-6})^2(1-10^{-6})^3 < 10^{-12} \end{aligned}$$

Kanavan muistittomuudesta eli kirjainten riippumattomuudesta johtuen virheiden määrällä on binomitodennäköisyys ja siis n:n pituisen sanan siirron aikana tapahtuu e virhettä todennäköisyydellä

$$P(e \text{ virhettä}) = \binom{n}{e} p^e (1-p)^{n-e}$$

Muistettakoon tämän kaavan merkitys: binomikerroin $\binom{n}{e}$ on niiden tapojen määrä, joilla n:stä bitistä voidaan valita e bittiä, p^e on todennäköisyys, että kiinteissä kohdissa olevat e bittiä muuttuvat siirron aikana ja $(1-p)^{n-e}$ on todennäköisyys, että loput $n-e$ bittiä siirretään oikein. Esimerkiksi, kun $n = 5$ ja $p = 10^{-6}$, virhetodennäköisyydet $P(e) = P(e \text{ virhettä})$ ovat

$$\begin{aligned} P(0) &= \binom{5}{0}(10^{-6})^0(1-10^{-6})^{5-0} \approx 0.999995, \\ P(1) &= \binom{5}{1}(10^{-6})^1(1-10^{-6})^{5-1} \approx 0.4999995 \cdot 10^{-5}, \\ P(2) &= \binom{5}{2}(10^{-6})^2(1-10^{-6})^{5-2} \approx 0.999997 \cdot 10^{-11}, \\ P(3) &= \binom{5}{3}(10^{-6})^3(1-10^{-6})^{5-3} \approx 0.999998 \cdot 10^{-17}, \\ P(4) &= \binom{5}{4}(10^{-6})^4(1-10^{-6})^{5-4} \approx 0.4999995 \cdot 10^{-23}, \\ P(5) &= \binom{5}{5}(10^{-6})^5(1-10^{-6})^{5-5} = 10^{-30}. \end{aligned}$$

Tämän mukaan kaikkein todennäköisin kanavasta vastaanotettu sana on sama kuin kanavaan lähetettykin ja epätodennäköisin tapahtuma on jokaisen viiden bitin virheellinen vastaanotto. On kuitenkin muistettava, että käytännössä sanoja lähetetään valtavat määrät ja tällöin on jo erittäin mahdollista, että jokin sana tulee väärin.

Virheiden korjaus

Yksinkertaisin virheitä korjaava koodi on toistokoodi: tiedon lähettäjä lähettää viestinsä jokaisen bitin moneen kertaan peräkkäin. Esimerkiksi kolmen pituista toistokoodia käytettäessä jokainen bitti lähetetään kolmeen kertaan ja viesti $v = 1001$ koodataan koodisanaksi $c = 1110000\ 00111$, joka lähetetään kanavaan. Vastaanottaja jakaa kanavasta vastaanottamansa sanan $r = 101100011011$ kolmen bitin pituisiin osiin, lohkoihin, ja dekoodaa jokaisen lohkon bitiksi 0 tai 1 sen mukaan, kumpaa on lohkossa enemmän. Esimerkkimme dekoodaus antaa $r = 101|100|011|011 \rightarrow 1011 = v'$. Näin menetellen ensimmäisessä, toisessa ja neljännessä lohkossa oleva virheellinen bitti saadaan korjattua, mutta kolmannessa lohkossa olevat kaksi virheellistä bittiä aiheuttavat dekoodausvirheen. Yleisesti, jos toistokoodin lohkon pituus on $n = 2e + 1$, koodilla voidaan korjata kaikki lohkot, joissa on korkeintaan e virheellistä bittiä, mutta lohkot, joissa on vähintään $e + 1$ virheellistä bittiä, dekoodataan aina väärin.

Toistokoodi on sikäli paras mahdollinen koodi, että millään muulla koodilla ei voida korjata enempää virheitä. Toisaalta, yhden lohkon n:stä bitistä ainoastaan yksi on varsinainen viestin informaatiobitti ja loput $n-1$ bittiä ovat virheenkorjausta varten lisättyjä pariteetintarkistusbittejä. Koodin informaatiosuhde $R = 1/n$ on pieni ja suurin osa ajasta kuluu pariteetintarkistusbittien siirtoon. Kun halutaan parempi informaatiosuhde $R = k/n$, jossa lohkon n:stä bitistä k on viestin informaatiobittejä ja loput $n-k$ ovat pariteetintarkistusbittejä, menetellään seuraavasti. Tiedon lähettäjä koodaa viestin $v = 101001101001$ jakamalla sen k:n pituisiin lohkoihin, $v = 1010|0110|1001$, ja koodaamalla kunkin lohkon v_i n-pituiseksi koodisanaksi c_i:

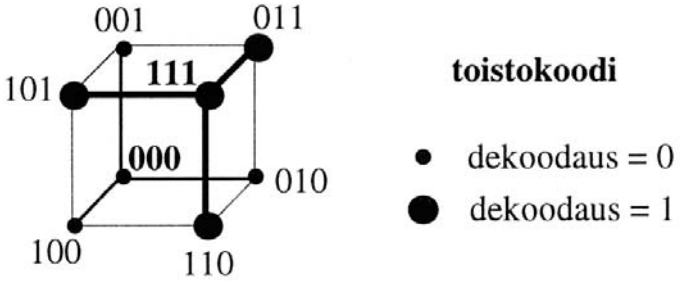
$v =$	1010	0110	1001	$\rightarrow$	$c =$	1010101	0110110	1001101
$v =$	v_1	v_2	v_3	$\rightarrow$	$c =$	c_1	c_2	c_3

Tässä käytetty koodi on esimerkki systemaattisesta lohkokoodista: yhden viestilohkon v_i koodaus suoritetaan lisäämällä sen loppuun pariteetintarkistusbittejä. Koodilohkon c_i 7:stä bitistä 4 on informaatiobittejä, joten koodin informaatiosuhde on $R = 4/7$. Lohkon 3 pariteetintarkistusbittiä esimerkkikoodissamme mahdollistavat yhden virheen korjaamisen lohkossa.

Aiemmin todettiin virhetodennäköisyyden $P(e)$ vähenevän virheiden määrän e kasvaessa. Tämän mukaisesti dekoodaus suoritetaan dekoodaamalla vastaanotettu sana ensin koodisanaksi, joka eroaa vähiten vastaanotetusta sanasta, ja sen jälkeen kyseinen koodisana dekoodataan edelleen vastaavaksi viestiksi. Tätä periaatettahan käytettiin toistokoodiesimerkissä: jos kolmen peräkkäisen vastaanotetun bitin joukossa on kaksi tai kolme 1:stä, se eroaa koodilohkosta 111 korkeintaan yhdessä kohdassa mutta eroaa koodilohkosta 000 vähintään kahdessa kohdassa.

Toistokoodin dekoodaus on siis seuraava:

vastaanotettu sana		**koodisana**		**viestibitti**
$000, 001, 010, 100$	$\rightarrow$	000	$\rightarrow$	0
$111, 110, 101, 011$	$\rightarrow$	111	$\rightarrow$	1



Virheenkorjaus perustuu siis yksinkertaisesti siihen, että koodattaessa k:n pituisia sanoja n:n pituisiksi koodisanoiksi koodisanoja on 2^k kappaletta kun taas n:n pituisia sanoja on kaikkiaan 2^n kappaletta ja näin vastaanottaja voi dekoodata koodisanan lähellä olevia sanoja samaksi koodisanaksi ja viestiksi. Koodin $\mathcal{C} = \{c_1, ..., c_M\}$ $(M = 2^k)$ kuhunkin koodisanaan c_i liittyy siis joukko Y_i sanoja, jotka dekoodaatan koodisanaksi c_i. Yllä olevalle toistokoodille koodisanat ja niitä vastaavat sanajoukot, jotka dekoodataan kyseiseksi koodisanaksi, ovat $c_0 = 000$ ja $Y_0 = \{000, 001, 010, 100\}$ sekä $c_1 = 111$ ja $Y_1 = \{111, 110, 101, 011\}$. Huomaa, että joukossa Y_0 on koodisana c_0 sekä ne sanat, jotka eroavat siitä vain yhdessä kohdassa, että joukossa Y_1 on koodisana c_1 sekä ne sanat, jotka eroavat siitä vain yhdessä kohdassa, ja että kukin kolmen pituinen sana on tarkalleen toisessa joukoista Y_0 ja Y_1. Yleisesti, kun halutaan e virhettä korjaava koodi, koodisanat c_i ja joukot Y_i on valittava siten, että joukossa Y_i on kaikki ne sanat, jotka eroavat koodisanasta c_i korkeintaan e:ssä kohdassa, ja että kahta eri koodisanaa c_i ja c_j vastaavissa joukoissa Y_i ja Y_j ei ole yhteisiä alkioita.

Mikäli kanavaan lähetetään koodisana c_i ja kanavan tulostus t on joukossa Y_i, dekoodaus suoritetaan oikein, $t \rightarrow c_i$, ja virheet saadaan korjattua, mutta jos tulostus t ei olekaan lähetetyn koodisanan c_i määräämässä joukossa Y_i vaan jonkin toisen koodisanan c_j määräämässä joukossa Y_j, niin vastaanottaja dekoodaakin tulostuksen virheellisesti koodisanaksi c_j. Tapahtuneen dekoodausvirheen todennäköisyyttä P_{virhe} voidaan arvioida joukkojen Y_i avulla. Tämä kuitenkin jätetään tekemättä.

Kaikki jatkossa tarkasteltavat koodit ovat lohkokoodeja ja niiden koodauksessa ja dekoodauksessa rajoitutaan tarkastelemaan vain yhtä lohkoa. Dekoodaus suoritetaan aina yllä mainitun periaatteen mukaisesti eli olettaen, että mahdollisimman vähän virheitä on tapahtunut, ja dekoodaamalla vastaanotettu sana lähimmäksi koodisanaksi.

Esimerkki 45 *Tarkastellaan minkälainen merkitys on tiedonsiirrossa yksinkertaisella virheentarkistuksella. Siirretään 8-pituisia sanoja kanavassa nopeudella 10000 bittiä sekunnissa. Oletetaan, että bittikohtainen*

virhetodennäköisyys on 10^{-6}. Mikäli ei ole minkäänlaista virheen tarkistusta, niin todennäköisyys, että sana saadaan virheettömänä läpi on

$$\left(1-10^{-6}\right)^{8}$$

ja siis todennäköisyys, että jonkinlaisia virheitä tulee on

$$1-\left(1-10^{-6}\right)^{8}\approx 7.\,999972\times 10^{-6}$$

Täten sekunnissa tulee virheellisiä sanoja keskimäärin

$$10000\cdot\left(1-\left(1-10^{-6}\right)^{8}\right)/8\approx 0.00\,999965$$

ja tunnissa n. ($60\cdot 60\cdot 0.00\,999965\approx$*)* 36.
Lisätään sitten koodiin yhden bitin pituinen pariteetintarkistus, joka ilmoittaa mikäli on tapahtunut virhe sanassa yhden bitin kohdalla. (Tämän tyyppinen on esim. sosiaalitunnuksen viimeinen merkki.) Nyt virhe jää havaitsematta vasta mikäli niitä tapahtuu 2 sanaa kohti. Tällaisen todennäköisyys on

$$1-\left(1-10^{-6}\right)^{9}-9\left(1-10^{-6}\right)^{8}10^{-6}\approx 3.\,599983\times 10^{-11}$$

Sekunnissa tulee nyt virheellisiä sanoja

$$10000\cdot\left(3.\,599983\times 10^{-11}\right)/9\approx 3.\,99998\,1333\times 10^{-8}$$

ja vuodessa

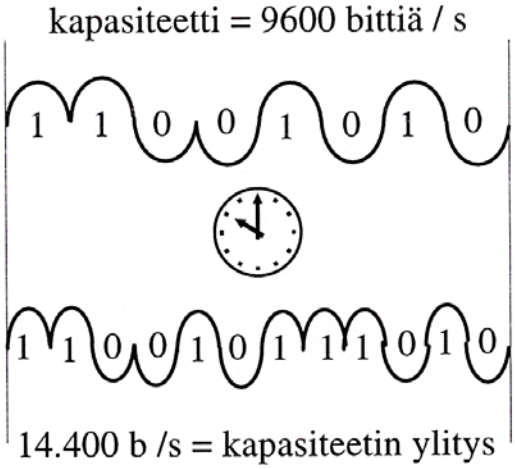
$$365\cdot 24\cdot 60\cdot 60\cdot 3.\,99998\,1333\times 10^{-8}\approx 1.\,261434113$$

Täten ilman tarkistusta tunnissa saatiin yhtä monta virheellistä sanaa kuin nyt n. $28\frac{1}{2}$ *vuodessa.*

Kapasiteetti

Siirrettäessä tietoa tietokoneesta toiseen puhelinverkkoja käyttäen, tiedonsiirtokanava koostuu lähettäjän modeemista (modulointi), puhelinlinjasta sekä vastaanottajan modeemista (demodulointi). Kanavan siirtonopeuden määrää hitaampi käytettävistä modeemeista. Jos esimerkiksi yrittää modeemilla lähettää 14.400 bittiä sekunnissa vastaanottajalle, jonka modeemin maksimisiirtonopeus on 9.600 bittiä sekunnissa, ei vastaanottaja kykenekään tunnistamaan lähetettyä tietoa. Tässä lähettäjä käyttää kanavaa väärin ylittäen sen kapasiteetin, sen mak-

simisiirtonopeuden 9.600 bittiä sekunnissa.



Tutkittaessa tiedonsiirtoa häiriöiden aiheuttamien virheiden kannalta tilanne on aivan vastaava. Virheiden korjaaminen vaatii ylimääräisen tarkistustiedon lisäämistä varsinaiseen viesti-informaatioon ja mikäli yritetään käyttää koodia, jonka informaatiosuhde on liian suuri virheiden esiintymistiheyden suhteen, virheitä ei pystytäkään korjaamaan. Tässä tapauksessa kapasiteetti ei ole fyysinen siirtonopeus, vaan siirtotodennäköisyyksien $P(t|s)$ - virheiden esiintymistiheyden - määräämä maksimi-informaatiosuhde, jota voidaan käyttää virheitä korjaavassa koodissa.

Systemaattista lohkokoodia käytettäessä koodilohkon rakenne on yksinkertainen:

10001010101010101	1010011
k viestibittiä	$n-k$ pariteetintarkistisbittiä

Kaava $n = k + (n - k)$ vastaa tätä rakennetta. Epäsystemaattista koodia käytettäessä koodilohkon n bittiä eivät jakaannu suoraan viestin k:ksi bitiksi ja $(n - k)$:ksi pariteetintarkistusbitiksi, vaan viestin k bittiä on koodauksessa upotettu koodilohkon n:ään bittiin ilman bittirajoja. Hieman toisin ajateltuna koodilohkon n bittiä muodostavat informaation, mikä pitää sisällään viestin k bittiä ja $n - k$ pariteetintarkistusbittiä. Näin kaava $n = k + (n - k)$ kuvaa edelleenkin koodilohkon sisäistä rakennetta kertoen viestibittien ja pariteetintarkistusbittien osuudet. Kun näitä osuuksia tarkastellaan keskimääräisesti siirrettävää bittiä kohti, yhtälö $n = k + (n - k)$ jaetaan lohkopituudella n, jolloin $1 = k/n + (n-k)/n$. Tämän mukaan yhden siirrettävän bitin mukanaan kuljettamasta kokonaisinformaatiosta 1 viestin osuus on k/n (koodin informaatiosuhde) ja tarkistustiedon osuus on $(n - k)/n = 1 - k/n$. Tarkistusten tarkoituksena on kompensoida kanavan vaikutukset, joten tutkimalla tiedonsiirtovirheiden keskimääräistä osuutta siirrettyä bittiä kohden, voidaan määrätä, paljonko tarkistusta $(1-k/n)$ on lisättävä viesti-informaatioon (k/n), jotta virheet voidaan kompensoida.

Siirrettäessä käyttöympäristölle tyypillistä tietoa valittua koodia käyttäen kanavan syöttöaakkoston $\mathcal{S}$ kirjaimilla on kyseiselle käytölle ominainen jakauma $Q(s)$, $s \in S$, joka määrää syötön entropian

$$H(S) = -\sum_{s \in S} Q(s) \log Q(s)$$

Lähdekoodauksen yhteydessä termi $-\log Q(s) = \log(1/Q(s))$ tulkittiin niiden bittien määräksi, mikä tarvitaan kirjaimen s esittämiseksi binääriaakkostossa, ja entropia merkitsi kaikkien kirjainten esityksiin tarvittavien bittimäärien keskiarvoa. Voidaan myös ajatella kirjaimen s sisältävän nuo tarvittavat bitit, jolloin $-\log Q(s)$ merkitsee kirjaimen s sisältämän informaation määrää. Tämä jälkimmäinen tulkinta on tiedonsiirron yhteydessä asianmukaisempi, sisältäähän kanavaan syötettävä koodisana koodaamattoman viestin sekä siihen lisätyn tarkistustiedon. Termiä $I(s) = -\log Q(s)$ nimitetäänkin yleisesti kirjaimen s sisältämäksi (itse-)informaatioksi.

Esimerkki 46 *Kun symmetriseen binäärikanavaan syötetään bittejä 0 ja 1 yhtä suurilla todennäköisyyksillä, $Q(0) = Q(1) = 1/2$, niiden informaatiot ovat samat,*

$$I(0) = -\log Q(0) = -\log 1/2 = 1$$
$$I(1) = -\log Q(1) = -\log 1/2 = 1$$

ja syöttöaakkoston $S = \{0, 1\}$ entropia on

$$\begin{aligned} H(S) &= -Q(0)\log Q(0) - Q(1)\log Q(1) \\ &= Q(0)I(0) + Q(1)I(1) = \frac{1}{2}\cdot 1 + \frac{1}{2}\cdot 1 = 1 \end{aligned}$$

Tämä on täysin odotettu tulos. Molemmat bitit ovat samanarvoiset, kummankaan syöttö ei sisällä informaatiota sen enempää kuin toisenkaan syöttö.

Esimerkki 47 *Jos taas bittien todennäköisyydet ovat $Q(0) = 1/4$ ja $Q(1) = 3/4$, niin*

$$I(0) = -\log Q(0) = -\log 1/4 = 2$$
$$I(1) = -\log Q(1) = -\log 3/4 = 0.415037499$$

$$H(S) = Q(0)I(0) + Q(1)I(1) = \frac{1}{4}\cdot 2 + \frac{3}{4}\cdot 0.415037499 = 0.811278124$$

Tässä tapauksessa harvinaisemman bitin 0 sisältämä informaatio on suurempi kuin yleisemmän bitin 1 sisältämä informaatio. Tulos voidaan tulkita kahdella tavalla: bittiin 0 on pakattu enemmän tietoa kuin bittiin 1, tai bitin 0 esiintyminen syöttönä kertoo koodaamattomasta viestistä enemmän kuin bitin 1 esiintyminen.

Syöttöaakkoston $\mathcal{S}$ jakauma $Q(s)$ sekä siirtotodennäköisyydet $P(t|s)$ määräävät yhdessä jakauman myös tulostusaakkostolle $\mathcal{T}$. Tapahtuma " syöttö on s ja tulostus on t" on syöttö-tulostusavaruuden $S \times T$ tapaus (s,t) ja sen todennäköisyys on

$$P(s,t) = Q(s)P(t|s)$$

eli syötön s todennäköisyys kerrottuna todennäköisyydellä, että tulostus on t, kun syöttö on s. Sama tulostus t voi olla tuloksena eri syötoistä ja sen todennäköisyys $P(t)$ on kaikkien eri tapausten todennäköisyyksien summa eli

$$P(t) = \sum_{s \in S} P(s,t) = \sum_{s \in S} Q(s)P(t|s)$$

Tämän mukaisesti tulostuskirjaimen t informaatio $I(t) = -\log P(t)$ pitää sisällään itse asiassa tietoa sekä syötöstä (termi $Q(s)$) että tiedon siirrosta (termi $P(t|s)$). Samoin näiden informaatioiden $I(t)$ keskiarvo eli tulostusaakkoston T entropia

$$\begin{aligned} H(T) &= -\sum_{t \in T} P(t) \log P(t) = -\sum_{t \in T} \sum_{s \in S} P(s,t) \log P(t) \\ &= -\sum_{t \in T} \sum_{s \in S} Q(s)P(t|s) \log P(t) \end{aligned}$$

sisältää informaatiota sekä syötöstä että siirrosta.

Esimerkki 48 *Kun symmetrisen binäärikanavan siirtotodennäköisyydet ovat*

$$\begin{aligned} P(0|0) = P(1|1) = 1 - p \\ P(1|0) = P(0|1) = p \end{aligned}$$

ja kanavan syöttöbitit ovat yhtä todennäköisiä, $Q(0) = Q(1) = 1/2$*, niin syöttö-tulostustapahtumien* (s,t) *todennäköisyydet ovat*

$$\begin{aligned} P(0,0) &= Q(0)P(0|0) = \tfrac{1}{2}(1-p) \\ P(0,1) &= Q(0)P(1|0) = \tfrac{1}{2}p \\ P(1,0) &= Q(1)P(0|1) = \tfrac{1}{2}p \\ P(1,1) &= Q(1)P(1|1) = \tfrac{1}{2}(1-p) \end{aligned}$$

ja tulostusbittien todennäköisyydet ovat

$$\begin{aligned} P(0) = P(0,0) + P(1,0) = Q(0)P(0|0) + Q(1)P(0|1) = \tfrac{1}{2}(1-p) + \tfrac{1}{2}p = \tfrac{1}{2} \\ P(1) = P(1,1) + P(0,1) = Q(1)P(1|1) + Q(0)P(1|0) = \tfrac{1}{2}(1-p) + \tfrac{1}{2}p = \tfrac{1}{2} \end{aligned}$$

Odotetusti tulostusbitit ovat yhtä todennäköiset, ovathan syöttöbititkin yhtä todennäköisiä ja kanavan symmetrisyydestä johtuen kummankin siirto on tilastollisesti samanlainen.

Esimerkki 49 *Jos taas syöttötodennäköisyydet ovat* $Q(0) = 1/4$, $Q(1) = 3/4$, *niin*

$$\begin{aligned} P(0,0) &= Q(0)P(0|0) = \tfrac{1}{4}(1-p) \\ P(0,1) &= Q(0)P(1|0) = \tfrac{1}{4}p \\ P(1,0) &= Q(1)P(0|1) = \tfrac{3}{4}p \\ P(1,1) &= Q(1)P(1|1) = \tfrac{3}{4}(1-p) \end{aligned}$$

$$\begin{aligned} P(0) = P(0,0) + P(1,0) = \tfrac{1}{4}(1-p) + \tfrac{3}{4}p = \tfrac{1}{4} + \tfrac{2}{4}p = \tfrac{1}{2} - \tfrac{1}{4}(1-2p) \\ P(1) = P(1,1) + P(0,1) = \tfrac{3}{4}(1-p) + \tfrac{1}{4}p = \tfrac{3}{4} - \tfrac{2}{4}p = \tfrac{1}{2} + \tfrac{1}{4}(1-2p) \end{aligned}$$

Mikäli kanava on täysin häiriötön, sen virhetodennäköisyys on $p = 0$ ja tulostustodennäköisyydet ovat luonnollisesti samat kuin syöttötodennäköisyydetkin: $P(0) = 1/4 = Q(0)$, $P(1) = 3/4 = Q(1)$. Toisena ääritapauksena on lähinnä teknistä vikaa vastaava virhetodennäköisyys $p = 1$, mikä merkitsee bitin muuttumista siirrossa aina toiseksi. Tämä näkyy todennäköisyyksissäkin: $P(0) = 3/4 = Q(1)$, $P(1) = 1/4 = Q(0)$. Tilanne on kaoottinen silloin, kun $p = 1/2$. Tällöinhän $P(0) = P(1) = 1/2$, mikä merkitsee, että lähetetään kumpi tahansa bitti 0 tai 1, niin tulostus on yhtä suurella todennäköisyydellä kumpi tahansa bitti 0 tai 1, joten tulostus on riippumaton syötöstä eikä vastaanottajalla ole ainakaan tilastollisia keinoja päätellä tulostuksesta, mikä on ollut syöttönä.

Tyypillisesti virhetodennäköisyys p on pieni, $p < 1/2$, ja tällöin tulostusbitti 0 todennäköisempi kuin bitti 1, kuten syötössäkin. Esimerkiksi kun $p = 10^{-6}$, niin

$$\begin{aligned} P(0,0) &= 0.24999975 \\ P(0,1) &= 0.25 \cdot 10^{-6} \\ P(1,0) &= 0.75 \cdot 10^{-6} \\ P(1,1) &= 0.74999925 \end{aligned}$$

$$\begin{aligned} P(0) = \tfrac{1}{2} - 0.2499995 = 0.2500005 \approx 1/4 = Q(0) \\ P(1) = \tfrac{1}{2} + 0.2499995 = 0.7499995 \approx 3/4 = Q(1) \end{aligned}$$

Näin ollen pieni virhetodennäköisyys ei vielä sanottavasti vaikuta bittien todennäköisyyksiin bittien kulkiessa kanavan läpi. Mutta tulostuksen ja syötön välillä on kuitenkin pieni ero, tulostuksessa on mahdollisten virheiden aiheuttama epävarmuus siitä, mitä kanavaan on syötetty. Tämä näkyy tietenkin myös tulostusbittien sisältämässä informaatiossa sekä tulostusaakkoston $\mathcal{T}$ entropiassa:

$$\begin{aligned} I_T(0) &= -\log P(0) = 1.999997115 \approx 2 = I_S(0) \\ I_T(1) &= -\log P(1) = 0.415038461 \approx 0.415037499 = I_S(1) \end{aligned}$$

$$H(T) = P(0)I(0) + P(1)I(1) = 0.811278917 \approx 0.811278124 = H(S)$$

Pariteetintarkistusten lisäämisen tarkoitus on varustaa siirrettävä tieto sellaisella määrällä tarkistustietoa, että mainittu epävarmuus tulostuksen oikeellisuudesta pystytään kompensoimaan ja (todennäköisin) syöttö pystytään ratkaisemaan saadusta tulostuksesta.

Dekoodaajan kannalta tulostusaakkoston entropia $H(T)$ on oleellista tilastollista dataa, sehän on keskimääräinen vastaanotetun bitin sisältämä informaatio, joka muodostuu syötön sekä siirron todennäköisyyksistä, ja dekoodaajan tehtävänähän on erottaa vastaanotetusta sanasta t erikseen kanavaan syötetty sana s ja siirron aikana tapahtuneet virheet ($s \to t$). Miten se suoritetaan valitulle koodille, on ongelma, jota käsitellään myöhemmin. Toistaiseksi tarkastelemme ongelmaa informaatiokäsitteen avulla.

Informaatio- ja entropiakäsitteet eivät rajoitu pelkästään kirjaimiin, vaan ne on määritelty kaikissa todennäköisyysavaruuksissa. Niinpä kanavan siirtotodennäköisyydet $P(t|s)$ määräävät ehdollisen informaation

$$I(t \,|\, s) = -\log P(\, t \,|\, s\,)$$

Deekodauksen kannalta tämä on se informaatiomäärä tiedon siirrosta mahdollisine virheineen, minkä tulostus t antaa dekoodaajalle, kun syöttö on s. Keskiarvoa laskettaessa huomioidaan kaikki mahdolliset syötöt ja tulostukset, jolloin virheitä keskimääräisesti kuvaavaksi entropiaksi saadaan

$$H(\, S \to T) = \sum_{s\in S}\sum_{t\in T} P(\, s,t) I(t \,|\, s\,) = -\sum_{s\in S}\sum_{t\in T} Q(s) P(t \,|\, s\,) \log P(t \,|\, s\,)$$

Esimerkki 50 *Symmetriselle binäärikanavan siirtotodennäköisyyksien*

$$\begin{aligned} P(1|0) &= P(0|1) = p \\ P(0|0) &= P(1|1) = 1-p \end{aligned}$$

informaatiot ovat

$$\begin{aligned} I(1|0) &= I(0|1) = -\log p \\ I(0|0) &= I(1|1) = -\log(1-p) \end{aligned}$$

Kun syöttötodennäköisyydet ovat $Q(0) = q$, $Q(1) = 1-q$, saadaan

$$\begin{aligned} H(\, S \;\to\; T) &= \sum_{s\in S}\sum_{t\in T} Q(s) P(t\,|s) I(t\,|s\,) \\ &= Q(0)P(00)I(00) + Q(0)P(10)I(10) \\ &\quad + Q(1)P(01)\, I(01) + Q(1)P(11)\, I(11) \\ &= -q(1-p)\log(1-p) - qp\log p \\ &\quad -(1-q)p\log p - (1-q)(1-p)\log(1-p) \\ &= -\left[q(1-p) + (1-q)(1-p)\right]\log(1-p) - \left[qp + (1-q)p\right]\log p \\ &= -(1-p)\log(1-p) - p\log p \end{aligned}$$

Lopputuloksen riippumattomuus syötön jakaumasta $Q(s)$ saattaa ensin tuntua yllättävältä, mutta on tarkemmin tarkasteltuna luonnollinen. Entropian $H(S \to T)$ on tarkoitus kuvata kanavan

$$\begin{array}{ccc} & P(t|s) & \\ s & \longrightarrow & t \end{array}$$

ominaisuuksia eikä sen käyttöä $Q(s)$ ja kun kanava on symmetrinen, käyttö $Q(s)$ ei ilmene entropiassa $H(S \to T)$, vaan entropia on tuttu 'siirtoaakkoston' kirjainten Oikein = '0 → 0', Virhe = '0 → 1' todennäköisyyksien $P(\text{Oikein}) = 1 - p$ ja $P(\text{Virhe}) = p$ määräämä entropia. Epäsymmetriselle kanavalle syötön jakaumakin $Q(s)$ näkyy entropiassa $H(S \to T)$.

Kanavan tulostuksen sisältämässä keskimääräisessä informaatiossa $H(T)$ on kanavan ja siis mahdollisten virheiden osuus $H(S \to T)$. On intuitiivisesti selvää, että mikäli virheet halutaan korjata, niin käytettävässä koodissa on pariteetintarkistusten kompensoitava tämä virheinformaatio $H(S \to T)$ samalla informaatiomäärällä. Loppuosa $H(T) - H(S \to T)$ on käytettävissä todellisen siirrettäväksi halutun viestin käyttöön. Tämä intuitio näkyy vertailussa

	kokon.	=	**data**	+	**virh. tai korj.**
tulostusbitti	$H(T)$	=	$H(T) - H(S \to T)$	+	$H(S \to T)$
koodilohko	n	=	k	+	$n - k$
koodibitti	1	=	k/n	+	$(n-k)/n$

Käytettävän koodin informaatiosuhde k/n voi siis olla korkeintaan kanavan sallima informaatiomäärä $H(T) - H(S \to T)$. Tämä riippuu kanavan käytöstä eli syötön jakaumasta $Q(s)$, sillä

$$\begin{aligned} H(T) - H(S \to T) &= -\sum_{t \in T} \sum_{s \in S} Q(s) P(t \mid s) \log P(t) \\ &\quad + \sum_{s \in S} \sum_{t \in T} Q(s) P(t \mid s) \log P(t \mid s) \\ &= \sum_{s \in S} \sum_{t \in T} Q(s) P(t \mid s) \frac{\log P(t \mid s)}{\log P(t)} \end{aligned}$$

Kun halutaan absoluuttinen, syöttöjakaumasta riippumaton mitta, etsitään tämän informaatiomäärän maksimi syöttöjakauman $Q(s)$ suhteen. Tuo maksimi on se ehdoton yläraja käytettävien koodien informaatiosuhteelle ja sitä nimitetäänkin kanavan kapasiteetiksi C:

$$C = \max_Q \left(H(T) - H(\, S \to T) \right) = \max_Q \left(\sum_{s \in S} \sum_{t \in T} Q(s) P(t \mid s) \frac{\log P(t \mid s)}{\log P(t)} \right)$$

Esimerkki 51 *Esimerkissä 50 todettiin symmetrisen binäärikanavan entropian*

$$H(S \to T) = -(1-p)\log(1-p) - p \log p$$

olevan syöttöjakaumasta $Q(s)$ riippumattoman, joten kanavan kapasiteetti löydetään etsimällä tulostusaakkoston entropian $H(T)$ maksimi. Koska $H(T)$ on kaksikirjaimisen aakkoston $\{0,1\}$ entropia, niin Lauseen 36 mukaan $H(T) \leq log_2 2 = 1$ ja $H(T) = 1$ tarkalleen silloin, kun tulostusbitit 0 ja 1 ovat yhtä todennäköisiä eli $P(0) = P(1) = 1/2$. Esimerkissä 48 nähtiin, että syöttöbittien ollessa yhtä todennäköiset, $Q(0) = Q(1) = 1/2$, myös tulostusbitit ovat yhtä todennäköisiä. Näin ollen syötön jakauma $Q(0) = Q(1) = 1/2$ antaa maksimaalisen tulostusentropian $H(T) = 1$ ja kanavan kapasiteetiksi tulee

$$C = 1 + (1-p)\log(1-p) + p \log p$$

Seuraavan lauseen mukaan tiedonsiirto saadaan miten luotettavaksi tahansa, kunhan käytetään riittävän pitkiä koodeja, joiden informaatiosuhde k/n on pienempi kuin kanavan kapasiteetti C. Sen sijaan informaatiosuhteen ylittäessä kapasiteetin, tehdään melko varmasti dekoodausvirheitä.

Lause 52 (Kanavakoodausteoreema) *Olkoon diskreetin muistittoman kanavan kapasiteetti C. Oletetaan, että tiedonsiirrossa käytetään koodeja, joiden informaatiosuhde k/n on sama luku R kaikilla mahdollisilla lohkopituuksilla n. Mikäli $R < C$, niin on olemassa sellainen jono koodeja, joille dekoodausvirheen todennäköisyys P_{virhe} lähenee eksponentiaalisesti nollaa koodin lohkopituuden n lähestyessä ääretöntä. Kääntäen, mikäli koodien informaatiosuhde $R > C$, niin dekoodausvirheen todennäköisyys P_{virhe} lähestyy 1:stä.*

Esimerkki 53 *Jos symmetrisen binäärikanavan virhetodennäköisyys on $p = 10^{-6}$, sen kapasiteetti on edellisen esimerkin perusteella $C = 0.999978626$. Tämä merkitsee, että oleellisesti lähes kaikki tiedonsiirtovirheet voidaan korjata käyttämällä pitkiä koodeja, joiden informaatiosuhde $k/n = 0.99$ eli joissa koodisanan n:stä bitistä 99 % on todellista dataa ja vain 1 % on virheiden korjaamiseen tarvittavia pariteetintarkistusbittejä. Jos esimerkiksi koodin pituus on $n = 100.000$, databittien määrä voi olla $k = 99.000$ ja loput 1000 bittiä ovat pariteetintarkistusbittejä. Todellisuus ei tietenkään ole aivan näin yksinkertaista, hyvien koodien konstruointi on vaikeaa ja niiden koodaus sekä dekoodaus voivat olla liian monimutkaisia tai kalliita toteuttaa.*

Luku 6

Algebralliset koodit

Kanavakoodausteoreema kertoo hyvien koodien olemassaolon, mutta sen todistus on puhtaasti todennäköisyyksiin perustuva eikä kerro koodien konstruoinnista senkään vertaa kuin lähdekoodausteoreeman todistus. Käyttökelpoisuusvaatimus koodille on koodauksen ja dekoodauksen nopeus. Tämä edellyttää, että koodisanoja ei valita satunnaisesti, vaan koodin on oltava jollakin tavalla säännöllinen, sillä on oltava koodauksen ja dekoodauksen nopeuden mahdollistava algebrallinen rakenne.

Additiiviset virheet

Binääriaakkosto $B = \{0, 1\}$ on itse asiassa jäännösluokkasysteemi $\mathbb{Z}_2$ modulo 2, joten siinä on määritelty sekä yhteenlasku että kertolasku:

$$\begin{array}{ll}
0 + 0 = 0 & \text{parillinen + parillinen = parillinen} \\
0 + 1 = 1 & \text{parillinen + pariton = pariton} \\
1 + 0 = 1 & \text{pariton + parillinen = pariton} \\
1 + 1 = 0 & \text{pariton + pariton = parillinen} \\
0 \times 0 = 0 & \text{parillinen} \times \text{parillinen = parillinen} \\
0 \times 1 = 0 & \text{parillinen} \times \text{pariton = parillinen} \\
1 \times 0 = 0 & \text{pariton} \times \text{parillinen = parillinen} \\
1 \times 1 = 1 & \text{pariton} \times \text{pariton = pariton}
\end{array}$$

Nämä toteuttavat tutut vaihdanta-, liitäntä- ja osittelulait kuten reaaliluvutkin. Edelleen havaitaan, että alkioiden 0 ja 1 vasta-alkiot ovat $-0 = 0$ ja $-1 = 1$ ja että ainoalla nollasta eroavalla alkiolla 1 on käänteisalkio $1^{-1} = 1$. (Nämä ominaisuudet merkitsevät sitä,

että aakkosto $B = Z_2$ on *äärellinen kunta*. Yleisesti, jokainen jäännösluokkasysteemi $\mathbb{Z}_p$, missä p on alkuluku, on kunta.) Kuntaominaisuuksista seuraa, että binäärivektorit $b \in B_n$ muodostavat vektoriavaruuden, joten lineaarialgebran menetelmät ovat käytettävissä. Tämän mukaisesti n:n pituiset sanat samaistetaankin tarvittaessa vastaaviin vektoreihin, esimerkiksi $01101 = (0, 1, 1, 0, 1)$. Huomaa, että koska binäärikunnassa on $-0 = 0$ ja $-1 = 1$, niin siinä on vähennyslasku sama kuin yhteenlasku: $a - b = a + b$.

Mikäli kanavaan on lähetetty koodisana c ja kanavasta vastaanotettu sana on r, niin tapahtunut virhe on vektori (sana) $e = r - c$. Koska vastaanotettu sana r voidaan esittää koodisanan ja virheen summana muodossa $r = c + (r - c) = c + e$, puhutaan additiivisista virheistä. Esimerkiksi jos kanavaan on lähetetty koodisana $c = 01101$ ja vastaanotettu sana on $r = 01011$, niin tapahtunut virhe e on $e = r - c = 01011 - 01101 = (0, 1, 0, 1, 1) - (0, 1, 1, 0, 1) = (0, 0, -1, 1, 0) = (0, 0, 1, 1, 0) = 00110$.

Huomaa, että virhevektorin e alkio on 1 tarkalleen silloin, kun kyseisessä kohdassa on tiedonsiirron aikana bitti muuttunut toiseksi; esimerkissä kolmas bitti ($1 \to 0$) ja neljäs bitti ($0 \to 1$). Virhevektorin e nollasta eroavien alkioiden määrä on siis tapahtuneiden virheiden määrä. Dekoodaajan tehtävänä on löytää tuo virhevektori e ainoana tietonaan käytetty koodi sekä vastaanotettu sana r. Mikäli dekoodaaja pystyy laskemaan sanasta r virhevektorin e, virheet voidaan korjata yksinkertaisella vektorien vähennyslaskulla $r - e$. Edellisille esimerkkisanoille, $r - e = 01011 - 00110 = (0, 1, 0, 1, 1) - (0, 0, 1, 1, 0) = (0, 1, 1, 0, 1) = 01101 = c$.

Sanan e *(Hamming)paino* $w(e)$ on sen nollasta eroavien kirjainten määrä. Esimerkiksi sanan $e = 00110$ paino on $w(e) = w(00110) = 2$ eli esimerkkimme virheiden määrä. Lohkokoodi on t virhettä korjaava, mikäli se korjaa kaikki korkeintaan t:n painoiset virheet. Koodi havaitsee t virhettä, mikäli sen avulla voidaan havaita kaikki korkeintaan t:n painoiset virheet. Esimerkiksi kolmen pituinen toistokoodi $C = \{000, 111\}$ on yhden virheen korjaava koodi, sillä yhden bitin muuttuessa tiedonsiirron aikana kaksi muuta eli enemmistö on vielä oikein eikä dekoodaajan tarvitse kuin dekoodata vastaanotettu kolmen pituinen lohko 0:ksi tai 1:ksi enemmistön mukaisesti. Koodi on myös kaksi virhettä havaitseva, sillä kahden bitin muuttuminen ei vielä muuta koodisanaa toiseksi koodisanaksi, joten dekoodaaja tietää virheen tapahtuneen vastaanottaessaan sanan, mikä ei ole koodisana. Huomautettakoon, että koodia voi käyttää ainoastaan joko yhden virheen korjaamiseen tai kahden virheen havaitsemiseen mutta ei molempiin yhtaikaa.

Kahden sanan a ja b (Hamming)etäisyys $d(a, b)$ on niiden kirjainten määrä, joissa a ja b eroavat toisistaan. Esimerkiksi sanojen $c = 01101$ ja $r = 01011$ etäisyys on $d(c, r) = d(01101, 01011) = 2$, sillä sanat eroavat toisistaan kolmannessa ja neljännessä bitissä. Koska sanojen eroavuuskohdat ovat samat kuin niiden erotuksen ($01101 - 01011 =$

00110) nollasta eroavien kirjainten paikat, niin etäisyys on sama kuin erotuksen paino eli $d(a,b) = w(a-b)$. Hammingetäisyys toteuttaa ehdot

$$\begin{cases} d(a,b) \geq 0, d(a,b) = 0 \Longleftrightarrow a = b \\ d(a,b) = d(b,a) \\ d(a,b) \leq d(a,c) + d(c,b) \qquad \text{(kolmioepäyhtälö)} \end{cases}$$

(Eo. ehdot toteuttavaa funktiota kutsutaan yleisesti metriikaksi.) Huomaa myös, että $d(a+x,b+x) = w((a+x)-(b+x)) = w(a-b) = d(a,b)$.

Koodin C *minimietäisyys* $d(C) = \min d(C)$ on pienin eri koodisanojen välisistä etäisyyksistä. Esimerkiksi toistokoodin $C = \{000, 111\}$ minimietäisyys on $d(C) = 3$; eihän koodissa ole kuin kaksi koodisanaa, joiden etäisyys on 3.

Esimerkki 54 *Jos koodin C koodisanat ovat*

$$c_1 = 0000000 \quad c_2 = 1111000 \quad c_3 = 0011111 \quad c_4 = 1100110$$

sen minimietäisyys on $d(C) = 4$, sillä kahden eri koodisanan etäisyys on joko 4 tai 5: $d(c_1, c_2) = 4$, $d(c_1, c_3) = 5$, $d(c1, c4) = 4$, $d(c2, c3) = 5$, $d(c2, c4) = 4$ $d(c3, c4) = 5$.

Koodin minimietäisyys on sen virheenkorjauskyvyn mitta: mitä suurempi minimietäisyys on, sitä enemmän virheitä koodilla voidaan korjata ja havaita.

Lause 55 *Koodi C havaitsee h virhettä, jos ja vain jos sen minimietäisyys $d(C) \geq h + 1$. Koodi korjaa t virhettä, jos ja vain jos sen minimietäisyys $d(C) \geq 2t + 1$.*

Todistus.Mikäli koodin minimietäisyys on $d(C) = \delta$, koodissa on kaksi koodisanaa a ja b, joiden etäisyys on $d(a,b) = \delta$. Koska $d(a-a, b-a) = d(a,b)$, voidaan olettaa, että toinen koodisanoista on nollavektori $(a-a = 0)$ ja toisessa on δ nollasta eroavaa kirjainta. Yksinkertaisuuden vuoksi oletetaan myös, että toisen nollasta eroavat kirjaimet ovat δ ensimmäistä kirjainta:

$$\begin{aligned} a &= 000\ldots0000\ldots0 \\ b &= \underbrace{111\ldots1}_{\delta}000\ldots0 \end{aligned}$$

Koodi havaitsee aina kaikki virheet, jotka eivät muuta kanavaan lähetettyä koodisanaa toiseksi koodisanaksi. Mikäli koodisana a lähetetään kanavaan ja virheet muuttavat juuri sanojen a ja b eroavuuskohdissa olevat δ bittiä (havainnollistuksessamme δ ensimmäistä bittiä 0→1), vastaanottaja saa koodisanan b eikä voi havaita virhettä tapahtuneeksi. Siksi virheenhavaintokyky $h < \delta = d(C)$. Toisaalta jokaisen koodisanaparin etäisyys on vähintään δ eikä siis korkeintaan $(\delta - 1)$:n bitin virheellisyys vielä muuta mitään koodisanaa toiseksi

koodisanaksi. Näin ollen koodi havaitsee kaikki korkeintaan $(\delta - 1)$:n painoiset virheet.

Jotta koodi korjaisi kaikki korkeintaan t-painoiset virheet, on oltava $c+e \neq c'+e'$ olivatpa c ja c' mitä tahansa koodisanoja sekä e ja e' mitä tahansa korkeintaan t-painoisia virheitä. Jos nimittäin $c + e = c' + e'$ jollekin koodisanaparille c, c' ja virheille e, e', niin dekoodaaja ei voi tietää vastaanotetusta sanasta $r = c + e = c' + e'$, onko kanavaan lähetetty koodisana c ja tapahtunut virhe e vai onko kanavaan lähetetty koodisana c' ja tapahtunut virhe e'. Erityisesti koodisanaparille a, b, joiden etäisyys $d(a, b)$ on koodin minimietäisyys δ, nähdään havainnollistuksesta

$$\begin{array}{rcl} a = 000\ldots0000\ldots0 & \rightarrow & \overbrace{111\ldots1}^{s\ virhettä}\overbrace{000\ldots000\ 000\ldots000}^{n-s\ oikein} \\ b = \underbrace{111\ldots1}_{\delta}000\ldots0 & \rightarrow & \underbrace{111\ldots1}_{s\ oikein}\underbrace{000\ldots000}_{\delta-s\ virhettä}\ \underbrace{000\ldots000}_{n-\delta\ oikein} \end{array}$$

että jos sanan a siirrossa tapahtuu sanojen a ja b eroavuuskohdissa s virhettä ja sanan b siirrossa taas loput $\delta - s$ eroavuuskohtaa ovat virheellisiä, niin tuloksena on sama vastaanotettu sana eikä dekoodaaja pysty korjaamaan molempia virheitä. Havainnollistuksen mukaisten eroavuuskohdissa tapahtuvien kahden virheen yhteispaino voi siis olla korkeintaan $\delta - 1$, joten virheenkorjauskykyä t rajoittaa epäyhtälö $t + t \leq \delta - 1$ eli $t \leq (\delta - 1)/2$. Havainnollistuksesta on myös selvää, että näin monta virhettä voidaan todella korjata. Jos nimittäin koodisanan a siirron aikana tapahtuu korkeintaan $t \leq (\delta - 1)/2$ virhettä, niin vastaanotettu sana$r = a + e$ eroaa koodisanasta b vähintään $\delta - t \geq \delta - (\delta - 1)/2 > (\delta - 1)/2 \geq t$ kohdassa ja dekoodausperiaatteella "dekoodaa lähimmäksi koodisanaksi" vastaanotettu sana r dekoodataan koodisanaksi a, jolloin virhe tulee korjattua.

Koska esimerkin 54 koodin $\mathcal{C} = \{0000000, 1111000, 0011111, 1100110\}$ minimietäisyys on $d(C) = 4$, sillä voidaan korjata kaikki yhden painoiset virheet tai sitä voidaan käyttää kaikkien korkeintaan kolmen painoisten virheiden havaitsemiseen.

Lauseen todistuksen loppuosa on yleisemmin esitettynä seuraava. Oletetaan, että koodisanan c siirrossa tapahtuu korkeintaan $t \leq (\delta - 1)/2$ virhettä, jolloin vastaanotettu sana on $r = c + e$, missä e on korkeintaan t-painoinen virhe. Mikäli c' on jokin toinen koodisana, niin kolmioepäyhtälön mukaan

$$\begin{array}{rcl} d(c', r) & = & d(c', c + e) \geq d(c', c) - d(c, c + e) \geq \delta - t \geq \delta - (\delta - 1)/2 \\ & = & (\delta - 1)/2 + 1 > (\delta - 1)/2 \geq t \geq d(c, c + e) = d(c, r) \end{array}$$

joten vastaanotettu sana $r = c + e$ on lähempänä koodisanaa c kuin mitään muuta koodisanaa c' ja näin virhe e voidaan korjata. Tämän mukaisesti lause voidaan esittää myös seuraavassa muodossa.

Lause 56 *Koodi $\mathcal{C} \subseteq \mathbb{Z}_2^n$ korjaa t virhettä, jos ja vain jos koodisanakeskiset t-säteiset pallot*

$$S_t(c) = \{r \in \mathbb{Z}_2^n \mid d(r, c) \leq t\} \quad c \in \mathcal{C}$$

ovat alkiovieraita.

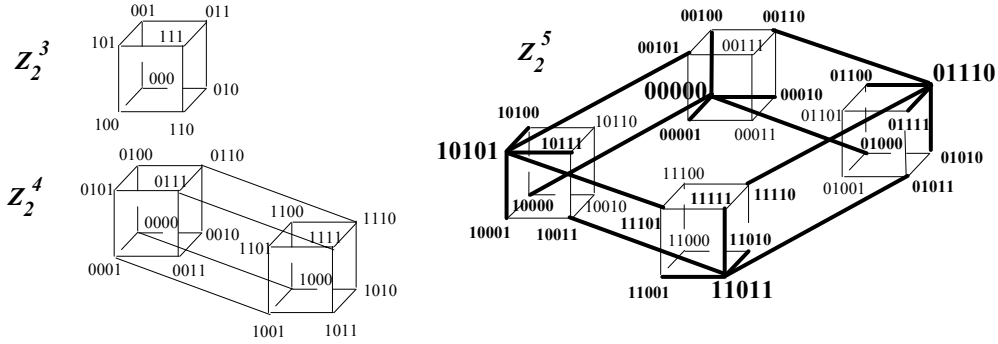
Esimerkki 57 *.Kun koodisanat ovat* $c_1 = 00000$*,* $c_2 = 01110$*,* $c_3 = 10101$*,* $c_4 = 11011$*, niin koodisanakeskiset 1-säteiset pallot ovat*

$$S_1(c_1) = \{00000, 00001, 00010, 00100, 01000, 10000\}$$
$$S_1(c_2) = \{01110, 01111, 01100, 01010, 00110, 11110\}$$
$$S_1(c_3) = \{10101, 10100, 10111, 10001, 11101, 00101\}$$
$$S_1(c_4) = \{11011, 11010, 11001, 11111, 10011, 01011\}$$

ja koska näillä ei ole yhteisiä alkioita, niin koodi korjaa yhden virheen. Saman toteaa helpommin koodin minimietäisyydestä $d(C) = 3$*.*

Diskreetissä avaruudessa Z_2^5 pallot $S_1(c_i)$ eivät tietenkään muistuta lainkaan tuttuja palloja, vaan enemmänkin koordinaattiakselien suuntaisten janojen muodostamia sädekimppuja. Viisiulotteisen avaruuden havainnollistus tarvinnee projektiivista geometriaa. Yritetäänpä ilman.

Kolmiulotteisen avaruuden $\mathbb{Z}_2^3$ voi kuvata kuutiona, jonka 8 kärkeä ovat avaruuden 8 kolmen mittaista sanaa. Kun tämän kopioi neljännen ulottuvuuden suuntaan 1000, saa avaruuden $\mathbb{Z}_2^4$ havainnollistukseksi kaksi kuutiota, joiden 16 kärkipistettä ovat 16 neljän pitusta sanaa. Kun tämän neliulotteisen avaruuden vielä kopioi viidennen ulottuvuuden suuntaan 10000, avaruuden $\mathbb{Z}_2^5$ havainnollistuksena on 4 kuutiota, joissa on yhteensä 32 kärkipistettä, kukin viiden pituinen sana. Havainnollistuksessa viiva kahden sanan välillä osoittaa sanojen etäisyyden olevan 1, mutta selvyyden vuoksi kaikkia ei ole merkitty näkyviin. Pallot on esitetty lihavoituna.



Olkoon n-pituisen binäärikoodin $\mathcal{C}$ koodisanojen määrä K. Sellaisia sanoja, joiden etäisyys annetusta koodisanasta on i eli jotka eroavat kyseisestä sanasta i:ssä kohdassa, on $\binom{n}{i}$ kappaletta, joten yhdessä t-säteisessä pallossa on

$$|S_t(c)| = \sum_{i=0}^{t} \binom{n}{i}$$

sanaa. Koska t virhettä korjaavalle koodille koodisanakeskiset pallot ovat alkiovieraita, niissä on yhteensä

$$K \sum_{i=0}^{t} \binom{n}{i}$$

sanaa ja koska n:n pituisia sanoja on kaikkiaan 2^n kappaletta, on oltava

$$K \sum_{i=0}^{t} \binom{n}{i} \leq 2^n$$

Tämä *Hammingraja* antaa siis koodisanojen määrälle K ylärajan virheenkorjauskyvyn t suhteen.

Esimerkki 58 *Kun $n = 10$ ja $t = 3$, saadaan Hammingrajan avulla K:lle yläraja 5.8. Täten koodissa on korkeintaan 5 sanaa.*

Toisaalta suurimman t virhettä korjaavan koodin koodisanojen määrä K täyttää *Gilbertrajan*

$$K \sum_{i=0}^{2t} \binom{n}{i} \geq 2^n$$

Jos nimittäin olisi

$$K \sum_{i=0}^{2t} \binom{n}{i} < 2^n$$

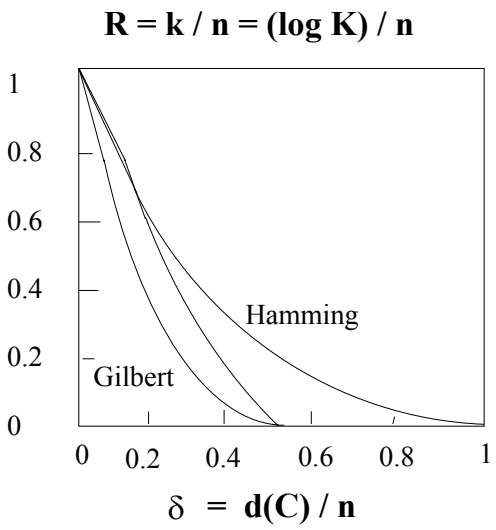
niin koodisanakeskiset $2t$-säteiset pallot eivät peittäisi koko avaruutta $\mathbb{Z}_2^n$ ja näin olisi olemassa jokin sana, jonka etäisyys kaikista koodisanoista olisi vähintään $2t+1$ ja lisäämällä tämä sana koodiin saataisiin koodi, jossa olisi $K+1$ koodisanaa ja joka olisi myös t virhettä korjaava.

Esimerkki 59 *Valitaan kuten edellä $n = 10$ ja $t = 3$. Gilbertrajaksi K:lle saadaan $1, 2$. Täten koodiin voidaan valita ainakin 2 sanaa. Yhdistämällä esimerkit 58 ja 59 saadaan siis K aina rajattua välille 2:sta 5:een, kun käytetään 10-pituista koodia ja halutaan koodin korjaavan 5 virhettä.*

Huom. Virheenkorjauskyky saadaan miten suureksi tahansa valitsemalla riittävän monta kertaa toistava toistokoodi. Samalla informaatiosuhde lähestyy nollaa. Toisaalta virheenkorjaus voidaan myös täysin unohtaa, jolloin informaatiosuhde $= 1$.

Koodin informaatiosuhde $R = k/n = (\log K)/n$ on edellä todistettujen Gilbert- ja Hammingrajojen välissä. Pitkille koodeille (pituus $n \to \infty$) nämä rajat ovat oheisen kuvan mukaiset; kuvassa on myös toinen, huomattavasti Hammingrajaa parempi yläraja (sisältää eräitä ehtoja). Vaikka parhaat asymptoottiset ala- ja ylärajat ovatkin jo melko lähellä toisiaan, hyvien koodien konstruointi on edelleenkin suuri avoin

ongelma.



Lineaariset koodit

Koodi $\mathcal{C} \subseteq \mathbb{Z}_2^n$ on lineaarinen, mikäli se on vektoriavaruuden $\mathbb{Z}_2^n$ aliavaruus. Aliavaruuskriteerin perusteella koodi on lineaarinen, jos $c - c'$ on koodisana aina, kun c ja c' ovat koodisanoja. Huomaa, että nollavektori $0 = 000\ldots0$ on aina lineaarisen koodin koodisana. Lineaariselle koodille kahden koodisanan etäisyys $d(c, c') = w(c - c')$ on aina sama kuin koodisanan $c - c'$ paino, joten minimietäisyyskin on helppo laskea: koodin minimietäisyys $d(C)$ on pienin nollasta eroavien koodisanojen painoista.

Esimerkki 60 *Koodi* $\mathcal{C} = \{c_1 = 00000000, c_2 = 11111000, c_3 = 00011111, c_4 = 11100111\}$ *on lineaarinen, sillä koodisanojen erotukset ovat (muista* $-1 = 1$*)*

$$\begin{aligned}
c_1 - c_2 &= 00000000 - 11111000 = 11111000 &&= c_2 \in \mathcal{C} \\
c_1 - c_3 &= 00000000 - 00011111 = 00011111 &&= c_3 \in \mathcal{C} \\
c_1 - c_4 &= 00000000 - 11100111 = 11100111 &&= c_4 \in \mathcal{C} \\
c_2 - c_3 &= 11111000 - 00011111 = 11100111 &&= c_4 \in \mathcal{C} \\
c_2 - c_4 &= 11111000 - 11100111 = 00011111 &&= c_3 \in \mathcal{C} \\
c_3 - c_4 &= 00011111 - 11100111 = 11111000 &&= c_2 \in \mathcal{C}
\end{aligned}$$

Koodin minimietäisyys on $d(C) = 5$*, sillä nollasta eroavat koodisanat* c_2 *ja* c_3 *ovat viiden painoisia sekä* c_4 *on kuuden painoinen. Koodi on siis kaksi virhettä korjaava. Koska koodisanoja on neljä, sen informaatiosuhde on* $R = (\log_2 4)/8 = 2/8$*. Koodauksena voi olla esimerkiksi*

$$00 \to c_1, 10 \to c_2, 01 \to c_3, 11 \to c_4$$

Koska äärellisdimensioisen vektoriavaruuden aliavaruudella on aina kanta, niin lineaarisella koodilla $\mathcal{C}$ on kanta $\{c_1, c_2, \ldots, c_k\}$, joka virittää koko koodin: $\mathcal{C} = L(c_1, c_2, \ldots, c_k)$ eli jokaisella koodisanalla $c \in \mathcal{C}$ on yksikäsitteinen kantaesitys

$$c = v_1c_1 + v_2c_2 + \cdots + v_kc_k$$

Tämä kantaesitys antaa myös luonnollisen koodauksen: viesti $v = v_1v_2 \ldots v_k$ koodataan koodisanaksi $c = v_1c_1 + v_2c_2 + \cdots + v_kc_k$. Huomaa, että kantavektoreiden määrä k on koodin C dimensio ja että koodissa on $K = 2^k$ koodisanaa, jolloin sen informaatiosuhde on $R = (\log_2 K)/n = (\log_2 2^k)/n = k/n$. Koodin generoijamatriisi G on matriisi, jonka rivit ovat koodin kantavektorit. Tällöin viestin v koodaus voidaan laskea matriisikertolaskuna $c = vG$, mikä on nopea suorittaa integroidulla piirillä.

Esimerkki 61 *Esimerkin 59 koodin* $\mathcal{C} = \{c_1 = 00000000, c_2 = 11111000, c_3 = 00011111, c_4 = 11100111\}$ *koodisanat* $c_2 = 11111000$ *ja* $c_3 = 00011111$ *voidaan valita kannaksi, joten generoijamatriisiksi voidaan valita*

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Viestien $v_1 = 00$, $v_2 = 10$, $v_3 = 01$, $v_4 = 11$ koodaukset ovat

$$\begin{aligned} c_1 = v_1G = 00 &\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} = 00000000 \\ c_2 = v_2G = 10 &\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} = 11111000 \\ c_3 = v_3G = 01 &\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} = 00011111 \\ c_4 = v_4G = 11 &\begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} = 11100111 \end{aligned}$$

Matriisin G redusointi ns. porrasmatriisiksi ja sarakkeiden järjestyksen vaihto antaa matriisin G', mikä on muotoa

$$G' = \left[I_k | P_{k\times(n-k)}\right]$$

missä I_k on $k \times k$ -identiteettimatriisi ja $P_{k\times(n-k)}$ $k \times (n-k)$-matriisi. Matriisin rivioperaatiot suorittavat kannanvaihdon muuttamatta matriisin rivien määräämää lineaarista aliavaruutta, ns. riviavaruutta, joten rivioperaatiot muuttavat ainoastaan koodausta eivätkä itse koodisanoja. Sarakkeiden järjestyksen vaihto puolestaan vastaa koodisanojen kirjainten järjestyksen vaihtoa. Lopputuloksena on täysin ekvivalentti koodi, jolla on sama virheenkorjauskyky ja joka on systemaattinen, mikä merkitsee, että jokaista viestiä v vastaava koodisana on muotoa $c = vp$, missä p on jokin $(n-k)$:n pituinen sana (tarkistusbitit).

Esimerkki 62 *Muutetaan generoijamatriisi*

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

systemaattiseen muotoon:

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \sim \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 \end{bmatrix} \sim \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} = G'$$

(Vaihdetaan sarakkeiden järjestystä ja lasketaan yhteen rivejä.) Matriisit G ja G' ovat saman koodin $\mathcal{C}$ generoijamatriiseja, ero on ainoastaan koodauksessa:

$$v = 1010 \rightarrow c = vG = 1010 \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} = 00100111$$

(missä viesti v?)

$$v = 1010 \rightarrow c = vG = 1010 \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} = v\underbrace{1010}p\underbrace{1001}$$

Esimerkki 63 *Esimerkin 61 generoijamatriisi*

$$G = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

saadaan systemaattiseen muotoon yksinkertaisesti vaihtamalla toinen ja kuudes sarake keskenään:

$$G = \left[\begin{array}{cc|cccccc} 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{array}\right] = [I_2 \mid P_{2\times 6}]$$

Nyt koodisanat ovat

$$c_1' = v_1G' = 00 \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} = 00000000$$

$$c_2' = v_2G' = 10 \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} = 10111100$$

$$c_3' = v_3G' = 01 \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} = 01011011$$

$$c_4' = v_4G' = 11 \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \end{bmatrix} = 11100111$$

Nämä eroavat alkuperäisen koodin koodisanoista ainoastaan kirjainten järjestyksen suhteen:

Kuten reaaliavaruudessa $\mathbb{R}^n$ myös avaruudessa $\mathbb{Z}_2^n$ on määritelty pistetulo ja kohtisuoruus: jos $a = a_1a_2\dots a_n$ sekä $b = b_1b_2\dots b_n$, niin $a\cdot b = a_1b_1+a_2b_2+\cdots+a_nb_n$ (laskut modulo 2) ja $a \perp b$, mikäli $a\cdot b = 0$. Aliavaruutena lineaarisella koodilla $\mathcal{C}$ on ortogonaalikomplementti

$$\mathcal{C}^\perp = \{x \in \mathbb{Z}_2^n \mid c\cdot x = 0 \text{ aina, kun } c \in \mathcal{C}\}$$

Jos koodin $\mathcal{C}$ dimensio on k, niin ortogonaalikomplementin $C^\perp$ dimensio on $n-k$ ja sillä on kanta $\{h_1, h_2, \dots, h_{n-k}\}$. Matriisi H, jonka vaakarivit ovat ortogonaalikomplementin $\mathcal{C}^\perp$ kantavektorit $h_1, h_2, \dots, h_{n-k}$, on koodin $\mathcal{C}$ *(pariteetin)tarkistusmatriisi.*

Koodisanalle c vektorin cH^T alkiot ovat pistetulot $c\cdot h_1, c\cdot h_2, \dots, c\cdot h_{n-k}$mitkä kaikki ovat nollia. Sana c on siis koodisana tarkalleen silloin, kun se toteuttaa (pariteetin)tarkistusyhtälöt

$$cH^T = 0$$

Erityisesti generoijamatriisin G riveinä olevat koodin kantavektorit toteuttavat tarkistusyhtälöt, joten $GH^T = 0$. Kun $k\times n$-generoijamatriisi G on annettu, tarkistusmatriisiksi H voidaan valita sellainen $(n-k)\times n$-matriisi H, joka toteuttaa yhtälön $GH^T = 0$ ja jonka pystyrivejä ei voida esittää toisten pystyrivien summana muutoin kuin ottamalla mukaan kaikki pystyrivit. Erikoisesti, mikäli G on systemaattinen eli muotoa $G = \left[I_k|P_{k\times(n-k)}\right]$, niin tarkistusmatriisiksi voidaan valita $H = \left[-P^T_{k\times(n-k)} \,|\, I_{n-k}\right]$, sillä tällöin

$$\begin{aligned} GH^T &= \left[I_k|P_{k\times(n-k)}\right]\left[-P^T_{k\times(n-k)}\,|I_{n-k}\right]^T = \left[I_k|P_{k\times(n-k)}\right]\begin{bmatrix} -P_{k\times(n-k)} \\ I_{n-k}\end{bmatrix} \\ &= \left[-I_kP_{k\times(n-k)} + P_{k\times(n-k)}I_{n-k}\right] = \left[-P_{k\times(n-k)} + P_{k\times(n-k)}\right] = 0. \end{aligned}$$

Jos taas tarkistusmatriisi H on annettu, voidaan generoijamatriisi G määrätä vastaavasti.

Esimerkki 64 *Olkoon C matriisin*

$$G = \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right] = [I_3\,|P]$$

generoima seitsemän pituinen systemaattinen koodi. Tällöin tarkistusmatriisi on

$$H = \left[\begin{array}{ccc|cccc} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right] = \left[P^T\,|I_4\right]$$

Viesti $v = 101$ *koodataan koodisanaksi*

$$c = vG = 101 \left[\begin{array}{ccc|cccc} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right] = 101\,|1010 = vp$$

Dekoodaaja voi tarkistaa vastaanotetun sanan r *virheettömyyden tarkastamalla pariteetintarkistusyhtälöt* $rH^T = 0$ *. Mikäli siirrossa ei tapahdu virheitä on* $r = c = 1011010$ *ja*

$$rH^T = 101 \mid 1010 \left[\begin{array}{ccc|cccc} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right]^T = 0000 = 0$$

Koska tarkistusyhtälöt ovat voimassa, vastaanottaja tietää, että r *on koodisana. Jos tiedonsiirron aikana tapahtuu virhe, esimerkiksi* $c = 1011010 \rightarrow c + e = 1011010 + 0101000 = 1110010 = r$*, dekoodaajan suorittama lasku*

$$rH^T = 111 \mid 0010 \left[\begin{array}{ccc|cccc} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{array}\right]^T = 0011 \neq 0$$

osoittaa, etteivät tarkistusyhtälöt toteudu. Näin dekoodaaja tietää, että vastaanotettu sana r *ei olekaan koodisana ja havaitsee virheen tapahtuneen.*

Vektoria $s = rH^T$, eo. esimerkissä 0011, sanotaan vastaanotetun sanan r syndromiksi (syndrome = enne, oire; tässä vaikkapa virheen tunnusmerkit). Sana r on koodisana tarkalleen silloin, kun sen syndromi on 0. Yleisesti vastaanotetun sanan $r = c + e$ syndromi on sama kuin tapahtuneen virheen e syndromi, sillä

$$s = rH^T = (c + e)H^T = cH^T + eH^T = 0 + eH^T = eH^T$$

koska $cH^T = 0$. Koodin $\mathcal{C} \subseteq Z_2^n$ yhden sivuluokan $e + \mathcal{C} = \{r \in \mathbb{Z}_2^n : r = e + c, c \in \mathcal{C}\}$ sanoilla on siis sama syndromi. Toisaalta jos sanoilla r ja r' on sama syndromi eli $rH^T = rH^T$, niin $(r - r)H^T = 0$, joten $r - r'$ on koodisana eli $r - r' = c \in \mathcal{C}$ eli $r = r' + c \in r' + C$ ja siis r ja r' ovat samassa sivuluokassa. Dekoodaus voidaan suorittaa käyttämällä syndromitaulukkoa, jossa kutakin syndromia s vastaavaa sivuluokkaa $e + \mathcal{C}$ edustaa painoltaan pienin sivuluokan sana e. Laskettuaan vastaanotetun sanan r syndromin s dekoodaaja etsii sen taulukosta ja korjaa sivuluokan edustajaksi valitun virheen e.

Esimerkki 65 *Kun koodin generoijamatriisi* G *ja tarkistusmatriisi* H *ovat*

$$G = \left[\begin{array}{ccccc} 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \end{array}\right] \quad H = \left[\begin{array}{ccccc} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

niin koodi C ja sen sivuluokat $e + C$ sekä niiden syndromit $s = eH^T$ ovat

$$\begin{array}{rll} C = & 00000 + C = \{00000, 10101, 01011, 11110\} & s = 000 \\ & 10000 + C = \{10000, 00101, 11011, 01110\} & s = 101 \\ & 01000 + C = \{01000, 11101, 00011, 10110\} & s = 011 \\ & 00100 + C = \{00100, 10001, 01111, 11010\} & s = 100 \\ & 00010 + C = \{00010, 10111, 01001, 11100\} & s = 010 \\ & 00001 + C = \{00001, 10100, 01010, 11111\} & s = 001 \\ & 11000 + C = \{11000, 01101, 10011, 00110\} & s = 110 \\ & 10010 + C = \{10010, 00111, 11001, 01100\} & s = 111 \end{array}$$

Nämä saadaan kokeilemalla C:hen kuulumattomia sanoja. Huomaa, että sivuluokilla ei ole mitään yhteisiä sanoja. Esimerkiksi vastaanotetun sanan $r = 11010$ syndromi on

$$s = rH^T = 11010 \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = 100$$

joten edeltävän syndromitaulukon mukaisesti dekoodaaja suorittaa korjauksen

$$r = 11010 \rightarrow r - e = 11010 - 00100 = 11110 = c \in C$$

Huomaa, että sivuluokan edustajaksi voitaisiin valita mikä tahansa sivuluokan sana, esimerkiksi

$$11010 + C = \{11010, 01111, 10001, 00100\} = 00100 + C$$

Sivuluokkien edustajiksi valitaan ne virheet, jotka koodilla korjataan, ja siksi edustajiksi valitaan painoltaan pienimmät edustajat. Kaikissa tapauksissa ei asia ole kuitenkaan yksikäsitteinen. Vertaa eo. esimerkin toiseksi viimeisen rivin määräämä sivuluokka.

Huomaa myös, että jos koodin dimensio on k, niin kussakin sivuluokassa on 2^k $(= 2^2 = 4)$ sanaa ja sivuluokkia on kaikkiaan 2^{n-k} $(= 2^{5-2} = 8)$ kappaletta, koska kaikki sivuluokat yhdessä muodostavat koko 2^n -alkioisen avaruuden, ts. kaikki alkiot / sivuluokan alkioitten määrä $= 2^n/2^k = 2^{n-k} =$ sivuluokkien lukumäärä.

Vaikka syndromiluettelon käyttäminen on kaikille koodeille soveltuva dekoodausmenetelmä, se ei kuitenkaan ole varsinaisesti käyttökelpoinen, sillä pitkillä koodeilla luettelon tallentaminen vaatii paljon muistitilaa ja taulukosta etsiminen on hidasta. Koodiluokilla on niille ominaiset rakenteet, jotka mahdollistavat virhevektorin laskemisen vastaanotetusta sanasta nopeammin kuin syndromitaulukosta etsimällä.

Tarkistusmatriisia voi käyttää myös koodin minimietäisyyden määräämiseen. Matriisituloa cH^T laskettaessa lasketaan yhteen matriisin

H sarakkeita $s_1, s_2, \ldots, s_n$ (transpoosin H^T rivejä). Koska sana c on koodisana tarkalleen silloin, kun $cH^T = 0$, niin matriisin H sarakkeiden lineaarikombinaatio $c_1s_1 + c_2s_2 + \cdots + c_ns_n$ voi olla nolla ainoastaan, kun $c = 0$ tai siinä on vähintään koodin minimietäisyyden d verran nollasta eroavia kertoimia c_i. Tämä havainto antaa seuraavan lauseen.

Lause 66 *Lineaarisen koodin C minimietäisyys on d tarkalleen silloin, kun sen tarkistusmatriisissa on d saraketta, joiden summa on nollavektori, mutta mikään korkeintaan $(d-1)$:n sarakkeen summa ei ole nollavektori.*

Koska tarkistusmatriisin aste on $n - k$, lause antaa minimietäisyydelle ylärajan $d \leq n - k$.

Hammingkoodit

Hammingkoodi on lineaarinen koodi, jonka tarkistusmatriisin H sarakkeina ovat kaikki nollasta eroavat r:n pituiset binäärivektorit. Koodin pituus on näiden sarakkeiden määrä $n = 2^r - 1$, tarkistusyhtälöiden eli koodisanassa olevien tarkistusbittien määrä on matriisin H rivien määrä $n - k = r$ ja koodisanan viestibittien määrä on siis $k = n - r = 2^r - 1 - r$. Koodilla on näin ollen suuri informaatiosuhde $R = k/n = (2^r - 1 - r)/(2^r - 1) = 1 - r/(2^r - 1)$ ja viestin siirto kanavassa on nopeaa. Toisaalta koodi kykenee korjaamaan ainoastaan yhden virheen, joten se on käyttökelpoinen vain hyvissä kanavissa. Tämä nähdään Lauseen 66 perusteella, sillä H:ssa on sarakkeiden 1000..., 0100... ja 1100... summa nolla. Täten $d = 3$ ja virheenkorjauskyky = 1.

Esimerkiksi kun $r = 4$, koodin pituus on $n = 2^4 - 1 = 15$ ja $k = n - r = 15 - 4 = 11$, joten informaatiosuhde on $R = 11/15$ eli koodilohkon 15:sta bitistä on 11 viestibittiä ja loput 4 tarkistusbittejä. Tarkistusmatriisi H ja generoijamatriisi G ovat

$$H = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Koodin minimietäisyys lasketaan Lauseen 66 mukaisesti tutkimalla tarkistusmatriisin H sarakkeiden lineaarista riippuvuutta. Koska sarakkeet ovat erit, on laskettava vähintään kolme saraketta yhteen, jotta tulos olisi nollavektori 0. Näin ollen jokaisen nollasta eroavan koodisanan paino on vähintään kolme. Toisaalta on helppo löytää kolme saraketta, joiden summa on nollavektori, jolloin koodissa on koodisana, jonka paino on kolme. Esimerkiksi sanalle $c = 000000100000101$ on

$$cH^T = 000000100000101 \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

joten $c = 000000100000101$ on koodisana ja koska sen paino on kolme, on koodin minimietäisyyskin kolme ja koodi kykenee korjaamaan vain yhden virheen.

Kun siirrossa tapahtuu yksi virhe, esimerkiksi $c = 000000100000101$ $\rightarrow 001000100000101 = r$, se voidaan korjata laskemalla syndromi $s =$

rH^T :

$$s = rH^T = 001000100000101 \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = 1011$$

Koska matriisin H^T riveinä (matriisin H sarakkeina) ovat kaikki nollasta eroavat sanat, virheen syndromi s on aina jokin niistä. Mikäli tapahtuu vain yksi virhe, on s virhekohtaa vastaava matriisin H^T rivi (matriisin H sarake), joten virhe saadaan paikannettua ja voidaan korjata. Esimerkin syndromi $s = 1011$ on matriisin H^T kolmannella rivillä, joten dekoodaaja korjaa vastaanotetun sanan kolmannen bitin: $r = 001000100000101 \rightarrow 000000100000101$.

Kun matriisin H sarakkeet tulkitaan lukuina 1 - 15 ($= 2^r - 1$) ja järjestetään suuruusjärjestykseen, niin syndromi s on binäärilukuna juuri virhekohdan indeksi vastaanotetussa sanassa. Todellisessa dekoodaajan toteutuksessa tarkistusmatriisin H sarakkeet järjestetään siten, että virheellinen bitti saadaan korjattua ilman syndromitaulukkoa yksinkertaisesti dekoodaajan eri siirtorekisterien tahdistuksen avulla.

Tasaiset koodit

Hammingkoodin ortogonaalikomplementtia sanotaan tasaiseksi koodiksi. Tasaisen koodin tarkistusmatriisina on siis Hammingkoodin generoijamatriisi ja generoijamatriisina taas on Hammingkoodin tarkistusmatriisi. Generoijamatriisin sarakkeina ovat näin ollen kaikki k:n pituiset nollasta eroavat vektorit. Esimerkiksi 15-pituisen tasaisen koodin generoijamatriisi G ja tarkistusmatriisi H ovat (oleellisesti systemaattisessa muodossa)

$$G = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \end{bmatrix}$$

Koodin pituus on $n = 2^k - 1$ ($n = 2^4 - 1 = 15$) ja sen informaatiosuhde on $R = k/n = k/(2^k - 1)$ ($R = 4/15$). Informaatiosuhde on siis pieni ja viestin siirto hitaampaa kuin Hammingkoodilla. Sen sijaan koodin virheenkorjauskyky on toistokoodien jälkeen suurin mahdollinen: koodin minimietäisyys on $\delta = d(C) = 2^{k-1} = (n+1)/2 (\delta = 2^{4-1} = 8)$ ja se kykenee korjaamaan $t = (\delta - 1)/2 = (n-1)/4$ virhettä ($t = (8-1)/2 = 3.5$ eli $t = 3$).

Esimerkki 67 *Valinnalla $k = 3$ saadaan esimerkin 64 koodi, jonka pituus on $n = 2^3 - 1 = 7$, informaatiosuhde on $R = 3/7$, minimietäisyys on $\delta = (n+1)/2 = 4$ ja virheenkorjauskyky on $t = 1$. Systemaattiset generoijamatriisi G ja tarkistusmatriisi H ovat*

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \quad H = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Vastaanotetun sanan $r = vp = v_1v_2v_3p_1p_2p_3p_4$ syndromi $s = rH^T = s_1s_2s_3s_4$ on

$$s_1s_2s_3s_4 = s = rH^T = v_1v_2v_3p_1p_2p_3p_4 \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \left[\; v_1 + v_2 + p_1, \quad v_1 + v_3 + p_2, \quad v_2 + v_3 + p_3, \quad v_1 + v_2 + v_3 + p_4 \; \right]$$

eli

$$\begin{cases} v_1 + & v_2 + & & p_1 & & & = s_1 \\ v_1 + & & v_3 + & & p_2 & & = s_2 \\ & v_2 + & v_3 + & & & p_3 & = s_3 \\ v_1 + & v_2 + & v_3 + & & & p_4 & = s_4 \end{cases}$$

Näin ollen tarkistusbitit p_1, p_2, p_3, p_4 tarkistavat kukin osan viestibiteistä v_1, v_2, v_3. Koodin dekoodaus perustuu tämän yhtälöryhmän erikoiseen muotoon. Laskettuaan vastaanotetun sanan r syndromin $s = rH^T = s_1s_2s_3s_4$ dekoodaaja tietää yhtälöryhmän oikealla puolella olevat vakiot s_1, s_2, s_3, s_4, joista sen on pystyttävä laskemaan tapahtunut virhe. Mikäli koodisanan $c = vp = v_1v_2v_3p_1p_2p_3p_4$ siirrossa on tapahtunut yksi virhe, se on voinut tapahtua joko viestiosassa $v = v_1v_2v_3$ tai tarkistusosassa $p = p_1p_2p_3p_4$. Jos virhe on tarkistusosassa p, se vaikuttaa ainoastaan yhteen yllä olevista yhtälöistä eli yhden yhtälön oikealla puolella olevassa syndromissa s on tarkalleen yksi 1 ja sen sijainnista dekoodaaja tietää, missä tarkistusbitissä virhe on. Jos taas virhe tapahtuu viestiosassa v, se vaikuttaa kolmeen yhtälöön ja, kuten yhtälöryhmän muodosta nähdään, syndromissa s on tarkalleen yksi 0 ja tässäkin tapauksessa sen sijainti kertoo, missä viestibitissä virhe on. Virheen paikantaminen ja korjaus voidaan virtapiiritasolla toteuttaa enemmistölogiikalla.
Esimerkiksi vastaanotettu sana $r = 1011110 = v_1v_2v_3p_1p_2p_3p_4$ antaa tarkistusarvot

$$\begin{cases} v_1+ & v_2+ & & p_1 & & & = 1+0+1 = 0 \\ v_1+ & & v_3+ & & p_2 & & = 1+1+1 = 1 \\ & v_2+ & v_3+ & & & p_3 & = 0+1+1 = 0 \\ v_1+ & v_2+ & v_3+ & & & p_4 & = 1+0+1+0 = 0 \end{cases}$$

joten virhe on toisessa tarkistusbitissä p_2 (syndromin ainoa 1 on toinen bitti) ja näin ollen dekoodaus $r = 1011110 \rightarrow 1011010 = c$ korjaa virheen (olettaen, että todella vain yksi virhe on tapahtunut).
Vastaavasti $r = 1001010$ antaa

$$\begin{cases} v_1+ & v_2+ & & p_1 & & & = 1+0+1 = 0 \\ v_1+ & & v_3+ & & p_2 & & = 1+0+0 = 1 \\ & v_2+ & v_3+ & & & p_3 & = 0+0+1 = 1 \\ v_1+ & v_2+ & v_3+ & & & p_4 & = 1+0+0+0 = 1 \end{cases}$$

joten virhe on kolmannessa viestibitissä v_3 (syndromin ainoa 0 on ensimmäisessä bitissä ja v_3 puuttuu ensimmäisestä yhtälöstä) ja korjataan dekoodauksella $r = 1001010 \rightarrow 1011010 = c$.