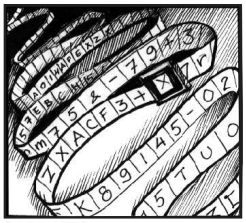


Lukuteoriaa ja salakirjoitusta, osa 2

Heikki Apiola
Matematiikan laitos, Teknillinen korkeakoulu

Virittelyksi

Kirjoitus on jatkoa numerossa 3/2007 olleelle esiosalle (`solmu.math.helsinki.fi/2007/3/apiola.pdf`), jossa esitellään lukuteoreettisia perustyökaluja tämän varsinaisesti salakirjoitukseen pureutuvan osan 2 tarpeisiin. Viittaan ykkösosaan roomalaisella I:llä.
Käytän myös joissakin kohdissa vektori- ja matriisitermejä. *Vektori* tarkoittaa yksinkertaisesti lukujonoa, lukujen järjestettyä listaa. Matriisi on lukutaulukko, jossa on samanpituisia vektoreita allekkain. Voit palauttaa vektorin aiempiin vektorimielikuviisi vaikkapa Solmun numerossa 1/2007 olleen kirjoituksen (`solmu.math.helsinki.fi/2007/1/apiola.pdf`) alkua silmäilemällä.

Kryptologia - salaustiede

Tiedon salaamisen/turvaamisen tarvetta on esiintynyt ihmiskunnan historiassa vuosituhansien ajan. Vanhimmat löydökset ovat n. 4500 vuoden takaa Egyptin vanhasta kuningaskunnasta.

Nimitykset *kryptologia* ja *kryptografia* on johdettu kreikankielen sanasta *kryptós*, salattu, salainen. *Kryptologia* näyttää vakiintuneen yleistermiksi, joka sisältää osinaan *kryptografian* eli salaamisteknisten menetelmien suunnittelutieteen ja *kryptoanalyysin*, jonka piiriin kuuluvat menetelmien luotettavuuden analysoiminen, heikkouksien paljastaminen, salakoodien murtaminen jne. Niin sankarit, konnat kuin viileän objektiiviset menetelmien tutkijat ja testaajat työskentelevät *kryptoanalyytikon* velvoittavan nimikkeen alla.

Lähihistorian dramaattisimpiin tapahtumiin kuuluvat liittoutuneiden onnistuneet saksalaisten *Enigma*-salakirjoituskoneen sanomien murtamiset (kts. [Wiki]). Tämä pohjautui puolalaisen matemaatikon Marian Rejewskin vuonna 1932 tekemiin keksintöihin. Hän onnistui algebrallista tekniikkaa käyttäen pääsemään Enigma-salauksen jäljille. Hänen ryhmänsä luovutti tuloksensa vuonna 1939 ranskalaisille ja englantilaisille kryptoanalyytikoille. Brittien Saksan merivoimiin kohdistuvaa kryptoanalyyttista toimintaa johti matemaatikko ja teoreettisen tietojenkäsittelytieteen isä, *Alan Turing*, jonka ryhmä onnistui joulukuussa 1940 murtamaan saksalaisten Enigma-koodin. Joissakin lähteissä esiintyy arvioita, joiden mukaan pelkästään tämän onnistumisen ansiosta toinen maailmansota lyheni useilla vuosilla.

Pieni englantilainen paikkakunta, jonne sodanaikainen kryptoanalyyttinen toiminta keskittyi, on nimeltään *Bletchley Park*. Siellä toimii nykyisin museo: `www.bletchleypark.org.uk/`. Sattumoisin *Turingin* elämästä kerrotaan musiikin keinoin Ooppera Skaalan esityksissä maaliskuusta 2008 alkaen.

Kryptologian historia tarjoaa monta muutakin

jännitysseikkailua. Niihin liittyviä lukuelämyksiä voi etsiä ehkä parhaiten alan klassikosta [CodeB]. Hiiren napsauttamisen päässä olevaa historiatietoa on tutussa lähteessä [Wiki].

Toisen maailmansodan jälkeisiin aikoihin saakka salakirjoitusmenetelmät perustuivat viestin kirjaimien jonkinlaiseen uudelleen järjestämiseen tai sekoittamiseen. Tietotekniikan kehittymisen myötä mekaaniset laitteet voitiin korvata tietokoneohjelmilla.

Alan vallankumouksellinen käännekohta ajoittuu 1970−luvun lopulle. Vuonna 1976 Stanfordin yliopiston tutkijat *Withfield Diffie* ja *Martin Hellman* esittivät julkaisussaan [DH] ns. *julkisen avaimen menetelmän*. Ensimmäisen konkreettisen esimerkin toimivasta julkisen avaimen salausjärjestelmästä esittivät MIT:n tutkijat *R. Rivest, A. Shamir, L. Adelmann* [RSA]. Heidän sukunimiensä alkukirjainten mukaan sai nimensä RSA-salausmenetelmä.

Kannattaa myös mainita, että alalla on suomalaisia, kansainvälisesti arvostettuja tutkijoita, kuten akateemikko, emeritusprofessori *Arto Salomaa* tutkimusryhmineen ja professori *Kaisa Nyberg*, jonka kirjoitus [KN] on kattava, monipuolinen ja ajantasainen katsaus kryptologiaan. *Salomaan* julkaisut ja monografiat, joista viitteenä [AS], ovat runsaasti referoituja, alan kansainvälistä huippua edustavia klassikoita.

1900-luvun puolestavälistä lähtien salausmenetelmät ovat alkaneet käyttää enenevässä määrin matematiikkaa, erityisesti abstraktia algebraa ja lukuteoriaa, kombinatoriikkaa, todennäköisyyslaskentaa ja tilastotiedettä, algoritmien vaativuusanalyysiä, jne.

Tietokoneiden prosessoritehon jatkuva kasvaminen antaa "ilmaiseksi" kryptoanalyytikoille raakaan voimaan perustuvia työkaluja salakoodien murtamiseen. Tämä pitää kryptografia-joukot vireinä ja pakottaa suunnittelemaan entistä nerokkaampia ja mielikuvituksellisempia menetelmiä. Kvanttitietokonelaskennan mahdollisuuksiin on jo alettu varautua, kts. [Wiki] ja *Mikko Möttönen*: Kvantti-informaatio – tämän vuosisadan vallankumous !?, Arkhimedes 1/2008.

Salaustieteen ja -tekniikan käytön painopiste on siirtynyt puolustusvoimiin ja tiedusteluun liittyvästä käytöstä selkeästi jokaisen kansalaisen arkipäivään kuuluvaksi asiaksi. Kaikki luottamuksellisten tietojen välittäminen Internetissä, sähköposti, sähköinen kaupankäynti, pankkitoiminta, tietojärjestelmien salasanat, matkapuhelinliikenne, sähköinen allekirjoitus, äänestys, sirukortit jne. ovat täysin salaamistekniikasta riippuvaisia.

Kirjoitukseni tarkoituksena ei ole antaa kattavaa katsausta alaan muuten kuin tarjoilemalla kiinnostuneille lisäviitteitä eri suuntiin. Varsinainen päämäärä on avata alkeista lähtien yksityiskohtainen matemaattisten päättelyiden ketju näyttääkseni, miten RSA-menetelmä toimii ja valaista sitä esimerkein. Tätä varten joudun vielä jonkinverran täydentämään kirjoituksen osassa 1 kehiteltyä lukuteorian välineistöä.

Parhaassa tapauksessa kirjoitus voisi inspiroida jotakuta matematiikan opettajaa kehittämään aineksia lukion erikoiskurssille tai matematiikkakerholle (jos sellaisia jossain on).

Kertaustietoisku mod-laskennasta

Modulaariaritmetiikka on keskeisessä osassa, joten kerrataan vielä:

$a \equiv b \pmod{n}$ tarkoittaa, että on olemassa kokonaisluku k siten, että $a = b + k\,n$. Toisin sanoen, $n \mid (a-b)$ eli $a-b$ on jaollinen n :llä.
Kun puhutaan luvusta $b \pmod{n}$, tarkoitetaan yleensä pienintä ei-negatiivista muotoa $b \pm k\,n$ olevaa lukua, toisin sanoen jakojäännöstä jakolaskussa b/n. Näin toimivat myös alla olevissa esimerkeissä käytettävät MATLAB- ja MAPLE-ohjelmistojen `mod`- funktiot: MATLAB:ssa `mod(b,n)` ja MAPLE:ssa `b mod n`.

Symmetrisistä salakirjoitusmenetelmistä

Monet lukijoista lienevät joskus nuoruudessaan kokeilleet jotain tapaa välittää viestiä salatussa muodossa sekoittamalla sopivasti viestin kirjaimia. Kenties yksinkertaisin menetelmä on korvata sanan kukin kirjain aakkosissa seuraavalla kirjaimella. Tätä menetelmää käytettiin myös kuuluisassa, vuonna 1968 valmistuneessa elokuvassa "2001: A Space Odyssey" (A.C. Clarke, S. Kubrick), jossa vallankaappausta avaruusaluksen miehistöltä yrittänyt tietokone oli nimeltään `HAL 9000`. Toki tässä nimessä oli kyse vain pienestä pikantista kuriositeetista, joka aukeni osalle katsojia.

Vakaviin sotilaallisiin tarkoituksiin menetelmää lienee käyttänyt ensimmäisenä *Julius Caesar*. Hän on tiettävästi soveltanut kolmen kirjaimen siirtoa eteenpäin viestin salaamiseen ja vastaavasti saman määrän siirtoja taaksepäin salakoodin avaamiseen. Tämän tyyppisiä menetelmiä, jossa kirjainta siirretään aakkosissa määrätty määrä, kutsutaankin yleisesti Caesarin menetelmiksi.

Esimerkki Caesarin menetelmästä

Suoritin itse esimerkin MATLAB-ohjelmaa hyödyntäen, mutta en paneudu ohjelman syntaksiin. Otan vain joitakin MATLAB-näytteitä ja sanontoja, jotka ovat itsensä selittäviä.

Muodostetaan merkkivektorit

```
>> AAKKOSET='ABCDEFGHIJKLMNOPQRSTUVWXYZÅÄÖ '
>> viesti='SAMMONRYÖSTÖ'
```

Muutetaan viestin kirjaimet numeeriseksi vektoriksi laskemalla kunkin kirjaimen sijainti `AAKKOSET`-vektorissa. Siis esim. $A = 0, B = 1, \ldots, Z = 26, \ldots$ Saadaan:

```
18 0 12 12 14 13 17 24 28 18 19 28
```

Alkuperäisessä *Caesarin menetelmässä* lisäämme jokaiseen vektorin komponenttiin luvun 3. Lienee aika selvää, että Ö, jota vastaa numero 28, siirretään syklisesti aakkosten alkupäähän ja siitä tulee siis B (koska valitsimme aakkosvektorimme viimeiseksi merkiksi välilyönnin).

Tämä syklinen siirto tarkoittaa, että redusoimme tuloksen modulo $N = 30$, koska aakkosvektorin pituus on 30. (Voidaan ajatella aakkosvektori kierretyksi ympyrän muotoon, jonka kehällä liikutaan.) Toisin sanoen lasku numeroilla on $(28 + 3) \,(\mathrm{mod}\, 30) = 1$.

Yleisesti salausfunktiomme on siis $(x + 3) \pmod N$, missä x on viestin numeerisen esitysvektorin komponentti (viestin kirjainmerkkiä vastaava numero), ja N on aakkosvektorin pituus. MATLAB-istuntomme jatkuisi näin, kun ajatellaan, että edellä oleva viestin numeerinen esitysvektori on muuttujassa `nviesti`.

```
>> nsalaviesti=mod(nviesti+3,30)
21   3   15   15   17   16   20   27   1   21   22   1
>> AAKKOSET(nsalaviesti+1)
   VDPPRQUÄBVWB       salaviesti kirjaimilla
```

Salaviestin avaus

Viestin vastaanottajalla on tieto menetelmästä ja avainluvusta 3. Sehän voisi yhtä hyvin olla jotain muutakin. Viestin avaaminen tapahtuu salausfunktion käänteisfunktiolla, joka on mitä ilmeisimmin
$f^{-1}(y) = (y - 3) \bmod N$.

```
>> avattu=mod(nsalaviesti-3,N)
     18 0 12 12 14 13 17 24 28 18 19 28
>> AAKKOSET(avattu+1)
SAMMONRYÖSTÖ
```

Salaviestin avaus tuotti alkuperäisen viestin, joten hyvin kävi.

Niille, joita kiinnostaa MATLAB-syntaksi mainitsen, että `AAKKOSET(avattu+1)`-komento tarkoittaa `AAKKOSET`-merkkivektorin indeksointia numeerisella vektorilla, jonka komponentteihin pitää lisätä 1, koska MATLAB:ssa vektorin indeksointi aloitetaan luvusta 1, kun taas jakojäännöksillä laskenta vaatii indeksin alkamaan 0:sta. Sama ilmiö on tietysti lausekkeessa `AAKKOSET(nsalaviesti+1)`.

Yleisesti menetelmä voitaisiin esittää tähän tapaan:
Merkitään $Z_n = \{0, \ldots, n-1\}$. Salausfunktio
$S_e : Z_n \to Z_n; S_e(x) = (x + e) \pmod n$, missä e:tä voidaan kutsua *salausavaimeksi* ja funktiota S_e salausfunktioksi, joskus itse funktiota kutsutaan myös salausavaimeksi.
Salakoodi saadaan avatuksi salausfunktion S_e käänteisfunktiolla $S_e^{-1}(y) = (y - e) \pmod n$.

Menetelmä on *symmetrinen* siinä mielessä, että kun salaus(avain)funktio tunnetaan, niin salakoodin avaamisen suorittava käänteisfunktio voidaan välittömästi määrittää.

Kehittyneempiä symmetrisiä menetelmiä

Caesarin menetelmä yleisessäkin muodossaan on luonnollisesti hyvin haavoittuva.

Erilaisten mahdollisuuksien (salausavaimien) määrää voidaan huikeasti lisätä ottamalla salausfunktioksi jokin muu tapa sekoittaa kirjaimia. Jos otetaan mielivaltainen aakkosten "permutaatio", eli uusi järjestys, niin kaikkien mahdollisten salausavainten lukumäärä on $N!$, missä N on edelleen aakkosvektorin pituus. Yllä olevassa tapauksessa $N = 30$, jolloin saadaan kaikkiaan 30! mahdollisuutta. Kyseessä on luku, jossa on 33 numeroa, joten raakaan voimaan perustuva kaikkien eri mahdollisuuksien läpikäyminen ei onnistu.

Huolimatta avainten lähes äärettömästä määrästä, tämäkin menetelmä on monin tavoin haavoittuvainen. Kun on tiedossa kieli, jolla viestintä tapahtuu, voidaan kirjainten esiintymäfrekvenssien perusteella päästä oikeista kirjaimista selville. Esimerkiksi Suomen kielessä kirjain A on yleisin, joten (pitkässä) salakirjoitetussa viestissä suurimman esiintymäfrekvenssin omaava kirjain on todennäköisesti A. Ja niin edelleen. Frekvenssianalyysin käytöstä salausavaimen selvittämiseksi on hyviä esimerkkejä ja opastettuja harjoitustehtäviä mm. kirjassa [Kob] luvussa III.

Frekvenssianalyysin vaikeuttamiseksi viesti voidaan myös koodata jakamalla se ensin osiin, lohkoihin, jotka numeroidaan. Jos vaikka käytettäisiin 2:n pituisia lohkoja, niin kunkin komponentin numerot olisivat välillä $0, \ldots, N^2 - 1 = 899$, ja kyseessä olisi 2-kirjaimisten tavujen tunnistaminen. Kasvattamalla lohkon kokoa, sanokaamme suuremmaksi kuin 4, frekvenssianalyysiin perustuva tunnistus vaikeutuu olennaisesti.

Caesarin menetelmää on kehitetty ottamalla mukaan avainsana, joka määrää kirjaimien muuntumisen. Jos otetaan avaimeksi satunnainen merkkijono, joka on lähetettävän viestin pituinen, niin lasketaan yhteen viestivektorin ja avainvektorin numeeriset esityksvektorit komponenteittain $\pmod N$. Yllättävää kyllä, tämä menetelmä, englanninkieliseltä nimeltään

"one time pad", on ainoa tunnettu informaatioteorian mielessä luotettavaksi todistettu salausmenetelmä. Käytännössä se on kuitenkin monin tavoin hankala ja epäluotettava, avain on yhtä pitkä kuin viesti, avainvektorit ovat periaatteessa kertakäyttöisiä, ja avainten vaihto on työläs ja riskialtis operaatio

Vaikka salausmenetelmien "avantgarde" on epäsymmetristen menetelmien puolella, symmetrisiä menetelmiä kehitellään myös edelleen. Kohta esiteltävät epäsymmetriset menetelmät, joihin edellä mainittu RSA kuuluu, ovat pitkien viestien käsittelyssä raskaita. Siksi joudutaan usein käyttämään yhdistelmää, jossa esimerkiksi lähetetään symmetrisen menetelmän avain salattuna epäsymmetrisellä menetelmällä ja symmetrisellä salattu (pitkä) viesti.

Lukuteorian täydennystä

Osassa I käsiteltiin kokonaislukujen jaollisuusasiota ja modulaariaritmetiikkaa. Kannattaa kaivaa aktiivimuistiin nuo asiat, ainakin määritelmät ja lauseiden johtopäätökset. Tärkeimpiä: Jakoyhtälö (Lause I.1), SYT-lause (I.5), Eukleideen algoritmi, Fermat'n pieni lause (Lause I.13). Perusasioita lukuteoriasta löytyy myös *Jukka Pihkon* Solmun verkkolehteen kirjoittamasta "lukuteorian helmiä"-artikkelista [JP]. Tarvitsemme osassa I esitetyn lisäksi vielä lukuteoreettisen jälkiruoka-annoksen.

Eukleideen algoritmin laajennus

Palautetaan mieleen lause $I.8$, joka sanoo, että $\mathrm{syt}(a,b) = \mathrm{syt}(b,r)$, kun a esitetään jakoyhtälön (lause $I.1$) ilmaisemassa muodossa: $a = b\,q + r, \quad 0 \le r < b$. Tämä edustaa askelta, joka riittävän monta kertaa toistettuna johtaa $\mathrm{syt}(a,b)$:n arvoon. Menettelyä kutsutaan *Eukleideen algoritmiksi*.

Edelleen muistellaan erityisellä lämmöllä "SYTlausetta" $I.5$. Sehän paljastaa, että $\mathrm{syt}(a,b)$:lle saadaan esitys muodossa $x\,a + y\,b$, missä x ja y ovat kokonaislukukertoimia.

Tähän mennessä olemme osanneet nautiskella pelkästään siitä tiedosta, että tuollaiset luvut x ja y ovat olemassa, menettelyä niiden määrittämiseksi emme ole esittäneet. Nyt on sen aika!

Katsotaanpa esimerkin valossa:

Esimerkki 1. *Määrättävä* $\mathit{syt}(171,30)$.
$171 = 5 \cdot 30 + 21$.
Euklalgo: $\mathit{syt}(171,30) = \mathit{syt}(30,21)$.
Jakoyhtälö: $30 = 1 \cdot 21 + 9$.
Euklalgo: $\mathit{syt}(30,21) = \mathit{syt}(21,9)$.
Jakoyhtälö: $21 = 2 \cdot 9 + 3$.
Euklalgo: $\mathit{syt}(21,9) = \mathit{syt}(9,3) = 3$.
(Seuraava jakoyhtälö: $9 = 3 \cdot 3 + 0$ *johtaisi jakojäännökseen* $r = 0$*, joka on algoritmin lopetusehto.)*
Siis $\mathit{syt}(171,30) = 3$.

Tähän saakka kerrattiin vanhaa. Johtaaksemme tavoitellun esityksen, käytämme luvuille kirjainsymboleja asian selkeyttämiseksi. Merkitään $a = 171, b = 30$, ja syntyvät jakojäännökset olkoot $r_1, r_2, r_3, \ldots$. Tässä $r_1 = 21, r_2 = 9, r_3 = 3, r_4 = 0$.
Päämääränä on siis lausua viimeinen nollasta poikkeava jakojäännös (tässä r_3) muodossa $r_3 = x\,a + y\,b$. Kirjoitetaan yllä olevat jakoyhtälöt näitä merkintöjä käyttäen:

$$\begin{cases} a = 5 \cdot b + r_1 \\ b = r_1 + r_2 \\ r_1 = 2 \cdot r_2 + r_3. \end{cases}$$

Kun tämä puretaan alhaalta ylöspäin, saadaan haluttu esitys. Tarkemmin sanottuna:

1. Ratkaistaan alimmasta yhtälöstä r_3 r_1:n ja r_2:n avulla.
2. Sijoitetaan tähän r_3:n lausekkeeseen r_2 ratkaistuna toiseksi alimmaisesta yhtälöstä r_1:n ja b:n avulla.
3. Sijoitetaan näin saatuun r_3:n lausekkeeseen r_1 yhtälöstä 1 lausuttuna a:n ja b:n avulla. Tämä strategia konkretisoituu esimerkkimme tilanteessa näin:

$r_3 = r_1 - 2 \cdot r_2$.
$r_2 = b - r_1$ sij. (3):een $\implies$ $r_3 = r_1 - 2 \cdot b + 2 \cdot r_1 = 3 \cdot \underbrace{r_1}_{a-5\cdot b} - 2 \cdot b$

Sieventämällä: $\mathrm{syt}(a,b) = r_3 = 3 \cdot a - 17 \cdot b$.

Menettely on nimeltään *Laajennettu Eukleideen algoritmi*.
Huomautan, että kirjoituksessa [JP] on toinen vastaavanlainen esimerkki laskettuna auki. Samaisessa kirjoituksessa [JP] ss. 6–7 esiintyy myös hauska konstruktiivinen todistus SYT-lauseelle. Todistus etenee *Eukleideen algoritmin tahdissa*, toisin kuin se, jonka esitin osassa I. Vastaavanlaisia esimerkkejä on myös kirjoissa [I. Lukio1] ja [I. Lukio2] aihepiireissä *Eukleideen algoritmi* ja *Diophantoksen yhtälöt*.

Ohjelma laajennettuun Eukleideen algoritmiin.

Nämä esimerkit huolella läpikäytyään lukija varmasti vakuuttuu siitä, että tällainen tehtävä voidaan aina ratkaista, ja osaa pyydettäessä ratkaisun suorittaa. En muotoile lausetta matemaattiseksi algoritmiksi tai lauseeksi, vaan kirjoitan sen suoraan tietokoneohjelmaksi, jollaisena se tietysti käytännön sovelluksissa esiintyy.

Kirjoituksen osassa I esitin rekursiivisen ohjelman Eukleideen algoritmille. Siitä sopivasti laajentamalla saadaan ällistyttävän yksinkertainen koodi laajennetulle. Koodi noudattaa MAPLE-kielen syntaksia, mutta

voidaan muokata helposti mille tahansa rekursion sallivalle kielelle.

```
EEukleides:=proc(a,b)
if b=0 then [a,1,0]
else L:=EEukleides(b,mod(a,b));
     d:=L[1]; x:=L[2]; y:=L[3];
     L:=[d,y,x - iquo(a,b)*y] #iquo:osamäärä
end if
end proc
```

Tämä rekursiivinen (itseään kutsuva) ohjelma laskee siis luvun $d = \mathrm{syt}(a, b)$ ja kertoimet x ja y siten, että $d = a\,x + b\,y$ (vrt. SYTlause).
Edellä oleva esimerkki laskettaisiin näin:

```
> tulos := EEukleides(171, 30);
> d := tulos[1]: x := tulos[2]: y := tulos[3]:
> d,x,y;
        3, 3, -17
```

Saatiin kuin saatiinkin sama kuin käsin laskemalla!
Algoritmi on kirjoitettu sovittamalla [I. Algo]-kirjassa annettu "pseudokielinen" ohjelma MAPLE-syntaksin mukaiseksi. Elegantti tapa laajennetun *Eukleideen algoritmin* yleiselle matemaattiselle todistukselle on yllä olevan ohjelman oikeaksi todistaminen, joka onkin yllättävän lyhyt ja ytimekäs ([I. Algo] Ch. 33. s. 812.

Kuten sanottu, käytän esimerkeissäni symbolilaskentaohjelmaa MAPLE. Esimerkkien lukeminen ei edellytä lainkaan ohjelman tuntemista. Selitän tarvitsemani komennot, joita on vain muutama. Ohjelman ottaminen mukaan on sikäli mielekästä, että voidaan esittää asiat ihan oikean kokoisilla luvuilla. Se myös antaa konkretiaa teoreettisesti kuvailtaville algoritmeille, ja osoittaa, että hyvä symbolilaskentaohjelma on hyödyllinen kryptologiatyökalu.Sekä MAPLE:n että MATHEMATICA:n käyttäjäryhmissä on suuri määrä kryptologian esimerkkityöarkkeja. (`http://www.maplesoft.com/applications/` → `mathematics` → `Cryptography` (23 kpl.) ja `http://www.wolfram.com/`)
Toisaalta kannattaa mainita vapaasti saatava ohjelma *Maxima*, jolla kiinnostuneet voivat kokeilla vastaavien asioiden toteuttamista, ellei MAPLE- tai MATHEMATICA-ohjelma ole käytettävissä. Viimemainituista painotan enemmän edellistä siksi, että se on itselleni tutumpi.

Huomautan vielä, että MAPLE:ssa on sisäänrakennettuna laajennettu *Eukleideen algoritmi* nimellä `igcdex`. Nimi koostuu osista: `i` - integer, `gcd` = *greatest common divisor* = `syt` ja `ex` - *extended* Komento `d:=igcdex(a,b,'x','y')` palauttaa tuloksen `d=syt(a,b)` ja lisäksi muuttujissa x ja y kertoimet, joilla $d = a\,x + b\,y$.
Edellinen esimerkki laskettaisiin näin:

```
> d:=igcdex(171,30,'x','y'):
> d,x,y;
        3, 3, -17
```

Jatkossa käytämme valmista `igcdex`-funktiota esimerkeissämme edellä esitetyn `EEukleides`-funktion sijasta.

Yhtälön $a\,x \equiv 1 \pmod{n}$ ratkaiseminen

Kyseessähän on mahdollisimman yksinkertainen ns. *Diophantoksen yhtälö*, jota käsiteltiin osassa I. Lauseen I.12 mukaan yhtälöllä on aina yksikäsitteinen ratkaisu (mod n), mikäli $\mathrm{syt}(a, n) = 1$. Ratkaisu saadaan *laajennetulla Eukleideen algoritmilla* laskemalla luvut x ja y siten, että $a\,x + n\,y = 1$. Kiinnostava on vain luku x. Kun se tunnetaan, niin
$a\,x = 1 - n\,y \equiv 1 \pmod{n}$.

Yllä olevalla MAPLE-funktiolla ratkaisu saadaan siis näin:

```
> igcdex(a,n,'x','y'):
```

Kuten sanottu, olemme kiinnostuneita vain x:stä, d:n tiedämme 1:ksi, y voidaan heittää romukoppaan. Jos halutaan minimaalinen x, ts. redusoida (mod n), niin ei muuta kuin `x:=x mod n :`

Kiinalainen jäännöslause

Aiheeseen liittyy monta tarinaa erityisesti kiinalaisen (ja kreikkalaisen) matematiikan historiasta. Löytöjä voi tehdä *Googlella* hakusanalla *Chinese remainder theorem*. Eräs tarinoista on [I. Algo]-kirjassa:
Noin vuonna 100 *ajanlaskumme alkuhetken jälkeen kiinalainen matemaatikko Sun-Tsu ratkaisi seuraavan ongelman:*
Määritä kokonaisluvut x, jotka antavat jakojäännökset $2, 3, 2$ *jaettaessa luvuilla* $3, 5, 7$. *Ratkaisuiksi hän sai luvut* $x = 23 + k \cdot 105$.

Huomattakoon, että jakajat $n_1 = 3, n_2 = 5, n_3 = 7$ ovat pareittain yhteistekijättömät ja $3 \cdot 5 \cdot 7 = 105$, mutta miksi näin, ja mistä tulee 23 ? Nykykielellä siis ratkaisu: $x \equiv 23 \pmod{n}$, missä $n = n_1 \cdot n_2 \cdot n_3$.

Miten hän ratkaisuunsa päätyi, ja osasiko yleistää? Nykylukijalla on helppoa, sovelletaan kiinalaista jäännöslausetta, jota tässä ei kuitenkaan esitetä. Mainittakoon kuitenkin, että lauseen muotoili ja todisti yleisessa muodossaan – kukas muu kuin *Euler* v. 1734.

RSA-algoritmin todistuksessa tarvittava kiinalaisen jäännöslauseen seuraus on siinä määrin erikoistapaus itse lauseesta, että sen suora todistus on miltei itsestäänselvyys. Kiinalaista jännöslausetta voidaan kyllä käyttää mm. RSA-menetelmän tehokkaampaan toteutukseen ja myös RSA-menetelmän murtamisyrityksiin, joten sen esittäminen olisi hyvinkin perusteltua tässä yhteydessä. Katson kuitenkin kirjoitukselleni olevan hyväksi keventää työkalupakkia, kun se on mahdollista.

Kutsun tarvitsemaamme seurausta "kiinalaiseksi selviöksi".

Lause 1 (Kiinalainen selviö)**.** *Olkoon* $n = n_1\, n_2\, \cdots n_k$, *missä* $syt(n_i, n_j) = 1$, *kun* $i \neq j$. *Olkoon* a *mielivaltainen kokonaisluku. Nyt* $x \equiv a \pmod{n} \iff x \equiv a \pmod{n_i}, i = 1, \ldots, k.$

Tod. (1) Olkoon $x \equiv a \pmod{n}$. Jatko jääköön lukijalle harjoitustehtäväksi.

(2) Olkoon $x \equiv a (\mathrm{mod}\, n_i), i = 1, \ldots, k$. Tällöin jokainen n_i on tekijänä $(x - a)$:ssa. Koska $syt(n_i, n_j) = 1$, kun $i \neq j$, niin myös tulo $n = n_1\, n_2 \cdots n_k$ on tekijänä $(x-a)$:ssa, eli $x \equiv a (\mathrm{mod}\, n)$. Tämä viimeinen päätelmä jätetään taas lukijalle harjoitustehtäväksi (miltei valmiiksi ohjeistettuna). □

Tehtävä 1. *Olkoon* $syt(n_1, n_2) = 1$ *ja olkoon* $n_1 \mid a$ *ja* $n_2 \mid a$. *Osoita, että* $n_1\, n_2 \mid a$.

Vihje Taas kerran päästään liikkeelle SYTlauseen I.5 avulla: $1 = n_1\, x + n_2\, y$. Kerro puolittain a:lla, niin alat olla perillä.

On selvää, että tämä yleistyy useampaan tekijään (formaalisti induktiolla, jota esitellään mm. kirjoituksessa [JP]) . Toisaalta tarvitsemme "kiinalaista selviötä" vain tapauksessa $n = n_1\, n_2$.

Modulaaripotenssilaskenta

Tässä on kaksi näkökulmaa, 1) laskentakaava "potenssi potenssiin" ja 2) tehokas potenssiinkorotusmenetelmä.

1. Potenssi potenssiin. Silloinhan eksponentit kerrotaan. Päteekö sääntö myös (mod)-laskennassa? Mitä on siis $(a^k \mathrm{mod}\, n)^j$?
$a^k \mathrm{mod}\, n = a^k + i\, n$ jollain i.
Siis $(a^k \mathrm{mod}\, n)^j = (a^k + i\, n)^j = a^{k\, j} +$ termejä, joissa jokaisessa on $i\, n$ tekijänä. (Joko binomikaavalla tai suoraan tulosta $(a^k + i\, n)(a^k + i\, n) \cdots (a^k + i\, n)$, jossa kaikkiin muihin termeihin paitsi siihen, jossa jokaisesta binomista otetaan ensimmäinen, tulee tekijäksi $(i\, n)$:n potenssi.) Eli saadaan muotoa $a^{k\, j} + K\, n$ oleva lauseke, missä K on jokin kokonaisluku. Siis

$$(a^k \mathrm{mod}\, n)^j \equiv a^{k\, j} \pmod{n},$$

ja potenssi potenssiin sääntö toimii kauneimmalla mahdollisella tavalla.

Modulaarinen potenssiinkorotus

Usein näissä yhteyksissä tulee vastaan tehtävä $a^b \pmod{n}$, missä esiintyvät luvut saattavat olla hyvin suuria. Jos vaikka a on luokkaa 10^4 ja b luokkaa 10^5, mikä on huimasti alakanttiin tyypillistä RSA-salausta ajatellen, niin a^b on luokkaa 10^{400000}. Tämänkokoisilla luvuilla laskenta alkaa olla epätoivoista mille tahansa laskentamyllylle, puhumattakaan ihan normaalista tapauksesta, kuten esimerkissämme, jossa a tai b voi olla luokkaa 10^{100}.

Operaatioon on olemassa tehokas ratkaisu, nimeltään "toistettu neliöön korottaminen" tai "neliöön korotus ja kerolasku". Algoritmi kuvataan [I. Algo]-kirjassa ss. 829 ja [Kob] ss. 23 – 24: Pääperiaate on, että kerätään tuloa, jossa edellisellä kierroksella saatu tulos korotetaan neliöön ja lisäksi kerrotaan sopivalla luvulla, mikäli n:n binääriesityksessä on ao. kohdalla 1. Kunkin operaation jälkeen redusoidaan modulo n, jolloin suurin laskennassa esiintyvä luku on kaiken aikaa $\leq n^2$. (Oletetaan, että $a < n$.) Jätän tilan säästämiseksi kuvauksen ylimalkaiseksi, yksityiskohdat on annettu mm. yllä mainituissa viitteissä.

MAPLE-ohjelmassa algoritmi on suoraan sisäänrakennettuna komentona: `> c:=a&^b mod n:` Jos tätä verrataan komentoon `c:=(a^b) mod n`, jota en suosittele, niin ero saattaa olla huikea, sanoisinko ääretön, ja ensin kannattaa varmistaa, että ohjelman `STOP`-nappula toimii.

Algoritmien vaativuus

Laskennallisten algoritmien suhteen on tärkeää arvioida niiden vaatimaa laskenta-aikaa (ja tilaa) suhteessa syötteenä olevien lukujen kokoon. Tällaisia arvioita esiintyy kaikissa numeerisen analyysin kirjoissa, alan systemaattisen tutkimuksen katsotaan kuuluvan teoreettisen tietojenkäsittelytieteen piiriin. Tässä kirjoituksessa esiintyviin lukuteoreettisiin algoritmeihin paneutuvaa vaativuusanalyysia käsitellään kirjassa [Kob] huolellisesti ja perusteellisesti. Myös toinen peruskirjamme [I. Algo] sisältää runsaasti vastaavaa analyysia. Tähän kirjoitukseen en voi enää mahduttaa uutta asiaa enempää, joten joudun tyytymään tämän tärkeän komponentin osalta kirjallisuusviitteisiin. Mainitsen vain esimerkin ja ulkonäön vuoksi, missä muodossa näitä arvioita annetaan. *Eukleideen algoritmin* suoritusaika on $O(\log^3 a)$, kun etsitään $syt(a, b)$, $a > b$. Lyhyesti tämä "iso-O"-notaatio tarkoittaa suluissa olevaan lausekkeeseen verrannollisuutta, eli arvioon kuuluu tarkemmin määräämätön vakiokerroin.
Aikaperspektiivin kannalta mainitsen, että viime marraskuuta voidaan pitää numeerisen analyysin ja laskennallisten numeeristen algoritmien tutkimuksen 60- vuotisjuhlakuukautena, sillä marraskuussa vuonna 1947 ilmestyi *von Neumann*'n ja *Goldstine*'n uraauurtava julkaisu aiheesta.

Epäsymmetriset menetelmät, julkinen salakirjoitus

Kaikki vuoteen 1976 mennessä käytetyt kryptosysteemit lasketaan klassisiin menetelmiin kuuluviksi. Niille on ominaista, että salausavaimesta S_e voidaan kohtuullisella laskentatyöllä johtaa purkuavain S_d. Siksi

niitä kutsutaan myös symmetrisiksi menetelmiksi, kuten edellä oli puhe. Käytän tulevaa ennakoiden kirjaimia e ja d, jotka liittyvät sanoihin "encrypt"(salata) ja "decrypt" avata salaua, dekryptata.

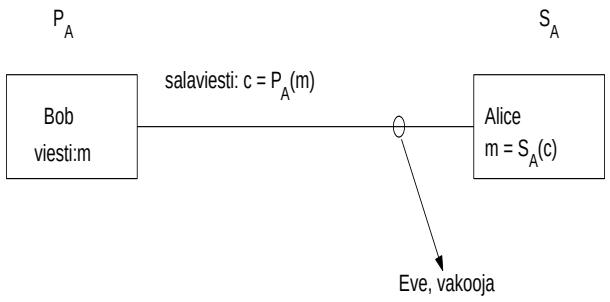
Nykyisin, kun sähköinen luottamuksellisen tiedon välitystarve on räjähdysmäisesti kasvanut sotilas- ja diplomaattialueen ulkopuolelle, on suoranainen välttämättömyys suunnitella yhtenäinen salaustekniikka, jotta laajojen järjestelmien yhteesopivuus olisi mahdollinen. Niinpä tarve yleiseen käyttöön sopivan julkisen salakirjoitusjärjestelmän kehittämiseen nousi 1970-luvulla voimakkaasti esiin.

Julkisen salakirjoituksen periaate

Jokaisella tiedonvaihtoon osallistuvalla osapuolella on kaksi avainta, julkinen ja salainen. Kryptologian esityksissä on tullut yleiseksi tavaksi kutsua kahta kommunikoivaa osapuolta nimillä *Alice* ja *Bob*.

Käytetään edellä olevien *e*- ja *d*- symbolien ohella myös kirjaimia P – "Public"ja S – "Secret" viittaamaan kyseisiin avainfunktioihin. *Alice*:lla on siten julkinen avain P_A ja salainen avain S_A ja *Bob*:lla vastaavasti P_B ja S_B.

Kommunikaatiota *Alice*:n ja *Bob*:n välillä esittää kuva, jossa on mukana ilkeä *Eve*, joka sieppaa linjalta salakirjoitettuja viestejä, minkä ehtii.



Kaikilla kommunikoivilla osapuolilla on käytössään kaikkien muiden julkinen avain, periaatteessa ne voitaisiin vaikka julkaista webbisivulla. Lisäksi jokaisella osapuolella X on oma henkilökohtainen salainen avain S_X, jota kukaan muu maailmassa ei tiedä.

Voimme ymmärtää tässä kuvailussa avaimen funktioksi, joka muuntaa viestin numeerisen esityksen selväkielisestä salakieliseksi tai päinvastoin. Joka tapauksessa funktiot P_X ja S_X ovat toistensa käänteisfunktioita. Niinpä, kun *Bob* salakirjoittaa viestin m käyttäen *Alice*:n julkista avainta P_A, hän tuottaa luvun $c = P_A(m)$. *Alice*:lla ja vain *Alice*:lla on salainen avain S_A, joka on *Alice*:n julkisen avaimen P_A käänteisfunktio. Siis *Alice* avaa *Bob*:n lähettämän viestin laskulla $S_A(c)$.

Mikä tässä on uutta ja ihmeellistä? Kaikki osapuolet tuntevat avaimen P_A, mutta tämä funktiopa onkin tehty sellaiseksi, että sen käänteisfunktiota on äärimmäisen vaikea laskea. Tässä on ero symmetrisiin menetelmiin nähden. *Diffie ja Hellmann* käyttivät termiä "trapdoor function", jolle [KN]:ssä käytetään suomennosta "salaovifunktio". Kehitin itse kevytmielisesti suomennoksen "loukkuluukkufunktio". Luukusta on helppo astua sisään huomatakseen joutuneensa loukkuun. Takaisin ulos päästäkseen on avattava systeemilukko, jossa on miljoonia mahdollisuuksia. Matemaattisemmin ilmaistuna, on löydettävä laskennallisesti kohtuullinen tehtävä, jonka käänteistehtävä on laskennallisesti kohtuuton, kuten 1000 vuotta vaativa laskenta-aika nopeimmalla supertietokoneella ja parhaalla laskenta-algoritmilla.

RSA-algoritmissa tuo laskennallisesti kohtuullinen tehtävä on kahden hyvin suuren alkuluvun p ja q kertominen: $n = p\,q$. Kun luvut valitaan riittävän suuriksi (luokkaa 150 numeroa), niin käänteinen tehtävä, luvun n tekijöihin jako on laskennallisesti kohtuuton. Käsitteiden "kohtuullinen" ja "kohtuuton" kvantifioiminen edellyttää algoritmien laskennallisen vaativuuden arvioita, mihin liittyvää "yleissivistystä" annettiin edellä lyhyesti.

On tietysti myös muistettava, että tietokoneiden prosessoritehojan kasvun ja aivan uusien laskentainnovaatioiden takia *loukkuluukku* vuonna 2008 ei välttämättä olekaan sitä enää vuonna 2018.

Mutta jatkakaamme RSA-menetelmän kuvailun tiellä. Edellä symmetrisistä menetelmistä puhuttaessa koodattiin viestit kirjain tai tavu kerallaan numeroksi ja muunnettiin niitä sopivasti sekoittamalla. Lukuteoria ja tietotekniikka antavat mahdollisuuden käsitellä viestejä suurina kokonaislukuina. Viestin numerokoodi käsitetään numeerisen vektorin sijasta yhdeksi suureksi kokonaisluvuksi, jolle tehdään sopiva matemaattinen muunnos P. Vastaanottaja avaa sen käytössään olevalla käänteismuunnoksella, eli salaisella avaimella S. Jos viesti on kovin pitkä, se jaetaan lohkoihin, joiden pituudet voivat olla vaikka $100 - 200$ merkkiä. Ts. viesti voidaan esittää numeerisena vektorina, jonka komponentit ovat (tarvittaessa) suuria kokonaislukuja.

Seuraavaksi esitettävän algoritmin kuvauksessa ajatellaan, että selväkielisen viestin numeerinen vastine edustaa koko viestiä tai yhtä viestivektorin komponenttia. Tälle käytetään merkintää m, niinkuin "message". Salakirjoitetulle muunnokselle käytetään nimeä c, niinkuin "ciphertext", yleisesti siis $c = P(m)$ ja $m = S(c)$, koska P ja S ovat toistensa käänteisfunktioita.

Miten nuo P ja S rakennetaan RSA-menetelmässä? Nyt olemme valmiit sen kertomaan ja perustelemaan.

Algoritmi 2 (RSA)**.** .

1. *Muodostetaan kaksi samaa suuruusluokkaa olevaa suurta alkulukua p ja q.*
2. *Lasketaan $n = p\,q$ ja $\phi = (p-1)(q-1)$.* [1]
3. *Valitaan luku e, $1 < e < \phi$ siten, että $\mathrm{syt}(e, \phi) = 1$. (Luvun e ei tarvitse olla kauhean suuri.)*
4. *Lasketaan laajennetulla Eukleideen algoritmilla luku d siten, että $e\,d \equiv 1 \pmod{\phi}$.*
5. *A:n **julkinen avain** on (e, n) ja A:n **salainen avain** on (d, n).*
6. *Olkoon m viestin numeerinen esitys, $0 \le m \le n$. **Muodostetaan salainen viesti** $c = m^e (\mathrm{mod}\, n)$ ja lähetetään A:lle.*
7. *A **avaa salaisen viestin** c omalla salaisella avaimellaan d kaavalla $a = c^d \pmod{n}$. Ja todellakin $a = m$, kuten seuraavaksi osoitetaan.*

RSA-algoritmin oikeaksi todistaminen

Tod. Salainen viesti $c = m^e \pmod{n}$, missä m on alkuperäinen viesti, e salausekponentti, $n = p\,q$, p ja q ovat alkulukuja. Avattu viesti $a = c^d \pmod{n}$ missä d on avauseksponentti ja toteuttaa yhtälön

$$e\,d \equiv 1 \pmod{\phi}, \quad \phi = (p-1)(q-1).$$

Pitää siis todistaa, että avattu viesti on sama, mistä lähdettiin, eli $a = m$.

No katsotaan: $c^d = (m^e \pmod{n})^d \equiv m^{e\,d} \pmod{n}$ potenssipotenssiin-säännön mukaan.

Nyt $e\,d = 1 + j\,\phi = 1 + j\,(p-1)(q-1)$ jollain j, joten

$m^{e\,d} = m\,m^{j(p-1)(q-1)}$. Järkevä oletus on, että $m < p$ ja $m < q$, jolloin varmasti $\mathrm{syt}(m, p) = \mathrm{syt}(m, q) = 1$.

Niinpä **Fermat'n pikkulauseen** mukaan
$m^{p-1} \equiv 1 \pmod{p}$ ja $m^{q-1} \equiv 1 \pmod{q}$, joten
$m^{j(p-1)(q-1)} = (m^{p-1})^{j(q-1)} \equiv 1 \pmod{p}$.
Vaihtamalla p:n ja q:n järjestys, saadaan
$m^{j(p-1)(q-1)} \equiv 1 \pmod{q}$.

Kiinalaisen selviön mukaan
$m^{j(p-1)(q-1)} \equiv 1 \pmod{n}$.

Siis $a = c^d \equiv m\,m^{e\,d} \pmod{n} \equiv m \pmod{n}$

Jos luovumme yllä olevasta "järkevästä oletuksesta", niin esim. $p \mid m$, jolloin $m^{e\,d} \equiv m \pmod{p}$, koskapa p on molemmissa yhtälön puolissa tekijänä. Tässä tapauksessa ei tarvita Fermat'n apua, mutta tilanne on syytä turvallisuussyistä kuitenkin sulkea pois. Joka tapauksessa väite pätee yleisesti, tehtiinpä yllä järkevä tai vähemmän järkevä oletus. □

Esimerkki

Esitykseni on mukaelma viitteen [Cos] tyylistä. Otan "mustina laatikoina" siinä annetut funktiot `to_number` ja `from_number`, jotka muuttavat merkkijonon eli tekstivektorin numeroksi ja vastaavasti takaisin merkkijonoksi annetun aakkosvektorin suhteen vastaavaan tapaan kuin edellä olleet MATLAB-funktiot. Erona on, että nyt emme muuta tekstivektoria numeerikseksi vektoriksi, vaan yhdeksi kokonaisluvuksi. Kaiken muun avaan komento komennolta lukijan silmien eteen.

Esimerkin avaimet ovat lähes turvallista kokoa, olisivat olleet vielä hiukan yli 10 vuotta sitten. Jäljempänä pohditaan tarkemmin.

Jos muutamme valitun aakkosvektorin suhteen sanan `SOLMU` numeeriseksi saamme:

```
> aakkoset:="ABCDEFGHIJKLMNOPQRSTUVWXYZÅÄÖ "
> to_number("SOLMU");
                  1915121321
```

Nähdään, että kirjaimia vastaavat numerot on pantu peräkkäin tyyliin: $A = 01, B = 02, \ldots, S = 19, \ldots U = 21$

Toinen, kenties luonnollisempi tapa olisi esittää viesti $N-$järjestelmän lukuna ja muuntaa se $10-$järjestelmään, missä N on aakkosvektorin pituus. Tällöin samainen `SOLMU` muuntuisi luvuksi
$21 + 13 \cdot 30 + 12 \cdot 30^2 + 15 \cdot 30^3 + 19 \cdot 30^4 = 15806211$.

Koska yllä mainittu `to_number`-funktio tekee edellisellä tavalla, noudatan sitä.

Alla esiintyvät MAPLE-komennot ovat suurelta osin itsensä selittäviä. Mainitsen ja kertaan tässä joitakin nimityksiä ja sellaisia komentoja, joiden merkitys voi olla epäselvä tai unohtunut.
Alkukirjain i viittaa sanaan "integer", kokonaisluku.

`ifactor`	Tekijöihin jako
`igcd(a,b)`	gcd = syt
`igcdex(a,b,'x','y')`	gcd extended = Laajennettu Eukleideen algoritmi
`a mod b`	Jakojäännös laskussa a/b

Muistutan vielä, että komento `d:=igcdex(a,b,'x','y')` palauttaa tuloksen `d=syt(a,b)` ja lisäksi muuttujissa x ja y kertoimet, joilla $d = a\,x + b\,y$.

[1] Kirjain ϕ viittaa ns. *Eulerin ϕ-funktioon*, emme kuitenkaan tarvitse siihen liittyvää *Eulerin lausetta*, vaan pärjäämme Fermat'n pikkulauseella.

Ja nyt se alkaa!

Alice valitsee kaksi suurta alkulukua ja laskee niiden tulon:

```
> pA := nextprime(10^60 + 1234567*rand()^5);
11980768160215583569801524136786 21
55248273111437740290921929815 59
> qA := nextprime(10^65 + 8765439999*rand()^5);
24557694884338801620018108793784400
77015568602607435050878572990629
> nA:=pA*qA:
> length(nA);
131
```

Luvussa n_A on 131 numeroa. Kokeillaan tekijöihinjakoa.

```
> ifactor(nA);
Warning,  computation interrupted
```

Painettiin STOP-nappulaa. Ei näytä siltä, että kannattaisi ryhtyä odottelemaan.
Kokeillaan tekijöihinjakoa, kun p_A kerrotaan jollain umpimähkään valitulla pienehköllä luvulla.

```
> ifactor(pA*465734);
(2) (337) (691) (11980768160215583569801524 1
3678621552482731114377402909219298155 9)
```

Onnistuu muotoa $p_A\, k$ olevalle luvulle hetkessä, kun kerroin k on miljoonan luokkaa, mutta jos vähän kasvatetaan, alkaa kestää tuskaisen kauan.

Seuraavaksi Alice laskee ϕ_A:n, jota varten siis täytyy tuntea n_A:n tekijät p_A ja q_A.

```
> phiA := (pA - 1)*(qA - 1):
```

Alice valitsee salauseksponentin ("encryption exponent") .

```
 > eA := nextprime(10^5 + rand());
                 624044487349
```

Todetaan, että syt$(e_A, \phi_A) = 1$. Tämä toteutuu hyvin suurella todennäköisyydellä (miksi?), mutta se on kuitenkin erikseen tarkistettava.

```
 > igcd(eA, phiA);
                       1
```

Alice laskee nyt laajennetulla Eukleideen algoritmilla oman salaisen avaimensa eksponentin d_A.

```
> igcdex(eA, phiA, 'xA', 'yA'):
> dA := xA mod phiA;
13684831311997866502152356 52 ...
> length(dA);
                      131
```

Salaisessa eksponentissa d_A on 131 numeroa. Tarkistetaan, vaikka tiedetään, että $e_A\, d_A \equiv 1 \pmod{\phi_A}$

```
> eA*dA mod phiA;
                       1
```

Alicen julkinen avain on (e_A, n_A) ja salainen avain on (d_A, n_A). Edellä esitellyn puhetavan mukaisesti voidaan myös sanoa, että A:n julkinen avain on parametrien (e_A, n_A) määräämä funktio
$P_A(m) = m^{e_A} \pmod{n_A}$ ja salainen avain vastaavasti funktio $S_A(c) = c^{d_A} \pmod{n_A}$.
Bob tekee vastaavat asiat tykönään:

```
> pB := nextprime(10^60 + 23453218*rand()^5):
> qB := nextprime(10^65 + 12456800*rand()^5):
> nB:=pB*qB: phiB:=(pB-1)*(qB-1):
> eB := nextprime(10^4 + 3*rand());
                 2336493690667
> igcd(eB, phiB);
                       1
> igcdex(eB, phiB, 'xB', 'yB'):
> dB := xB mod phiB:
> length(dB);
                      131
```

Bob'n julkinen avain on (e_B, n_B) ja salainen avain (d_B, n_B), ja ne määräävät samalla tavoin julkisen ja salaisen avainfunktion P_B ja S_B. Huomaa, että kummankin salainen avain pysyy kunkin omana tietona. Sitä ei kukaan missään vaiheessa lähetä kenellekään.

Bob salaa numeroksi koodatun viestin m käyttäen Alicen julkista avainta, ts. hän suorittaa laskun
$c = P_A(m) = m^{e_A} \pmod{n_A}$, ja lähettää salaviestin c Alicelle, kas näin:

```
> m := to_number(‘TAVATAAN HIEKKALAATIKOLLA‘);
 20012201200101143208090511110112010120091115 1
 21201
> c:=m&^eA mod nA:  # modulaarinen potenssi
```

Alice avaa viestin omalla salaisella avaimellaan:

$a = S_A(c) = c^{d_A} \pmod{n_A}$. Edellä oikeaksi todistetun algoritmin mukaan pätee: $a = m$.

```
> a:=c&^dA mod nA;
 20012201200101143208090511110112010120091115 1
 21201
from_number(a);
            "TAVATAAN HIEKKALAATIKOLLA"
```

Sähköinen allekirjoitus

Alice vastaa ja varustaa viestinsä digitaalisella allekirjoituksella, jotta *Bob* varmasti tietää, että vastaaja on *Alice* ja viesti on tarkalleen se, jonka *Alice* on lähettänyt.

Viesti voisi olla
`v:="OLEN ALIISA JA TULEN KANSSASI"`. Olkoon m tämän viestin numeerinen esitys. *Alice* muodostaa allekirjoituksen $s = S_A(m)$, toisin sanoen hän koodaa viestinsä omalla **salaisella avaimellaan**, ja lähettää *Bob*:lle viestin, jonka perään liittää tuon koodin, eli luvun s. *Bob* vastaanottaa sanoman vektorina $[v, s]$, lukee selväkielisen viestin v, jonka perusteella tietää käyttää *Alice*:n julkista avainta allekirjoituksen tarkistamiseen.

Bob suorittaa laskun $P_A(s)$. Jos tuloksena on m, on *Alice*:n identiteetti ja viestin sisältö varmennettu.

Sama laskettuna auki

Määritellään MAPLE:ssa edellä käyttettyjen avaimien avulla *Alice*:n julkinen ja salainen avainfunktio P_A ja S_A

```
> PA:=m->m&^eA mod nA:  SA:=s->s&^dA mod nA:
> v:="OLEN ALIISA JA TULEN KANSSASI";
> m:=to_number(v);
   15120514270112090919012710012720211205142
   71101141919011909
```

Alice:n digitaalinen allekirjoitus

```
 > s:=SA(m);  (130 numeroa):
93271633443294939878071410488947105653963 53..
```

Alice lähettää viestin:

```
> Bobille:=[v,s];
["OLEN ALIISA JA TULEN KANSSASI", 9327163...]
```

Bob muuntaa saamansa vektorin 1. komponentin numeeriseksi ja tarkistaa viestivektorin 2. komponentin avulla viestin allekirjoitetun sisällön.

```
> m:=to_number(Bobille[1]);
  15120514270112090919012710012720211205 14..
> t:=PA(Bobille[2]);
  15120514270112090919012710012720211205 14..
 > m - t
                      0
```

Kaikki hyvin!

Tähän voidaan tietysti vielä lisätä allekirjoitetun viestin salaus mukaan tarvittaessa. Edellinen vastaisi (avointa) allekirjoituksella varmennettua paperia ja jälkimmäinen kuoreen sinetöityä allekirjoitettua paperia.
Tyypillisiä tilanteita ovat vaikkapa pankkisovellukset. Saman allekirjoituksen voi haluta tarkistaa useampikin taho. Sehän käy, koska julkinen avain on kaikilla käytössään ja salainen vain asianomaisella allekirjoittajalla. Tyypillinen esimerkki voisi olla *Bob*:n *Alice*:lle lähettämä sähköinen shekki, jonka allekirjoituksen *Alice* tarkistaisi, lähettäisi edelleen pankkiin, jossa se niinikään voitaisiin tarkistaa *Bob*:n julkista avainta käyttäen ja suorittaa asianmukainen rahojen siirto.

RSA:n turvallisuus

RSA:n turvallisuus perustuu suuren luvun tekijöihinjakotehtävän vaativuuteen. Jos modulin n tekijöihin jako onnistuu, niin avaimet saadaan käden käänteessä noudattaen yleisen algoritmin kuvausta tai sitä seurannutta esimerkkiä. Entä kääntäen, pitääkö paikkansa, että jos suuren luvun tekijöihin jako on vaikeaa, niin RSA:n murtaminen on vaikeaa? Ongelmaa on tutkittu tiiviisti RSA:n julkistamisesta lähtien, asiaa ei ole pystytty todistamaan, mutta mitään muuta tapaa menetelmän murtamiseen ei myöskään ole löydetty. Lainaan [I. Algo]-kirjaa s. 835: Jos valitaan satunnaisesti kaksi 100-numeroista alkulukua, niin niiden avulla voidaan muodostaa avain, joka on "murtamaton" nykyteknologialla (v. 1998).

Kuitenkin on erinäisiä seikkoja, jotka täytyy ottaa huomioon, sillä koodinmurtajien työkalupakissa on taatusti ainakin modulaariaritmetiikan peruslauseet, Fermat'n pikku lause, Eukleideen algoritmi, Kiinalainen jäännöslause ja parhaat laskenta-algoritmit. Lisäksi raakaa laskentavoimaa on oletettava olevan suurimman supertietokoneen verran ja tehokkaasti hajauttaen paljon enemmänkin.

[HandB] käsittelee aihetta laajasti esittäen ainakin 8 erilaista hyökkäysmahdollisuutta ja niiden torjuntalääkkeet. Monissa muissa lähteissä, kuten [Nyberg], [Wiki] ja aivan erityisesti [Sti] (s. 225) on lisää kryptoanalyyttisiä konnankoukkuja ja niiden vastalääkkeitä.

Pari yksinkertaisinta seikkaa mainitakseni, on selvää, että alkulukujen p ja q on molempien oltava niin suuria, ettei pienistä luvuista lähtevä alkulukujen laskenta ja tekijätarkistus onnistu kohtuuajassa. Toisaalta ne eivät saa olla niin lähellä toisiaan, että voitaisiin tekijän etsintä aloittaa $\sqrt{n}$:n läheltä.
Pieni salauseksponentti e on salaamisen kannalta tehokas, mutta ongelmallinen, jos viesti (tai viestin osa) m on niin pieni, että $m < n^{1/e}$. Tällöinhän salaviesti $c = m^e \pmod n = m^e$ (mieti!). Murtaja *Eve* muodostaa luvun $c^{1/e}$, ja lukee m:ää kuin avointa kirjaa. Tämä voidaan estää lisäämällä viestiin riittävästi "suolaa", eli ylimääräistä puppua.

Kryptologia on aina ollut kahden joukkueen välistä kilpajuoksua. Rauhanomaiseen kilpailutoimintaan liittyy RSA129-projekti, joka lähti liikkeelle, kun RSA:n keksijät Rivest, Shamir ja Adleman esittivät vuonna 1977 Scientific American-lehdessä 129-numeroisen kahden alkuluvun tulon haasteeksi tiedeyhteisölle tekijöiden löytämistä varten. He arvioivat tekijöihin jakoon kuluvan aikaa n. 20000 vuotta silloisilla menetelmillä ja tekniikalla. Hollantilainen lukuteoreetikko *Arjen Lenstra* organisoi maailmanlaajuisen laskentaverkoston
http://www.math.okstate.edu/∼wrightd/numthry/rsa129.html
Lenstran ryhmä käytti 1980-luvulla kehitettyä uutta lukuteoreettista menetelmää ja kykeni purkamaan algoritmin rinnakkaislaskentaa hyödyntävään muotoon. Huhtikuussa 1994 tekijöihin jako onnistui, ja RSA:lla salattu viesti saatiin auki. Viestin sisältö oli: "The magic words are squeamish ossifrage."

Kilpajuoksu ei päättynyt tähän. [Wiki]:ssä mainitaan luku RSA-200, joka jaettiin tekijöihin vuonna 2005. Kirjassa [Sti] (v. 2006) s. 175 mainitaan. että nykyiset tekijöihinjakoalgoritmit löytävät 512:n pituisen binääriluvun tekijät. Luvun pituus kymmenjärjestelmässä saadaan kertomalla $\log_{10} 2$:lla (eikö

vain), joten se on n. 150 numeroa. Turvalliseksi luvun $n = pq$ suuruudeksi *Stinton* vakuuttaa nykytietämyksellä 1024 bittiä, siis n. 300 numeroa 10-järjestelmässä.

Ajantasaista tietoa voi hakea [Wiki]:sta sanoilla "RSA Factoring challenge", joka kertoo, että tämä kilpailumuoto lopetettiin vuonna 2007, koska "nykyteollisuudella on aiempaa huomattavasti kehittyneempi ymmärrys yleisistä kryptoanalyyttisistä periaatteista".

Tulevaisuuden näkymiä

Edellä nähtiin, että arviot avainten turvalliseen kokoon liittyvistä vaatimuksista tahtovat jäädä jälkeen siitä, minkä tänään oletetaan riittävän pitkälle tulevaisuuteen. Tästä syystä ei voida pysähtyä lepäämään RSA-laakereilla, vaan on välttämätöntä kehittää uusia "loukkuluukkuideoita".

Kryptologiset menetelmät ovat laajentuneet abstraktia algebraa, algebrallista geometriaa, elliptisiä käyriä, ym. kehittyneitä moderneja matemaattisia teorioita käyttämään. Lisäksi deterministisen menetelmien ohella on ryhdytty kehittämään myös todennäköisyyslaskentaan pohjautuvia satunnaisuuteen perustuvia menetelmiä.

Kryptologiassa yhdistyvät kiehtovalla tavalla vuosituhansien aikana kehittyneet lukuteorian menetelmät uusien abstraktien matemaattisten teorioiden tarjoamiin mahdollisuuksiin. Tärkeänä komponenttina on tietotekniikka teoreettisena työvälineenä, tehokkaan laskentavälineistön mahdollistajana, ja tietysti syyn ja motiivin antajana koko toiminnalle tietoverkkojen, matkapuhelimien, sirukorttien maailmassa.

Viitteet

[I. Algo] Cormen, Leiserson, Rivest: Kts. Osa 1

[DH] W. Diffie, M. Hellman. New directions in cryptography, IEEE Transactions on Information Theory IT-22 (1976), 644-654.

[HandB] A. Menezes, P. van Oorschot, S. Vanstone. Handbook of applied cryptography: `http://www.cacr.math.uwaterloo.ca/hac/`.

[CodeB] D. Kahn, The Codebreakers, Macmillan 1967, kirjasta on uudempi painos.

[Crt] Bernhard Esslinger. Cryptography and Mathematics, `http://www.cryptool.com/` . Vapaan lähdekoodin ohjelmapaketti ja opetusteksti.

[InCode] Sarah Flannery, David Flannery. In Code: A Mathematical Journey.
Lukuteorian ja kryptologian periaatteiden yleistajuinen esitys nerokkaan irlantilaisen koulutytön ja isän kirjoittamana. `http://www.amazon.com/` [Hakusana In Code] (Ehkä tästä joku koulutyttö tai poika voisi innostua vaikka kirjoittamaan kirja-arvion.)

[Cos] John B. Cosgrave. Bill Clinton, Bertie Ahern, and digital signatures (A Maple-based introduction to public-key cryptography) `http://staff.spd.dcu.ie/johnbcos/Maple_public_other.htm`

[Ke-Te] Veikko Keränen, Jouko Teeriaho. Salausmenetelmät, Rovaniemen AMK 2006: `http://ta.ramk.fi/~jouko.teeriaho/krypto2006/krypto.htm`. Kurssimateriaali, käyttää hyväkseen MATHEMATICA-ohjelmaa.

[Skk] Simo Kivelä. RSA-menetelmän MATHEMATICA-toteutus.
`http://matta.hut.fi/matta2/mma/rsa.pdf`

[Kob] N. Koblitz. A Course in Number Theory and Cryptography, Springer 1994.

[KN] Kaisa Nyberg. Kryptologia – tiedon turvaamisen tiede, Tietojenkäsittelytiede **26** kesäkuu 2007 ss. 31–52.

[JP] Jukka Pihko, Lukuteorian helmiä, `solmu.math.helsinki.fi/2008/1/pihko.pdf`

[RSA] R. Rivest, A. Shamir, L. Adelman. A method for obtaining digital signatures and public-key cryptosystems, Communications of the ACM, 21 (1978), 120-126. Luettavissa tästä: `theory.lcs.mit.edu/~rivest/rsapaper.pdf`

[AS] Arto Salomaa. Public-Key Cryptography, Springer-Verlag, Berlin 1990.

[Sti] D. R. Stinson. Cryptography, Theory and Practice, Chapman & Hall/CRC Press, Boca Raton-London-New York, Third Edition, 2006
Kurssikirjaviitteenä Kaisa Nybergin (TKK) ja Keijo Ruohosen (TTY) opetussivuilla. (Kirjassa 353 viitettä.)

[Wiki] `http://en.wikipedia.org/wiki/`
Hakusanoja: *Cryptography, History of cryptography, World_War_II_cryptography, Enigma machine, Quantum Cryptography, RSA, RSA Factoring challenge,*