Aiheet:

# Julkisen avaimen salausjärjestelmät

Public key encryption

# Julkisen avaimen salaus: public key encryption

Idean julkisen avaimen salauksesta esittivät v. 1977 **Diffie ja Hellman**

Periaate:
Jokaisella käyttäjällä X on kaksi avainta:

- julkinen avain, jolla X:lle lähetettävät viestit salataan
- yksityinen avain, jolla X purkaa saamansa viestit

**Viestit salataan käyttämällä** viestin **vastaanottajan julkista avainta**. Julkinen avain saadaan tavallisesti avainpalvelimelta (CA), joka pitää yllä julkisten avainten rekisteriä. Viestin vastaanottaja purkaa salauksen käyttämällä yksityistä avaintaan.

Avainparia voidaan käyttää myös toisinpäin: yksityisellä avaimella salattu viesti voidaan purkaa julkisella avaimella. **Autentikoinnissa eli käyttäjän todennuksessa** A varmistuu B:n identiteetistä lähettämällä tälle satunnaisluvun R, johon B vastaa lähettämällä luvun salattuna yksityisellä avaimellaan. A purkaa saamansa vastauksen B:n julkisella avaimella. Jos purettu vastaus sisältää alkuperäisen satunnaisluvun R, B:n aitous on varmistettu.

# RSA – salausalgoritmi

Ensimmäisen toimivan julkisen avaimen salausmenetelmän esittivät

Rivest, Shamir ja Adlemann v. 1977.  RSA – salaus on tätä kirjoitettaessa (toukokuussa 2019) edelleen käytössä osana mm. Suomen verkkopankkien salausta.

Rivest, Shamir ja Adlemann 1970  -luvulla

# RSA:n avaimet

Jokaisella käyttäjällä on kaksi avainta:

**Julkinen avain sisältää kaksi kokonaislukua**

**modulus n = p*q, missä p ja q ovat alkulukuja**

**eksponentti e = 65537** **(TLS:ssä sama kaikille käyttäjille *)**

**Yksityinen avain d = $e^{-1}$ mod (p-1)(q-1)**

ts. d = eksponentin e käänteisluku modulo (p-1)(q-1)

Huom! Matematiikkaosassa on käsitelty Eulerin funktiota φ. Kun n on kahden alkuluvun p ja q tulo, niin φ(n) = (p-1)(q-1)

*) Yleisessä tapauksessa eksponentti e voi olla käyttäjäkohtainen, sen ei siis tarvitse olla vakio 65537. Ainoa ehto on, että e:llä pitää olla käänteisluku mod (p-1)(q-1), mikä toteutuu jos GCD(e, (p-1)(q-1))=1

# RSA-avainten generointi WolframAlpha Online laskimella

1. Luodaan 2 alkulukua p ja q, joiden pituus on tässä esimerkissä 15 bittiä.
   Lasketaan julkinen avain kaavalla n = p q

```
p=RandomPrime[{2^15,2^16}]; q=RandomPrime[{2^15,2^16}]; n=p*q
```

$$p = 59\,023\,, \quad q = 43\,313\,, n = 2\,556\,463\,199$$

2. Lasketaan yksityinen avain d kaavalla $d = e^{-1} \bmod (p-1)(q-1)$, missä e=65537

```
p=59023; q=43313;  65537^-1 mod (p-1)(q-1)
```

d = $1\,626\,331\,841$

Avainpari on siten: Julkinen avain 2 556 463 199, yksityinen avain 1 626 331 841

# Viestin salaus ja purku RSA:lla

## Viesti esitetään kokonaislukuina m

**Salaus tapahtuu käyttämällä vastaanottajan julkisia avaimia**

$$c = m^e \mod n$$

**Vastaanottaja purkaa salakirjoituksen c yksityisellä avaimellaan d**

$$m = c^d \mod n$$

# Salaus ja purku laskuesimerkki

Bobin RSA avaimet:  Julkinen avain n= 2 556 463 199,  yksityinen d = 1 626 331 841
Laske Bob:lle lähetettävän viestin m = 12345  salakirjoitus. Näytä miten Bob purkaa sen.

Salaus (kaava $c=m^e \bmod n$)

```
12345^65537 mod 2556463199
```

1 706 161 508

Purku (kaava $n=c^d \bmod n$)

```
1706161508^1626331841 mod 2556463199
```

12 345

# Webpalvelimen autentikointi RSA:lla

**TLS protokollassa RSA:n tärkein tehtävä liittyy palvelimen todennukseen, jossa palvelin todistaa aitoutensa käyttämällä yksityistä RSA- avaintaan -** siis päinvastoin kuin tavanomaisessa viestin salauksessa, jossa salaukseen käytetään julkista avainta.

Oletetaan, että palvelimen julkinen avain on n, yksityinen avain on d ja vakioeksponentti e = 65537

1. Asiakkaan selain lähettää satunnaisluvun R (ns. Haasteluku) palvelimelle.

2. Palvelin lähettää vastauksen RES = $R^d$ mod n
   ts. palvelin salaa saamansa haasteluvun R yksityisellä avaimellaan d

3. Asiakas lukee palvelimen sertifikaatista palvelimen julkisen avaimen n ja purkaa vastauksen RES potenssiin korotuksella $RES^e$ mod n.
   Mikäli tulos on R, palvelin on todennettu

# Palvelimen autentikointi RSA:lla: esimerkki

Palvelimen avaimet: e = 65537 , d = 1 626 331 841 ja n = 2 556 463 199

1. Asiakas lähettää haasteluvun R = 112233

2. Palvelin lähettää vastauksen RES = $R^d$ mod n

112233^1626331841 mod 2556463199

2017034810

3. Asiakas purkaa vastauksen: $RES^e$ mod n ja vertaa tulosta lähettämäänsä haastelukuun.

2017034810^65537 mod 2556463199

112233

Tulos täsmää lähetetyn haasteluvun kanssa => palvelin on autentikoitu

# RSA:n matematiikkaa

1. Nopea potenssiinkorotus (PowerMod) ja käänteisluvun laskeminen (ExtendedGDC)
   - esitetty matem.osuudessa
2. Satunnaislukujen generointi  (tarvitaan mm. haastelukujen luonnissa)
3. Alkulukutestit ja alkulukujen generointi (julkisen avaimen luonnissa)
4. Eulerin teoreema (RSA:n oikeaksi todistaminen)
5. RSA:n turvallisuus ja suurten kokonaislukujen faktoroinnin kompleksisuus

# Viestin koodaus kokonaisluvuksi ja päinvastoin

**Tekstilohkot (esim. lohkonpituus 8 merkkiä) muutetaan kokonaisluvuksi siten, että muunnetaan ensin merkit niiden ASCII – koodeiksi. Tämän jälkeen ASCII koodeista muodostetaan 256 – kantajärjestelmän lukuja seuraavalla tavalla.**

## Viestin ”Helsinki” koodaus kokonaisluvuksi

**Viestin ”Helsinki” merkkien koodit ovat {72, 101, 108, 115, 105, 110, 107, 105} . Tästä saadaan kokonaisluku**
**$m = 72*256^7 + 101*256^6+108*256^5+115*256^4+105*256^3+110*256^2+107*256+105 = 52166949863244706 33$**

## Kokonaislukuesityksen dekoodaus tekstimuotoon

**Luku 52166949863244706 33 esitetään 256 –kantaisessa lukujärjestelmässä. Käsin tehtynäkin tämä onnistuu, mutta esim. WolframAlphassa muunnoksen voi tehdä komennolla IntegerDigits[52166949863244706 33, 256] ,**
**joka antaa viestin merkkien ASCII koodit {72, 101, 108, 115, 105, 110, 107, 105}**

**ASCII koodeja vastaava merkkijono on ”Helsinki”**

WolframAlpha.com :ssa koodauksen ja dekoodauksen voi tehdä seuraavilla komennoilla. Kokeile!

```
FromDigits[ToCharacterCode["Helsinki"],256]
```

tulos : 52166949863244706 33

```
FromCharacterCode[IntegerDigits[52166949863244706 33,256]]
```

tulos : Helsinki

# Katsaus RSA:n taustalla olevaan matematiikkaan

# Nopea potenssiin korotus ”Powermod”

$$7^{11} \bmod 53$$

$$7^{10} \cdot 7 \bmod 53$$

$$49^{5} \cdot 7 \bmod 53$$

$$49^{4} \cdot 49 \cdot 7 \bmod 53$$

$$2401^{2} \cdot 343 \bmod 53$$

$$16^{2} \cdot 25 \bmod 53$$

$$256 \cdot 25 \bmod 53$$

$$44 \cdot 25 \bmod 53$$

$$1100 \bmod 53 = 40$$

Toistettava silmukka:

Jos eksponentti on **parillinen**, puolita eksponentti ja neliöi kantaluku:

**$a^b \bmod n = (a^2)^{b/2} \bmod n$**

Jos eksponentti on **pariton**, kirjoita potenssi muodossa

**$a^b \bmod n = a^{b-1} . a \bmod n$**

**Voidaan osoittaa, että potenssiin korotukset mod n voidaan toteuttaa muistissa, jonka koko on $n^2 + 3n$**

**Esim. RSA:ssa, jossa avain on 2048 bittiä, potenssiin korotukset vaativat muistia n. 4048 bittiä, joka on 506 kB**

# SATUNNAISLUKUJEN GENEROINTI

Salausavaimet luodaan satunnaislukugeneraattorilla

- ovatko tuotetut satunnaisluvut todella satunnaisia?
- Ovatko jotkin satunnaisluvut todennäköisempiä kuin toiset?
  (avainten murtaminen nopeampaa, jos näin on)
- Voiko satunnaislukugeneraattoreihin sisältyä tietoturvauhka esim. takaportin muodossa?

# Mikä on satunnaisluku?

**Näyttääkö tämä satunnaisluvulta ?**
**10010110101011000110100101101101110101001100101010100000110110011**

**Solomon W Golomb on määritellyt satunnaisen bittijonon seuraavasti. (Nämä ominaisuudet esiintyvät esim. kolikon heiton tuloksissa)**

**G1. Bittijonossa on n. 50% ykkösiä ja 50% nollia**

**G2. Yhden bitin , kahden saman bitin, kolmen saman bitin ja neljän saman bitin jaksojen suhteelliset esiintymistodennäköisyydet jonossa suhtautuvat toisiinsa kuten 1/2 , 1/4 , 1/8 and 1/16**

**G3. Kun jonoon tehdään bittien rotaatio (k askelta) ja verrataan alkuperäistä ja uutta jonoa biteittän toisiinsa, samojen bittien määrä tulisi olla n. 50% .**

k:n pituisella jaksolla tarkoitetaan toistuvaa saman bitin jaksoa
Jota reunustavat toisenlaiset bitit.

$$0\,\underbrace{11\ldots1}_{k}\,0 \quad \text{or} \quad 1\,\underbrace{00\ldots0}_{k}\,1$$

# Salauksessa tarvitaan vielä lisäominaisuuksia

4. Kaikilla satunnaislukugeneraattoreilla on jokin periodi. (Ne siis tuottavat samat luvut periodin täytyttyä). **Periodin tulisi olla hyvin suuri**

5. Satunnaislukujen generoinnin pitää olla **hyvin nopeaa**

6. Jostakin generoidusta **satunnaisluvusta** ei saa laskea edeltäviä tai seuraavia lukuja.

## Voiko satunnaislukugeneraattori olla tietoturvauhka?

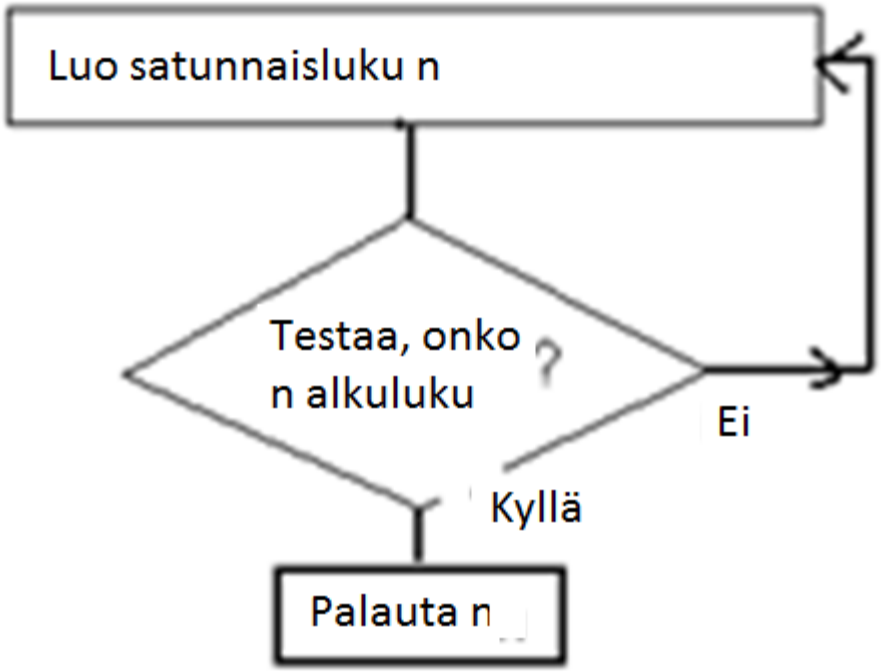
- **Jos jotkut ”satunnaisluvut” ovat todennäköisempiä kuin toiset, ja tämä on tiedossa, tämä lyhentää avaimen murtamisaikaa**

## Ovatko satunnaislukugeneraattoriin liittyvät uhkat toteutuneet?

- **Dual ECDRPG** oli standardoitu satunnaislukugeneraattori OpenSSL vuoteen 2014. Siinä oli takaportti, jonka Eduard Snowden paljasti.

=> Kyseinen generaattori vedettiin 2014 pois standardeista

# Alkulukugenerointi tapahtuu generoimalla satunnaislukuja ja tekemällä niille alkulukutesti.



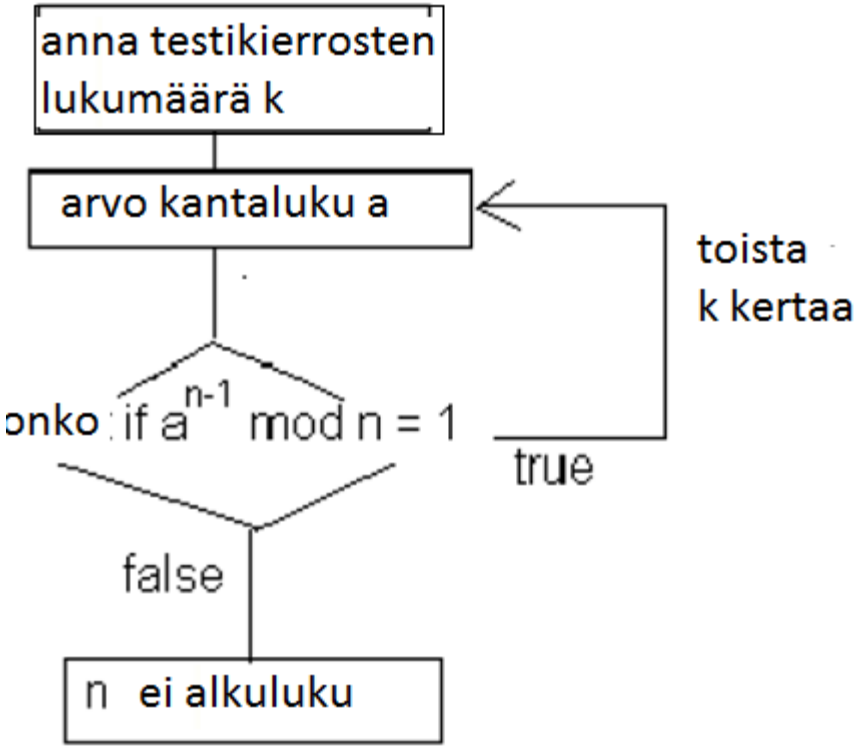

**ALKULUKUTESTIT**

1. Rabin – Miller testi
2. Fermat'n testi (PGP)

# Fermat’n alkulukutesti perustuu Fermat’n teoreemaan

**Jos p on alkuluku, $a^{p-1} \bmod p = 1$ kaikille $0 < a \leq p-1$**

**Fermat’n alkulukutesti**




**Testi on probabilistinen:**

**Jos jollekin kantaluvulle $a^{n-1} \bmod n \neq 1$ , niin n on varmasti yhdistetty luku.**

**Jos jollekin kantaluvulle $a^{n-1} \bmod n = 1$ , ei se vielä takaa, että n on alkuluku. Testiä pitää jakaa muilla kantaluvuilla.**

**Jos Fermat’n testi menee läpi k:lla kantaluvulla, ei ole silti tiedossa tarkkaa todennäköisyyttä sille, että n olisi alkuluku.**

Kuuluisa PGP salausohjelma käyttää tätä testiä

# ESIM: OSOITETAAN, ETTÄ LUVUT a) 4763 b) 561 EIVÄT OLE ALKULUKUJA

**Testataan luku 4763**

```
2^4762 mod 4763

Result:

158
```

⇒ **4763 on yhdistetty luku**

**Testataan luku 561**

```
2 ^560 mod 561

Result:
 1

test passed
```

```
5 ^560 mod 561

Result:
 1

test passed
```

```
3^560 mod 561

Result:
375

test failed =>
561 is composite
```

Luku 561 kuuluu ns. Charmicaelin lukuihin, joille Fermat'n testi saattaa mennä läpi useilla kantaluvuilla.

Tässä tapauksessa testi menee läpi kantaluvuilla 2 ja 5.

Kantaluku 3 kuitenkin osoittaa, että luku on yhdistetty luku, eikä alkuluku

# Rabin Miller testi : positiivisen tuloksen luotettavuus arvioitavissa

Perustuu faktaan, että jos p on alkuluku, luvun 1 neliöjuuri mod p voi olla vain 1 tai -1 (eli p-1)

## Rabin Miller testi, yksi kierros

p = testattava luku
1. Valitaan satunnainen kantaluku a
a. Suoritetaan Fermat'n testi :
Jos $a^{p-1} \bmod p = 1$
b. Otetaan neliöjuuri vasemmasta puolesta:
ts. lasketaan

$$a^{(p-1)/2} \bmod p$$

Jos tulos = p- 1 , luku p läpäisee testin
Jos tulos = 1, jatketaan taas ottamalla neliöjuuri ts. Lasketaan

$$a^{(p-1)/4} \bmod p$$

Jos tulos = p- 1 testi on läpi, jos tulos = 1, otetaan taas neliöjuuri. Eksponenttia puolitetaan, kunnes se on pariton. Mikä tahansa muu tulos kuin 1 tai p-1 merkitsee yhdistettyä lukua.

**Tämäkin testi on probabilistinen:**

**Jos testi ei mene läpi jollakin kantaluvulla, luku on varmasti yhdistetty luku**

**Jos ehdokas n läpäisee testin k:lla satunnaisella kantaluvulla a, todennäköisyys sille, että n on alkuluku P $> 1 - 1/4^k$**

**Tässä testissä 10 läpäistyä kierrosta antaa varmuuden 99.9999% siitä, että luku on alkuluku.**

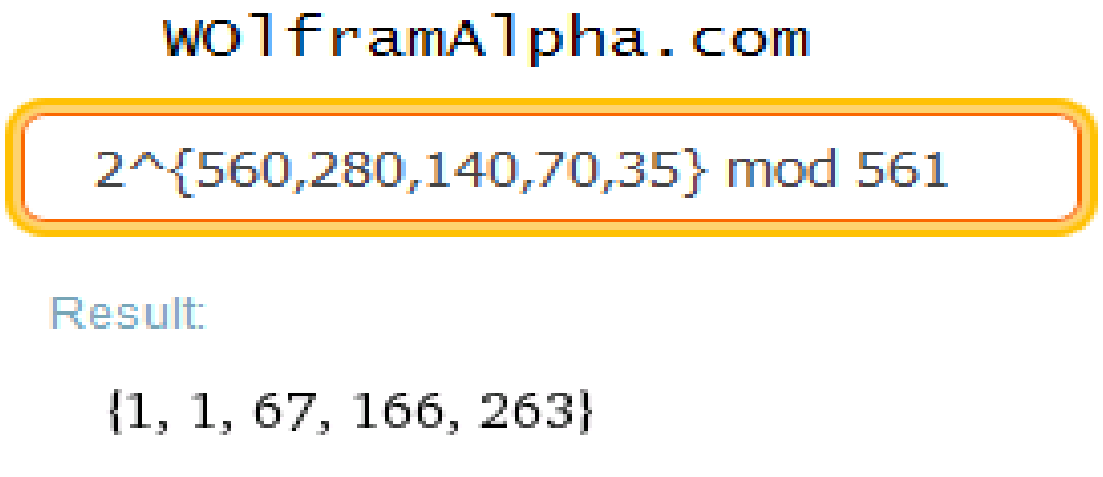
# Testataan luku 561 Rabin Miller testillä

Tutkitaan, kuinka monta kertaa p – 1 voidaan puolittaa:

**560** = $2 \cdot 280 = 2^2 \cdot 140 = 2^3 \cdot 70$ **= $2^4 \cdot 35$**

Valitaan kantaluvuksi a = 2:
**Lasketaan jono:**
**$2^{560}$ mod 561 = 1**
**$2^{280}$ mod 561 = 1**
**$2^{140}$ mod 561 = 67** **=> fail**
**$2^{70}$ mod 561 =**
**$2^{35}$ mod 561 =**

WolframAlphassa voidaan suorittaa yhdellä komennolla useita potenssiinkorotuksia eri eksponenteilla:




Koska kolmas potenssiin korotus antoi 67, joka ei ole 1 eikä 560, niin
**561 on yhdistetty luku, eikä alkuluku**

# Testaa ”Rabin Miller testillä” luku 1973

Askel 1: **$p-1 = 1973-1 = 1972 = 2^2 * 493$** (voidaan puolittaa kahdesti)

Askel 2: Suoritetaan Rabin Miller testi neljällä satunnaisella kantaluvulla: 2 , 35 , 854 ja 114

**Kantaluku 2:**
**$2^{1972} \bmod 1973 = 1$**
**$2^{986} \bmod 1973 =$ 1972**
**Testi läpäisty**

**Kantaluku 35:**
**$35^{1972} \bmod 1973 = 1$**
**$35^{986} \bmod 1973 = 1$**
**$35^{493} \bmod 1973 = 1$**
**Testi läpäisty**

**Kantaluku 854:**
**$854^{1972} \bmod 1973 = 1$**
**$854^{986} \bmod 1973 =$ 1972**
**Testi läpäisty**

**Kantaluku 114:**
**$114^{1972} \bmod 1973 = 1$**
**$114^{986} \bmod 1973 = 1$**
**$114^{493} \bmod 1973 = 1$**
**Testi läpäisty**

**Todennäköisyys että 1973 on alkuluku on**
**$P > 1 - 4^{-4} = 0.996 = 99.6\%$**

# Alkulukugenerointi WolframAlphassa

**Ohjelmointikielissä, joilla salausohjelmia kirjoitetaan on yleensä funktio, jolla voi generoida halutun bittimäärän pituisia alkulukuja**

## Esim: Luo 100 –bittinen alkuluku

```
RandomPrime[{2^100,2^101}]
```

2 422 530 443 145 414 600 337 950 658 763

## Esim: Luo 512 –bittinen alkuluku

```
RandomPrime[{2^512,2^513}]
```

15 787 372 807 814 935 269 337 946 439 168 767 722 475 175 033 260 767 647 751 154 530 ⋱
333 820 130 747 181 919 458 745 158 006 634 759 921 476 598 518 589 413 311 054 638 ⋱
013 394 363 696 097 684 553 327 539

# RSA:n kaavojen matemaattinen perustelu nojaa Eulerin teoreemaan

$$a^{\varphi(n)} \bmod n = 1$$

n = positiivinen kokonaisluku
a = mikä tahansa kokonaisluku , jolle GCD(a,n) = 1
φ(n) = Eulerin funktio luvulle n = niiden kokonaislukujen määrä välillä 1 – (n-1) , joilla ei ole yhteisiä tekijöitä n:n kanssa.

**Matematiikkaosuudessa on esitetty Eulerin funktion φ(n) ominaisuuksia:**

- 1) **Alkuluvuille φ(p) = p – 1**
  **esim: φ(13) = 12**
- **2) Kahden alkuluvun tulolle φ(p*q) = (p-1)(q-1)**
  **esim: φ(21) = φ(3*7) = 2*6 = 12**
- **3) Yleisesti φ(n) = n (1-1/$p_1$)(1-1/$p_2$)... , missä $p_1$, $p_2$, ... ovat luvun n eri alkulukutekijät**

**esim: φ(45) = 45*(1-1/5)*(1-1/3) = 45*4/5*2/3= 24**

# RSA:ssa julkinen avain n = p*q (p,q alkulukuja)

Tällöin Eulerin funktio $\varphi(n) = \varphi(pq)=(p-1)(q-1)$

**Jos yksityinen avain $d = e^{-1} \mod (p-1)(q-1)$ ,**
**niin $e\,d = 1 \mod (p-1)(q-1) \Rightarrow e\,d = 1 + k*(p-1)(q-1)$**

Seuraavassa osoitetaan, miksi potenssiinkorotukset eksponentteina e ja d ovat toistensa käänteisoperaatioita

$$(m^{e})^{d} \mod n$$
$$= m^{e\,d} \mod n$$
$$= m^{1 + k\,(p-1)(q-1)}$$
$$= m * (m^{(p-1)(q-1)})^{k}$$
$$= m*1 = m$$

$m^{(p-1)(q-1)} = m^{\varphi(n)} = 1$

# RSA:n turvallisuus perustuu suurten alkulukujen tekijöihin jaon vaikeuteen

Suurten kokonaislukujen tekijöihinjako kuuluu matematiikan ”hard problems” luokkaan. Nopeimmat faktorointimetodit ovat ”Guadratic Number Field Sieve” and GNFS (General Number Field Sieve).

**Suurin faktoroitu RSA:n julkinen avain (n = p*q) on RSA-768 , (768 - bittinen luku). Menetelmä oli GNFS. Murtaminen kesti 2 vuotta satojen koneiden verkolla.**

**RSA-768** = 123018668453011775513049495838496272077285356959533479219732245215172640050726365751874520219978646938995647494277406384592519255732630345373154826850791702612214291346167042921431160222124047927473779408066535141959745985 6902143413

# Turvallisen julkisen RSA-avaimen  minimipituus

Jos hyökkääjä kykenee jakamaan palvelimen julkisen avaimen n sen tekijöihin p ja q, hän kykenee laskemaan helposti palvelimen yksityisen avain d.  Tällöin palvelimen tietoliikenteen salaus on murrettu.

Esimerkiksi julkinen avain n = 502560280658509 on aivan liian lyhyt. Mm. Wolfram Alpha antaa sen tekijät.

| factor | 502 560 280 658 509 |
|---|---|

p q

15485863 × 32452843

Luvuista p ja q  on helppo laskea yksityinen avain kaavalla $d = e^{-1} \bmod (p-1)(q-1)$

**TURVALLINEN RSA:N AVAINPITUUS**

- RSA on vielä yleisesti käytössä TLS yhteyksissä, mm.  Suomen verkkopankkien julkiset avaimet ovat RSA avaimia, joiden pituus on 2048 bittiä.
- Tätä on pidettävä turvallisen avaimenpituuten minimirajana.

# ECC  (Elliptic Curve Cryptosystem)

= Julkisen avain salaus, jota mm. Suomen viranomaiset suosittelevat  RSA:n seuraajaksi, on ECC

ECC on jo käytössä mm. Suomen verkkopankkien TLS – yhteyksien ECDHE -protokollassa, jolla sovitaan istunnon AES -avaimesta.

Verkkopankkipalvelimien julkinen avain näyttää olevan vielä tänään (17.6.2019) 2048-bittinen RSA avain.

Elliptisten käyrien salauksesta kerrotaan tarkemmin seuraavassa luennossa, jonka aihe on avaimesta sopimisen protokollat

# PK-salauksen turvalliset avainpituudet

Alla on EU:n salausmenetelmätyöryhmän arvioita RSA:n ja seuraajaksi nimetyn ECC:n turvallisuudesta eri avainpituuksilla. Taulukko osoittaa että ollakseen turvallinen, RSA tarvitsee yli 2000 bitin julkisen avaimen, kun taas ECC –salauksessa riittää 256 bittiä.

| Kuvaus | RSA | ECC |
|---|---|---|
| Voidaan murtaa perustekniikoilla | 816 bits | 128 |
| Voidaan murtaa lyhyessä ajassa | 1008 | 144 |
| Teoriassa riittävä | 1248 | 160 |
| Yleisesti katsotaan ehdottomaksi minimiksi | 1776 | 192 |
| Minimitason takaava turvallisuus | 2432 | 224 |
| Riittävä taso paitsi huippusalaisia asiakirj. | 3248 | 256 |
| Riittävä myös top secret asiakirjoihin | 15424 | 512 |

Älykorteissa ja mobiililaitteissa on RSA – avaimia. Kasvava avainpituus on ongelma. Muisti ei riitä ja laskenta on hidasta.

Elliptic curve cryptosystem ECC on pienemmän avainpituuden vuoksi suositus tulevaisuuden julkisen avaimen salaukseksi.