## Yksisuuntaiset funktiot

- Mahdollisia ominaisuuksia:

  **Kompressio**: **hajautusfunktio** ottaa argumentikseen mielivaltaisen pituisen sanoman ja palauttaa vakiomittaisen hajautusarvon.

  **Yksisuuntaisuus**: funktion arvo on helppo laskea, mutta käänteisfunktion laskeminen on käytännössä mahdotonta.

  **Heikko törmäyksettömyys**: jos on annettu argumentti ja sitä vastaava funktion arvo, ei ole ei ole mahdollista löytää toista argumenttia, joka tuottaa saman funktion arvon.

  **(Vahva) törmäyksettömyys** (*collision resistance*) : ei ole mahdollista löytää kahta argumenttia, jotka tuottavat saman funktion arvon.

  **Avaimellisuus**: funktion arvon pystyy laskemaan vain salaisen avaimen haltija, eikä tunnetusta argumentti-arvoparista ole mahdollista päätellä muita pareja avainta tuntematta.

  **Salaovi**: käänteisfunktion pysty laskemaan vain salaisen avaimen haltija.

- Käänteisfunktio (ei yksikäsitteinen) ja törmäyksiä tietenkin on olemassa. Niiden laskeminen on olemassa olevilla resursseilla mahdotonta.



## Kryptografiset hajautusfunktiot

(*one-way collision-resistant hash, cryptographic hash, secure hash, message digest, fingerprint*)

- **Kompressio**:
  - Muuttaa mielivaltaisen sanoman kiinteän mittaiseksi hajautusarvoksi,
  $$h = H(M).$$

  **Yksisuuntaisuus**:
  - Hajautusarvo $h$ on helppo laskea sanomasta $M$.
  - Jos $h$ tunnetaan, on vaikea löytää sanomaa $M$, jolla $H(M) = h$.

  **Törmäyksettömyys**:
  - On vaikea löytää kahta eri sanomaa $M$ ja $M'$, joille $H(M) = H(M')$.

- **Käytetään sanoman muuttumattomuuden tarkistamiseen.** Jos sanoma muuttuu, muuttuu myös hajautusarvo.

  Lyhyt hajautusarvo on myös helpompi allekirjoittaa sähköisesti kuin koko sanoma.

- Laskennallisessa mielessä hajautusarvo sisältää alkuperäisen sanoman koko redundanssin tiiviisti pakattuna.



## Kompressiofunktio

- Kryptografisen hajautusfunktion rakenne käytännössä:

  **kompressiofunktio**, joka tiivistää pitemmän kiinteämittaisen lohkon lyhyemmäksi. Yhdistetään lohko kerrallaan lohkon sisältö edellisen lohkon hajautusarvoon.

  $$h_{i+1} = f(M_i, h_i).$$


- Sama idea kuin lohkosalauksessa: iteroidaan epälineaarisia funktioita. Koska ei tarvitse olla bijektio, voidaan funktiot valita vapaammin $\Longrightarrow$ yksinkertaisempi, nopeampi.

  Salaus on mahdollisimman epälineaarinen (satunnaisen näköinen) avaimesta riippuva permutaatio. Käänteisfunktio on mahdollista laskea salaisen avaimen avulla.

  Yksisuuntainen hajautusfunktio ei ole permutaatio eikä bijektio, se ei riipu avaimesta, eikä käänteisfunktiota ole mahdollista laskea.



## Yksisuuntaiset funktiot ja salaus

- Lohkosalauksesta saadaan yksisuuntainen hajautus, esim.

  $$h_i = E_{M_i}(h_{i-1}) \oplus M_i.$$

  Huom: 64-bittinen tulos on usein liian lyhyt.

- Yksisuuntaisesta hajautuksesta saadaan salaus käyttämällä CFB-moodia (*Ciphertext-Feedback*) .

  $$C_i = P_i \oplus H(K, C_{i-1})$$
  $$P_i = C_i \oplus H(K, C_{i-1})$$

  Yksisuuntaista hajautusfunktiota voidaan käyttää sekoitusfunktiona Feistel-salauksessa (**Luby–Rackoff**):

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

  Huom: hajautusfunktio on altis differentiaaliselle kryptoanalyysille.

- Näin vain teoriassa. **Käytännössä kannattaa käyttää kuhunkin tarkoitukseen siihen suunniteltuja algoritmeja.**

## Syntymäpäivähyökkäys

- Jos kahdella ihmisellä on sama syntymäpäivä, sitä pidetään suurena yhteensattumana. Kuitenkin jos ryhmässä on 23 ihmistä, yli 50 % todennäköisyydellä kahdella heistä on sama syntymäpäivä.

[graafi]

- Jos $N$ alkiosta tehdään $k$ riippumatonta valintaa, ovat kaikki valinnat eri alkioita todennäköisyydellä

$$p = \frac{N!}{(N-k)!\,N^k}.$$

  Yhteensattuman todennäköisyys on noin $1/2$, kun $k = \sqrt{N}$.
- Hajautusarvon pituuden oltava vähintään $2 \cdot 64 = 128$ bittiä.



## Hajautusfunktioiden käyttö

- MD5 (Message Digest): 512-bittiset blokit, 128-bittinen hajautusarvo, **ei pidetä enää turvallisena**
- SHS (Secure Hash Standard): 512-bittiset blokit, 160-bittinen hajautusarvo
- Parempi käyttää kuhunkin tarkoitukseen siihen erityisesti suunniteltua algoritmia.
  - Väärässä käytössä paljastuu uusia heikkoiksia.
  - Tehokkuus–turvallisuussuhde optimoitu tiettyyn käyttöön.
  - Blokkikoko $\neq$ hajautusarvon pituus.
- Hajautusfunktiot ovat muita kryptografisia funktioita nopeampia laskea. Niitä kannattaa suosia protokollissa.



## MAC

(*message authentication code MAC, message integrity check MIC, keyed hash function*)

- Salaisesta avaimesta riippuva hajautusfunktio. **Vain avaimen haltijat voivat laskea ja tarkistaa. Vain avaimen tunteva voi luoda uusia sanoma–MAC-pareja.** (yksisuuntaisuus avaimen osalta!)

  Käytetään sanoman alkuperän ja eheyden todentamiseen. (törmäyksettömyys!)

  Ulkopuolisen tekemät muutokset havaitaan. Ei todista mitään kolmansille osapuolille. (ei kiistämättömyyttä)
- Yleinen tapa muodostaa MAC on käyttää kryptografista hajautusfunktiota:

$$h_K(M) = h(K, M, K).$$

  - Salainen avain pitää lisätä sekä sanoman alkuun että loppuun.
  - Suorituskyvyn vuoksi voidaan avaimen jälkeen laittaa $512 - |K|$ bitin vakiotäyte $h(K, p, M, K)$, jotta avain voidaan prosessoida vain kerran ja etukäteen.
  - Riittää ottaa tuloksen 64 ensimmäistä bittiä.

  $h(K, M)$ ei kelpaa. Jos $h$ tehty kompressiofunktiosta, voi hyökkääjä jatkaa sanomaa.



- Teoreettisesti varmempi konstruktio on HMAC:

$$HMAC_K(M) = h(K, h(K, M)).$$

  - Suorituskyvyn vuoksi voidaan täydentää avainlohkot $h$:n lohkonpituuteen ja laskea etukäteen: $h(K, p, h(K, p, M))$.
- Periaatteessa voidaan käyttää kryptografisesti heikkoa tarkistussummaa, jos sanoma ja tarkistussumma salataan ja salaus estää sanoman muuttamisen ennustettavasti.

  **Käytännössä heikon tarkistussumman käyttö ei kannata.** Useimmiten se johtaa hyökkäyksiin.
- Kryptografisesti heikko tarkistussumma (esim. 32-bittinen CRC)
  - ei sovi lineaarisuutensa takia yhteen bittivirtasalauksen eikä monien lohkosalauksen moodien kanssa
  - on yleensä liian lyhyt.

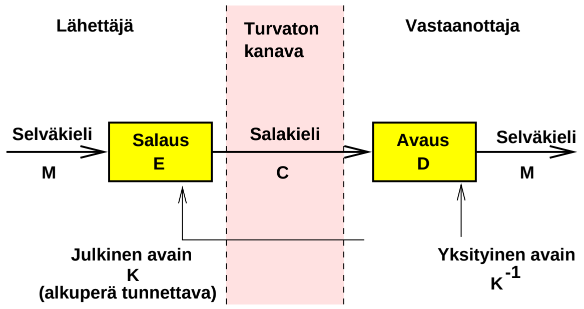
## Julkisen avaimen salaus

(*public key encryption, asymmetric cryptosystem*)

- Salakirjoitus **julkisella avaimella** ja avaus **yksityisellä avaimella**.

$$C = E_K(M)$$

$$M = D_{K^{-1}}(C)$$




- Yksityisen avaimen laskeminen julkisesta ja sanoman avaaminen ilman yksityistä avainta on mahdotonta.
- Julkisen avaimen aitous pitää varmistaa!



- Suurten kokonaislujen aritmetiikkaa, laskutoimitukset eivät vakioaikaisia.
- Lohkon pituus yleensä 200–2000 bittiä.
- Hitaita ja vaikeasti raudalla toteutettavia.
- Käytetään lohkosalausalgoritmin avainten välittämiseen.

  **Hybridisalaus**: Generoidaan sanomakohtainen avain ja salataan sanoma lohkosalauksella. Liitetään mukaan julkisen avaimen menetelmällä salattu sanomakohtainen avain.
- Hyötynä on helpompi avaintenhallinta:
  $k$ **julkista** avainta $k^2/2$ kahdenkeskisen avaimen sijaan.

  Yksityisiä avaimia ei tarvitse koskaan siirtää. Julkiset avaimet voidaan jakaa salaamatta.

  **Julkisen avaimen autentikointi on tärkeää!** Oletko varma, kenen avaimella olet salaamassa?



- Aina tunnetun tai valitun selväkielen hyökkäys.
- Turvallisuus riippuu tunnetuista vaikeaksi oletetuista ongelmista:
  - selkäreppuongelma
  - kokonaisluvun tekijöihin jako
  - diskreetti neliöjuuri
  - diskreetti logaritmi
  - vastaavat ongelmat elliptisille käyrille.
- Muita julkisen avaimen algoritmeja salauksen lisäksi:
  - allekirjoitus
  - avaintenvaihto
  - tunnistaminen
  - ym. ym.



## Modulolaskentoa

- Alkuluku
- Suurin yhteinen tekijä $\gcd$, Eukleideen algoritmi
- Moduloaritmetiikka, $\mathbb{Z}_n = < \mathbb{Z}_n, +, \cdot >$
- Kongruenssi $a \equiv b \pmod{n}$, operaattori $\bmod$
- $a|b$ ($a$ on $b$:n tekijä.)

## Käänteisluvut $\mathbb{Z}_n$:ssä

- Etsitään käänteislukuja modulo $n$.

$$x * a \equiv 1 \pmod{n}$$

joss

$$x * a + k * n = 1 \text{ jollekin } k \in \mathbb{Z}.$$

Tällainen $k$ on olemassa, joss $a$:lla ja $n$:llä ei ole yhteisiä tekijöitä eli $\gcd(a, n) = 1$.

Käänteisluku voidaan löytää Eukleideen algoritmilla. Esim. $4^{-1} \equiv 2 \pmod{7}$.

Jos $n$ on alkuluku, on kaikilla luvuilla $1, \ldots, n-1$ käänteisluku modulo $n$. ($< \mathbb{Z}_n, +, * >$ on **kunta**.)



## Käänteisluvut $\mathbb{Z}_n$:ssä

- Käänteisluvut lasketaan Eukleideen algoritmilla.

$$49^{-1} \equiv ? \pmod{4056}$$

$$\begin{aligned} 4056 &= 82 \cdot 49 + 38 \\ 49 &= 1 \cdot 38 + 11 \\ 38 &= 3 \cdot 11 + 5 \\ 11 &= 2 \cdot 5 + 1 \end{aligned}$$

$$\begin{aligned} 1 &= 11 - 2 \cdot 5 \\ &= 11 - 2 \cdot 38 + 6 \cdot 11 = -2 \cdot 38 + 7 \cdot 11 \\ &= -2 \cdot 38 + 7 \cdot 49 - 7 \cdot 38 = 7 \cdot 49 - 9 \cdot 38 \\ &= 7 \cdot 49 - 9 \cdot 4056 + 738 \cdot 49 = 745 \cdot 49 - 9 \cdot 4056 \end{aligned}$$

$$49^{-1} \equiv 745 \pmod{4056}$$

- Suoritusaika neliöllinen modulin pituuteen nähden.



## Potenssiinkorotus $\mathbb{Z}_n$:ssä

- Korotetaan toistuvasti neliöön ja kerrotaan eksponentin bittien osoittamat neliöt.

$$1234^{49} = 1234^{32} \cdot 1234^{16} \cdot 1234^{1}.$$

$$\begin{array}{lll} 1234^1 \bmod 4187 & & = 1234 \\ 1234^2 \bmod 4187 & = 1234^2 \bmod 4187 & = 2875 \\ 1234^4 \bmod 4187 & = 2875^2 \bmod 4187 & = 487 \\ 1234^8 \bmod 4187 & = 487^2 \bmod 4187 & = 2697 \\ 1234^{16} \bmod 4187 & = 2697^2 \bmod 4187 & = 990 \\ 1234^{32} \bmod 4187 & = 990^2 \bmod 4187 & = 342 \end{array}$$

$$\begin{aligned} 1234^{49} &= 342 \cdot 990 \cdot 1234 \bmod 4187 \\ &= 3620 \cdot 1234 \bmod 4187 \\ &= 3738 \end{aligned}$$

- Suoritusaika kuutiollinen modulin pituuteen nähden.

  Jos satunnaisesta valitussa eksponentissa on $k$ bittiä, tarvitaan neliöön korotuksia $k$ kappaletta ja muita kertolaskuja noin $k/2$ kappaletta.



## Lukuteoriaa RSA:ta varten

- **Määritelmä**: **Redusoitu jäännösjoukko** modulo $n$, $R_n$, koostuu niistä kokonaisluvuista $1 \leq a < n$, joille pätee $\gcd(n, a) = 1$.

  **Esimerkki**: Redusoitu jäännösjoukko modulo 10 on

$$R_{10} = \{1, 3, 7, 9\}.$$

- **Määritelmä (Eulerin totienttifunktio)**: $\phi(n)$ on resusoidun jäännösjoukon (mod $n$) kardinaliteetti.

  **Esimerkki**: $\phi(10) = 4$.

- **Lause**: Jos $n = \prod_{i=1}^{t} p_i^{e_i}$ ($p_i$:t alkulukuja), niin

$$\phi(n) = \prod_{i=1}^{t} p_i^{e_i - 1}(p_i - 1).$$

  **Seuraus**: Jos $p$ ja $q$ ($p \neq q$) ovat alkukuja, niin

$$\phi(p\, q) = (p-1)(q-1).$$

  **Esimerkki**: $\phi(10) = (2-1)(5-1) = 4.$

- **Lemma**: Jos $R_n$ on redusoitu jäännösjoukko modulo $n$ ja $gcd(a,n)=1$, niin

  $$\{a\,r \bmod n \mid r \in R_n\} = R_n.$$

  **Todistus**: Osoitetaan, että $a$:lla kertominen on $R_n$:n permutaatio. Kun $\gcd(a,n)=\gcd(r_i,n)=1$, niin myös $\gcd(a\,r_i,n)=1$. Niinpä $a\,r_i \in R_n$. Oletetaan sitten, että $a\,r_i \bmod n = a\,r_j \bmod n$. Kun $\gcd(a,n)=1$, on olemassa $a^{-1}$ modulo $n$. Kertomalla yhtälön molemmat puolet $a^{-1}$:llä saadaan $r_i = r_j$. Niinpä jokainen $r_i$ kuvautuu $a$:lla kerrottaessa eri alkiolle. Injektiivinen kuvaus äärelliseltä joukolta itselleen on bijektio ja siten permutaatio.

  **Esimerkki**: $a = 3$,

  $$\{3\cdot 1 \bmod 10,\ 3\cdot 3 \bmod 10,\ 3\cdot 7 \bmod 10,\ 3\cdot 9 \bmod 10\}$$
  $$= \{3, 9, 1, 7\} = R_{10}.$$

- **Seuraus**: $R_n$ on kertolaskuryhmä.

- **Lause (Euler)**: Jos $a$ on positiivinen kokonaisluku ja $\gcd(n,a)=1$, niin

  $$a^{\phi(n)} \equiv 1 \pmod{n}.$$

  **Todistus**: Olkoon $R = \{r_1, r_2, \ldots, r_{\phi(n)}\}$ redusoitu jäännösjoukko (mod $n$). Silloin $\{a\,r_1, a\,r_2, \ldots, a\,r_{\phi(n)}\} = R$.

  $$\prod_{i=1}^{\phi(n)} r_i \equiv \prod_{i=1}^{\phi(n)} a\,r_i \equiv a^{\phi(n)} \prod_{i=1}^{\phi(n)} r_i$$
  $$\Rightarrow\ a^{\phi(n)} \equiv 1 \pmod{n}.$$