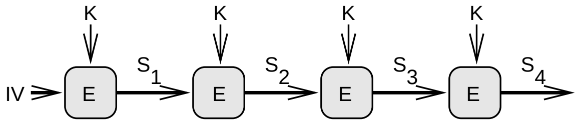
Pseudosatunnaislukugeneraattorit

(*cryptographically secure pseudo-random number generator*)

- Pseudosatunnaista bittivirtaa voidaan tuottaa lohkosalausalgoritmilla OFB-moodissa.



Halutaan nopeampi generaattori.

- Tavalliset pseudosatunnaislukugeneraattorit kuten **lineaariset ja polynomiset kongruenssigeneraattorit** (esim. UNIX:in rand) eivät ole turvallisia kryptografisessa käytössä.

Bittivirtasalaus

(*stream cipher*)

- Generaattorin tuottamaa bittivirtaa voidaan käyttää salaukseen *one-time-pad*:in tapaan:

  Alustetaan generaattori avaimella ja tuotetaan ns. **avainvirtaa**. XORataan avainvirran pseudosatunnaiset bitit salattavan sanoman kanssa.

$$c_i = p_i \oplus rand_i$$

  Vastaanottaja avaa sanoman XORaamalla sen saman avainvirran kanssa.

$$p_i = c_i \oplus rand_i$$

- Lähettäjän ja vastaanottajan generaattoreiden pitää olla alustettu samalla avaimella ja niiden täytyy pysyä synkronoituina.
- Virhe salakielisessä tekstissä muuttaa vain samaa bittiä selväkielessä.

  Salakielen bittien kääntäminen aiheuttaa saman muutoksen selväkielessä. (Salaus ei suojaa eheyttä.)
- Käyttö: laajakaistainen tiedonsiirto, häiriöaltis radioliikenne, edulliset rautatoteutukset

Siirtorekisterit

- Lasketaan uusi bitti n edellisen bitin funktiona,

$$b_i = f(b_{i-1}, \dots, b_{i-n}).$$

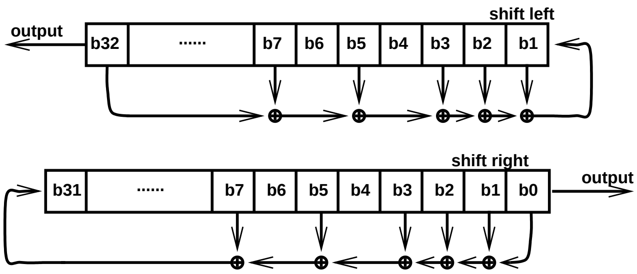
- Toteutetaan n bitin siirtorekisterinä. Uusi oikeanpuoleisin bitti lasketaan rekisterin vanhan sisällön funktiona.
- Perinteiset sotilassalaukset tehty siirtorekistereitä **yhdistelemällä**.

  Tehokkaita, muttei turvallisia standardiratkaisuja, joita kuka tahansa voi huoletta ottaa käyttööön.

Lineaarinen siirtorekisteri (LFSR)

- Uusi bitti lasketaan XOR:lla joistain n:stä edellisestä bitistä.
- Valittujen bittien numerot voidaan esittää karakteristisena polynomina. Esim. uusi bitti on XOR rekisterin biteistä 32, 7, 5, 3, 2 ja 1. Vastaava polynomi:

$$(32, 7, 5, 3, 2, 1, 0) = x^{32} + x^7 + x^5 + x^3 + x^2 + x^1 + 1.$$



Sama bittivirta toiseen suuntaan generoidaan alemmalla rekisterillä. (Schneier sekoittaa pahasti nämä kaksi esitystapaa.)

- Siirtorekisterin (kuten kaikkien äärellisten automaattien) tuottama bittijono alkaa lopulta toistua. Syklin suurin mahdollinen pituus on $2^n - 1$ bittiä, missä n on siirtorekisterin pituus.

  (n bitin siirtorekisterillä voi olla 2^n tilaa, joista $0, 0, \ldots, 0, 0$ johtaa triviaaliin sykliin.)

- Bittijonon pituus on maksimaalinen $2^n - 1$, joss karakteristinen polynomi on **primitiivinen**. Primitiivisiä polynomeja on taulukoitu kirjallisuudessa.

- Tällainen **Fibonacci-LFSR** on tehokas rautatoteutuksena. **Galois-LFSR** on ohjelmallisesti toteutettuna tehokkaampi siirtorekisteri.

- LFSR:n tuottama bittivirta **ei sellaisenaan kelpaa salaukseen**. Ratkaisuun riittää $2n$ bitin näyte.

LFSR:ien yhdistemät

- Yhdistetään useampia LFSR:iä epälineaarisesti.

- **Stop-and-Go**-generaattoreissa siirtorekistereitä kellotetaan pseudosatunnaisesti (samojen tai eri) siirtorekistereiden määrääminä aikoina.

  Kutistusgeneraattorissa siirtorekistereiden bittejä hylätään tai hyväksytään (samojen tai eri) siirtorekistereiden määräminä aikoina.

- **Korrelaatiohyökkäys**: pyritään päättelemään bittivirrasta yhden rekisterin toiminta kerrallaan.

- **Tiheät** primitiiviset polynomit tuottavat parempia tuloksia kuin harvat, **pitkät** parempia kuin lyhyet.

GSM-salausalgoritmi A5/1

- Algoritmi on salainen, päälinjat tiedossa:
  - Kolme lineaarista siirtorekisteriä, pituudet 19,22 ja 23.
  - Harvat karakteristiset polynomit.
  - *Stop-and-Go*: Katsotaan rekistereiden keskimmäisiä bittejä. Jos kahden tai kolmen rekisterin keskimmäinen bitti on 1, kellotetaan niitä. Jos kahden tai kolmen keskimmäinen bitti on 0, kellotetaan niitä.
  - Rekisterit alustetaan avaimella (19+22+23=64 bittiä) ja kehyksen numerolla.

- Väitetty *brute force*-hyökkäys (2^{41}) ei toimi!

- Hyökkäyksiä tämän tyyppisille salauksille tunnetaan.

  Periaate ok, mutta parametrit valittu tahallaan heikoiksi (lyhyet rekisterit, harvat karakteristiset polynomit).

- Vielä heikennetty version A5/2 käytössä itämaissa.

Muita salausalgoritmeja

- Lohkosalausalgoritmeja: RC5, Skipjack, Blowfish

- Bittivirtasalauksia: RC4; Blum, Blum ja Shub

  Salausnopeuksia 75 MHz Pentiumilla (yhden kokeilun mukaan):

Algoritmi	tavua/s
DES	397187
IDEA	218453
RC5 (r=12)	521679
Blowfish	489988
RC4	1476867

Todelliset satunnaisluvut

- Todellisia satunnaislukuja tarvitaan lähinnä avainten, nonssien ja alustusvektorien generointiin.
- Fyysisiä satunnaisuuden lähteitä:
  - radioaktiivinen hajoaminen
  - lämpöliikkeen aiheuttama kohina
  - epästabiili värähtelypiiri
  - elävä ääni ja kuva
  - käyttäjän ja fyysisten laitteiden ajoitus

  Jossain määrin satunnaisia arvoja tietokoneessa:
  - kello
  - käyttäjän syötteet hiirellä ja näppäimistöllä (aika ja sisältö)
  - I/O- ja tietoliikennetapahtumat (aika ja sisältö)
- Mikään lähde ei yksin ole täysin satunnainen eikä luotettava.

  Ratkaisu:
  Kerätään jossain määrin satunnaista dataa useammasta lähteestä ja laitetaan se yksisuuntaisen funktio läpi.

Jaetut salaisuudet

- On salaisuuksia, joita ei voi luottaa yhden ihmisen haltuun:
  - avainten varmuuskopiot
  - ydinaseiden laukaisukoodit
  - pankkiholvin avainkoodi.

  Auttaisiko salakirjoitus? Ehkä, mutta on parempia konsteja.
- Jaetaan salaisuus n osaan, jotka yhdessä paljastavat salaisuuden.
- Salaisuus s, satunnaisluvut $k_1, k_2, \ldots, k_{n-1}$,

  $$k_n = s \oplus k_1 \oplus k_2 \oplus \ldots \oplus k_{n-1}.$$

  Unohdetaan s.

  Salaisuus paljastuu, jos kaikki osat tiedetään. Muuten mitään tietoa s:stä ei saada.

  $$s = k_1 \oplus k_2 \oplus \ldots \oplus k_n.$$
- Käytetään esim. avaimen varmuuskopiointiin. Jaetaan osat eri henkilöille, jotka vain yhdessä voivat laskea avaimen.
- Ongelma: mitä jos joku kadottaa osansa tai kieltäytyy kertomasta sitä?

(m, n)-**kynnysrakenne**

- Jaetaan salaisuus n osaan (**varjoon**), joista mitkä tahansa m riittää paljastamaan salaisuuden.
- **LaGrangen interpolaatiopolynomi** (Shamir 79)

  m pistettä (x_1, y_1), $(x_2, y_2), \ldots, (x_m, y_m)$ määräävät yksikäsitteisen polynomin astetta $m-1$.

  Jos pisteitä on enemmän, ne ovat resundantteja.
  $m-1$ pistettä ei riitä polynomin laskemiseen.
- Halutaan jakaa salaisuus s m:ään osaan. Generoidaan satunnainen polynomi asetetta $m-1$, jonka vakiotermi on s.

  $$F(x) = c_{m-1}x^{m-1} + c_{m-2}x^{m-2} + \cdots + c_1 x + s.$$

  Varjot ovat pisteitä käyrältä:

  $$k_1 = F(1),\ k_2 = F(2), \ldots, k_n = F(n).$$

  Mitkä tahansa m pistettä riittävät polynomin laskemiseen. Salaisuus on

  $$s = F(0).$$

- Käytännössä käytetään modulo-aritmetiikkaa:

  $$F(x) = (c_{m-1}x^{m-1} + c_{m-2}x^{m-2} + \cdots + c_1 x + s) \bmod p,$$

  missä p on alkuluku, joka on suurempi kuin s ja n.
  Kertoimet c_i valitaan satunnaisesti väliltä $0, \ldots, p-1$.

  (p on alkuluku, joten $\mathbb{Z}_n$ on kunta. Interpolaatio toimii.)
- Käytetään mieluummin aina samaa p, esim. $p = 2^{16} + 1$. Jaetaan salaisuus tasakokoisiin paloihin (esim. 16 bittiä), jotka käsitellään erikseen.
- Shamir-kynnysrakenteen ominaisuuksia:
  - **laajennettava**: varjoja voidaan luoda ja jakaa lisää
  - **luottamus parametrisoitavissa**: varjoja voi jakaa eri tahoille eri määriä
  - **täydellinen**: vähemmän kuin m varjoa ei paljasta mitään sanomasta, turvallisuus perustuu matemaattisiin tosiasioihin.

Kynnysrakenne-esimerkki

Luonti: salaisuus $s = 1234$, $p = 2^{16} + 1$, $n = 5$, $m = 3$.

$$F(x) = (c_2 x^2 + c_1 x + 1234) \bmod (2^{16} + 1).$$

Valitaan satunnaiset kertoimet.

$$F(x) = (37425x^2 + 50374x + 1234) \bmod (2^{16} + 1).$$

Varjot:

$$\begin{array}{ll} F(1) = 23496 & F(2) = 55071 \\ F(3) = 30422 & F(4) = 15086 \\ F(5) = 9063 & \end{array}$$

Ratkaisu:
tunnetaan 3 varjoa: $(1, 23496), (2, 55071), (5, 9063)$.

$$\begin{aligned} F(1) &= \quad (c_2 + \quad c_1 + s) \bmod 65537 = 23496 \\ F(2) &= (4{*}c_2 + 2{*}c_1 + s) \bmod 65537 = 55071 \\ F(5) &= (25{*}c_2 + 5{*}c_1 + s) \bmod 65537 = 9063 \end{aligned}$$

Matriisimuodossa $\pmod{65537}$

$$\begin{bmatrix} 1 & 1 & 1 \\ 4 & 2 & 1 \\ 25 & 5 & 1 \end{bmatrix} \begin{bmatrix} c_2 \\ c_1 \\ s \end{bmatrix} = \begin{bmatrix} 23496 \\ 55071 \\ 9063 \end{bmatrix}$$

Ratkaisu: $c_2 = 37425$, $c_1 = 50374$, $s = 1234$.

Kiinalainen jäännöslause

- **Lause**: Olkoon $n_1, \ldots, n_k$ positiivisia kokonaislukuja, joille pareittain $\gcd(n_i, n_j) = 1$, jos $i \neq j$. Yhtälöryhmällä

$$\boxed{\begin{array}{ll} x \equiv a_1 & \pmod{n_1} \\ x \equiv a_2 & \pmod{n_2} \\ \multicolumn{2}{c}{\cdots} \\ x \equiv a_k & \pmod{n_k} \end{array}}$$

on yksikäsitteinen ratkaisu modulo $N = n_1 n_2 \cdots n_k$.

- Ratkaisun laskeminen:

$$N_i = N/n_i, \quad i = 1, \ldots, k.$$

Kongruensseilla $N_i x_i \equiv 1 \pmod{n_i}$ on yksikäsitteiset ratkaisut, sillä $\gcd(n_i, N_i) = 1$. Ratkaisu löydetään Eukleideen algoritmilla.

$$\boxed{x = a_1 N_1 x_1 + a_2 N_2 x_2 + \ldots + a_k N_k x_k.}$$

Tämä on ratkaisu, sillä kaikille i

$$x \equiv a_i N_i x_i \equiv a_1 \cdot 1 \equiv a_1 \pmod{n_i}.$$

- Ratkaisu on yksikäsitteinen modulo N: Oletetaan

$$x \equiv a_i \equiv x' \pmod{n_i}, \quad i = 1, \ldots, k.$$

Tällöin $n_i \mid (x - x')$, $i = 1, \ldots, k$. Koska n_i:llä ei ole yhteisiä tekijöitä, $n_1 n_2 \cdots n_k \mid (x - x')$. Niinpä

$$x \equiv x' \pmod{N}.$$

- **Esimerkki**:

$$\begin{cases} x \equiv 2 & \pmod{3} \\ x \equiv 3 & \pmod{5} \\ x \equiv 2 & \pmod{7} \end{cases}$$

$$N = 3 \cdot 5 \cdot 7 = 105.$$

$$N_1 = N/3 = 35,\ N_2 = N/5 = 21,\ N_3 = N/7 = 15.$$

Ratkaistaan lineaariset kongruenssit

$$35x_1 \equiv 1(\bmod 3),\ 21x_2 \equiv 1(\bmod 5),\ 15x_3 \equiv 1(\bmod 7).$$

$$x_1 = 2,\ x_2 = 1,\ x_3 = 1.$$

Ratkaisu

$$x = 2 \cdot 35 \cdot 2 + 3 \cdot 21 \cdot 1 + 2 \cdot 15 \cdot 1 = 233.$$

Redusoidaan modulo 105,

$$x = 233 \equiv 23 \pmod{105}.$$

Toinen kynnysrakenne

- Perustuu kiinalaisen jäännöslauseeseen.
- Halutaan muodostaa (m, n)-kynnysrakenne salaisuudelle s. Valitaan alkuluku $p > s$ ja luvut $d_1 < d_2 < \ldots < d_n$, joilla ei ole yhteisiä alkutekijöitä. Lisäksi pitää päteä

$$\underbrace{d_1 \cdot d_2 \cdots d_m}_{m \text{ pienintä}} > p \cdot \underbrace{d_{n-m+2} \cdot d_{n-m+3} \cdot d_n}_{m-1 \text{ suurinta}}.$$

Sekoitetaan alkuperäinen viesti laskemalla

$$s' = s + r\,p,$$

missä r on suuri satunnaisluku. Lasketaan varjot

$$k_i = s' \bmod d_i.$$

s saadaan selville mistä tahansa m:stä varjosta kiinalaisella jäännöslauseella ja siitä

$$s = s' \bmod p.$$

Kynnysrakenne-esimerkki

- $n = 5$, $m = 3$. Valitaan $p = 2887$ ja sopivat d_i:t,

$$d_1 = 20011,\ d_2 = 20021,\ d_3 = 20023,$$
$$d_4 = 20029,\ d_5 = 20057.$$

- Jaetaan salaisuus $s = 1234$. Valitaan satunnaisesti $r = 14334$. Tällöin

$$s' = 1234 + 14334 \cdot 2887 = 41383492.$$

Lasketaan varjot

$$k_1 = 744,\ k_2 = 85,\ k_3 = 15974,$$
$$k_4 = 3578,\ k_5 = 5901.$$

- Tunnetaan 3 varjoa: (1,744), (2,85), (5,5901). Saadaan kongruenssiyhtälöt

$$\begin{cases} s' \bmod 20011 = 744 \\ s' \bmod 20021 = 85 \\ s' \bmod 20057 = 5901 \end{cases}$$

Ratkaisu on $s' = 41383492$. Tästä saadaan

$$s = s' \bmod p = 41383492 \bmod 2887 = 1234.$$

- Kynnysrakenteille on olemassa monia muita toteutustapoja.
- Eri henkilöille voidaan antaa erilaiset painot jakamalla eri määrä varjoja.

  Varjojen määrää n voidaan lisätä.
- Voidaan jakaa henkilöt ryhmiin ja vaatia tietty määrä osallistujia kustakin ryhmästä salaisuuden selvittämiseen.
- Ongelma: vain varjojen jakaja tietää, ovatko ne oikeita. Hän voi antaa joillekin vääriä varjoja tai varjot voivat korruptoitua matkalla.

  On olemassa protokollia, joilla voidaan verifioida jakajan reiluus salaisuutta paljastamatta.
- Tärkein sovellutus avaimen varmuuskopiointi
  1. Jaetaan avain varjoihin.
  2. Salataan varjot luotettujen henkilöiden julkisilla avaimilla.
  3. Talletetaan salatut varjot samalle levylle salattujen tiedostojen kanssa.