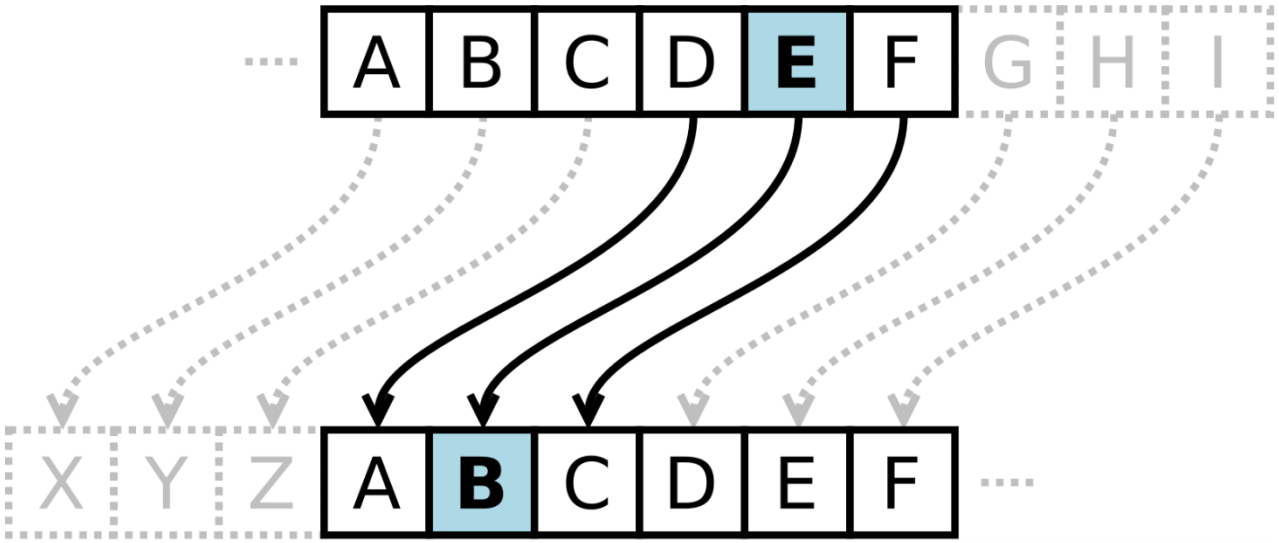
Kriptografija

Cryptography has been a major force in human history for more than 3,000 years

The Codebreakers by David Kahn

Zgodovina kriptografije

- "**Cezarjev kodirnik**" v času Rimskega imperija:
  - Za vsako črko se premakni za n mest naprej po abecedi in za Z ponovno priči pri A.
  - enostavno za dekodiranje: analiza frekvence črk v kriptiranem besedilu (samoglasniki so pogostejši).



https://en.wikipedia.org/wiki/File:Caesar_cipher_left_shift_of_3.svg

Zgodovina kriptografije

- Leta 1920 je bil zasnovan prvi teoretično dokazano varen informacijski kodirnik: **one-time pad**:
  - nekodirano besedilo *P* zapišemo z bitnim zaporedjem in ga preko operacije XOR prištejemo k bitnemu zaporedju kodirnega ključa *K*, ki je enake dolžine kot osnovno besedilo. Dobimo kodirano besedilo *C*.
  - Prejemnik (ki pozna *K*) lahko kodirano besedilo dekodira z novo operacijo XOR:

    $C \oplus K = P \oplus K \oplus K = P$ kjer je $\oplus$ bitna operacija XOR
  - Težava kodirnika "one-time pad" je v tem, da **morata bitno zaporedje ključa poznati tako oddajnik kot prejemnik in da mora biti to zaporedje dolžine sporočila, ki ga kodiramo**.
  - Če je isto zaporedje ključa uporabljeno pri dveh ali več sporočilih, potem kodirnik ni več varen, saj velja

$$C_1 \oplus C_2 = P_1 \oplus K \oplus P_2 \oplus K = P_1 \oplus P_2$$

Stopnje varovanja podatkov

- Varnost komunikacij in varovanje podatkov sta v širšem pomenu opredeljena z načeli lastništva (*possession*), zaupnosti (*confidentiality*), celovitosti (*integrity*), dostopnost (*availability*), pristnost (*authenticity*), nezatajljivosti (*non-repudiation*) in uporabnosti (*utility*).
- **Zaupnost**

  Zaupnost izraža preprečevanje razkrivanja informacij nepooblaščeni osebi ali sistemu. Navadno se zagotavlja s šifriranjem podatkov, z omejevanjem mesta dostopnosti podatkov (v podatkovnih bazah, dnevnikih sistemov, varnostnih kopijah, obvestilih uporabnikom itd.) ter z omejevanjem dostopa do mesta, kjer so podatki shranjeni.

  Zaupnost je potreben (a ne zadosten) pogoj za zagotavljanje zasebnosti oseb in za varovanje njihovih podatkov.
- **Celovitost**

  Celovitost podatkov pomeni, da podatkov ni mogoče spreminjati brez dovoljenja. Definicija celovitosti na področju informacijske varnosti se torej močno razlikuje od definicije podatkovne celovitosti v podatkovnih bazah. Celovitost je na primer kršena, če oseba ali program po naključju, napaki ali zlonamerno izbriše ali spremeni pomembne podatke ali ko na kakršnikoli način zlonamerno vpliva na njihovo zbiranje (npr. ko ista oseba odda veliko število glasov na spletni anketi).

Stopnje varovanja podatkov

- **Dostopnost**

  Za kateri koli informacijski sistem morajo biti podatki na voljo, ko je to potrebno. To pomeni, da morajo pravilno delovati vse komponente sistema od računalniška sistema, ki ga uporabljajmo za shranjevanje in obdelavo informacij, do komponent za nadzor varnosti in komunikacijskih kanalov, ki se uporabljajo za dostop do podatkov. Sistemi z visoko dostopnostjo omogočajo neokrnjen dostop do podatkov, preprečujejo motnje v delovanju zaradi izpadov električne energije, okvar strojne opreme in nadgradnje sistema. Zagotavljanje dostopnosti vključuje tudi preprečevanje napadov za preprečevanje dostopa (denial-of-service attacks).

- **Pristnost**

  V računalništvu, e-poslovanju in informacijskem varovanju je potrebno zagotoviti, da so podatki, transakcije, sporočila ali dokumenti (elektronski ali fizični) pristni. Prav tako je za verodostojnost pristnosti pomembno, da so v komunikacijo vpletene stranke to, kar trdijo, da so.

- **Nezatajljivost**

  V pravu, nezatajljivost pomeni namero pogodbene stranke, da izpolni svoje s pogodbo določene obveznosti. Pomeni tudi, da stranka v transakciji ne more zanikati prejetja informacije, ki jo je poslala druga stranka. Elektronsko poslovanje zagotavlja verodostojnost in nezatajljivost s pomočjo tehnologij digitalnih podpisov in šifriranja.

Kriptoanaliza

- **Kriptografska definicija varnosti podatkov:**
  - V splošnem, nudi kriptografski sistem **zaščito stopnje λ**, če potrebuje uspešen napad na sistem približno **vložek v velikosti $2^{\lambda-1}$**.
  - Velikost zahtevanega vložka se v praksi ocenjuje s **produktom zahtevanega časa in cene zahtevane opreme**.
  - Po splošno sprejetem konsenzu nudi kriptografski sistem primerno varnost do leta X, če znaša strošek uspešnega napada na sistem v danem letu vsaj **40 M$-dni**. V zadnjem času se ta strošek meri v številu operacij in znaša nekje med **2^{80}** in **2^{100}** operacij.
  - Zaradi nenehnega napredka programske in strojne opreme, ki vpliva tako na učinkovitost kot na ceno uporabljene opreme, prihaja do **naravne degradacije stopnje varnosti** hranjenih podatkov.
  - Po Moorovem zakonu, se zaradi tehnološkega razvoja zahtevan vložek prepolovi vsakih 9 mesecev (vsakih 18 mesecev zaradi napredka v opremi in vsakih 18 mesecev zaradi napredka v algoritmih kriptoanalize). To imenujemo **dvojni Moorov zakon faktorizacije.**

Varnost in uporabnost kriptografije

- Leta 1940 je Claude Shannon dokazal:

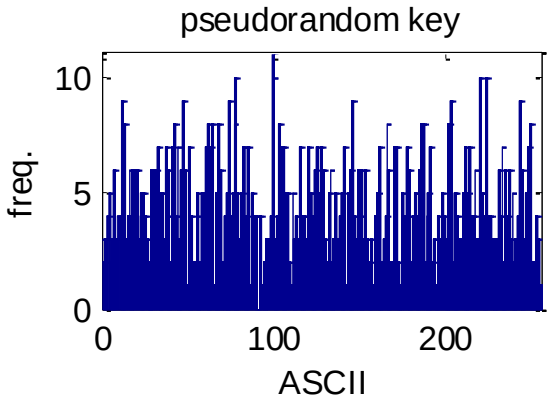
Kodirnik je teoretično varen, če in samo če tako oddajnik kot prejemnik uporabljata bitno zaporedje ključa, ki je dolžine komunikacijskega sporočila.

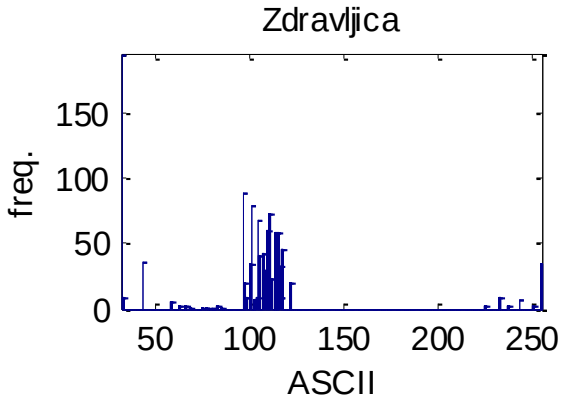
- Toda izmenjava tako velikih ključev je povsem nepraktična. Ali lahko uporabimo krajše ključe?
- Kaj če predpostavimo, da je prisluškovalec omejen s polinomskim izvajalnim časom (torej, da je časovno omejen)?

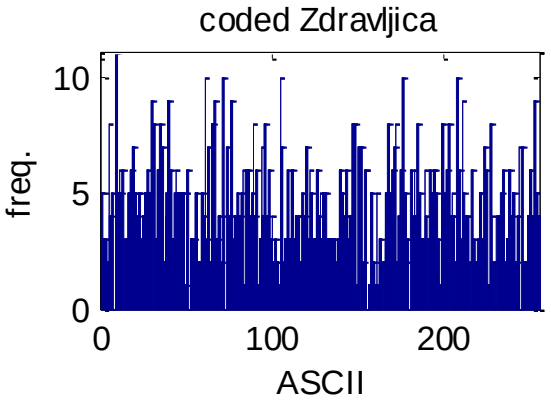
Psevdonaključni generatorji

- psevdonaključni generator je funkcija *f*, ki ima naslednje lastnosti:
  1. *f preslika n*-bitno vhodno vrednost (imenovano *seme*) v *p(n)*-bitno izhodno zaporedje, kjer je *p(n)* polinomsko večji od *n*.
  2. *f* je izračunljiva v polinomskem času po *n*.
  3. zlonamerna oseba ne more v polinomskem času razlikovati izhoda funkcije *f* od resnično naključnega zaporedja.

- Zgled: matlab & funkcija rand








```
k = uint8(rand(1,10000)*255);
```

```
p = 'Prijatlji! obrodile so…'
```

```
c = bitxor(uint8(p), …
                k(1:length(p)));
```

Kriptografske sekljalne funkcije
(Cryptographic hash functions)

- Večina kriptografski sekljalnih funkcij prejme na vhodu niz poljubne dolžine in vrne sekljano kodo fiksne dolžine.
- Splošne lastnosti dobrih sekljalnih funkcij lahko strnemo v naslednje postavke:
  1. enostaven izračun sekljane vrednosti kateregakoli sporočila;
  2. praktično nemogoče je najti sporočilo, ki daje izbrano sekljano vrednost;
  3. nemogoče je spremeniti sporočilo, ne da bi spremenili njegovo sekljano vrednost;
  4. praktično nemogoče je najti dve različni sporočili z isto sekljano vrednostjo;
  5. efekt plaza (*avalanche effect*): sprememba enega samega bita na vhodu povzroči veliko spremembo izračunane sekljane vrednosti.

- **Znana kriptografske sekljalne funkcije**:
  - MD4, MD5 (H=128, λ=64),
  - RIPEMD-160 (H=160, λ=80),
  - SHA-1 (H=160, λ=80),
  - SHA-256 (H=256, λ=128)

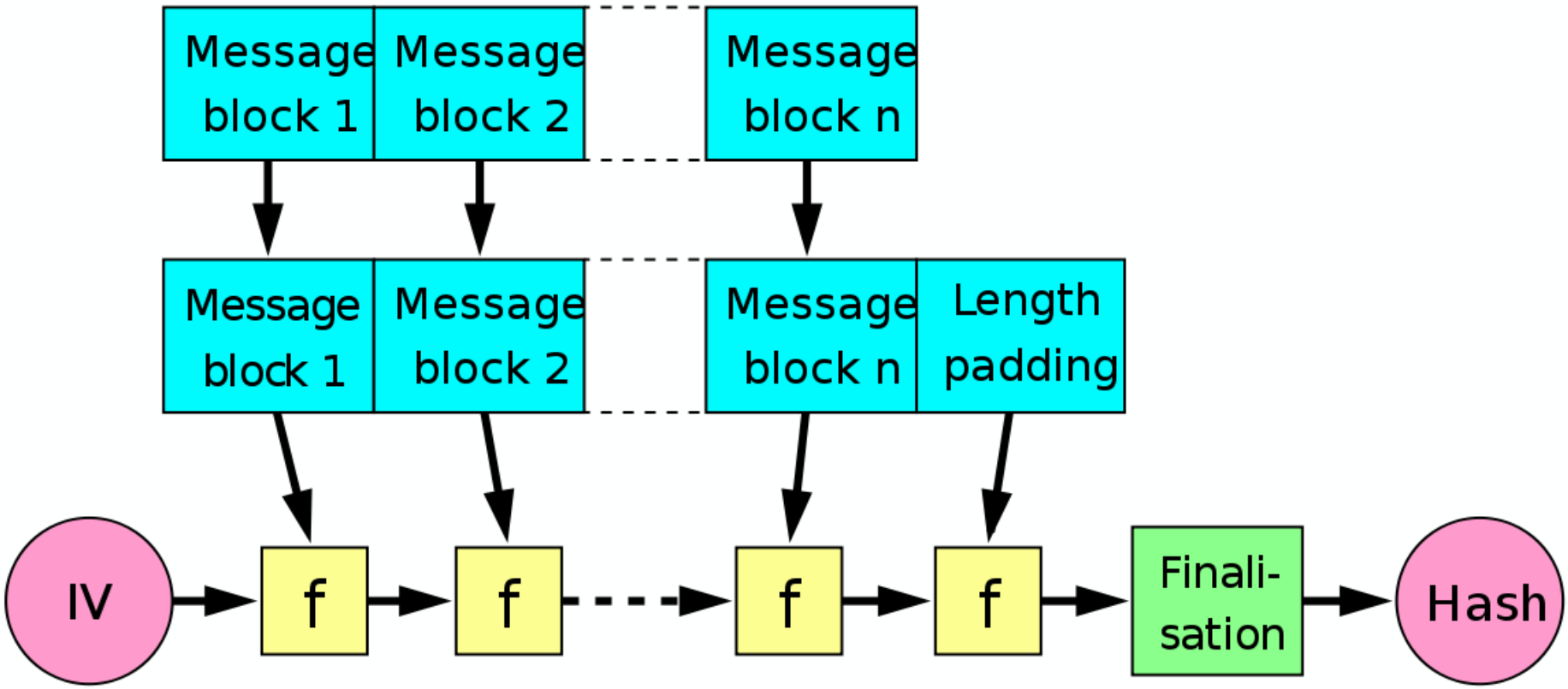
  H označuje **bitno dolžino** ključa sekljanja: Da bi dosegli **nivo varnosti λ** in da bi zadostili prvima dvema zgornjima zahtevama (1. & 2.), mora biti **H ≥ λ**.

Kriptografske sekljalne funkcije
(Cryptographic hash functions)

- **H** označuje **bitno dolžino** ključa sekljanja.
- **Paradoks rojstnega dne** (*birthday paradox*): Če izbiramo naključne elemente množice s kardinalnostjo *C* je povprečno število elementov, ki bodo izvlečeni preden bo isti element izvlečen dvakrat (t.j. preden nastopi kolizija) enako 1.25√C.
  - Če računamo sekljano vrednost različnih naključno izbranih vhodov in ima sekljalna funkcija *H* možnih izhodov, lahko podvojitev sekljalne vrednosti pričakujemo po $1{,}25 \cdot 2^{H/2}$ poskusih.
  - Za dosego stopnje varnosti λ in za zagotovitev odpornosti na zgoraj opisane kolizije mora veljati $\boldsymbol{H \geq 2\lambda}$, kjer je *H* bitna dolžina sekljalne vrednosti in λ stopnja varnosti.
- **Konkatinacija kriptografskih sekljalnih funkcij:** SSL uporablja konkatinacijo sekljalnih funkcij MD5 in SHA-1, da bi zagotovila varnost komunikacijskega protokola v primeru razvozlanja ene izmed sekljalnih funkcij. V splošnem sta MD5 in SHA-1 od leta 2009 dalje dve najbolj pogosto uporabljeni kriptografski sekljalni funkciji. Vendar pa je bila MD5 razvozlana in napad nanjo je bil leta 2008 uporabljen za zlom SSL [Sotirov 2009].

Merkle-Damgård-ova konstrukcija sekljalne vrednosti

- za sekljanje podatkov poljubne dolžine:



https://en.wikipedia.org/wiki/File:Merkle-Damgard_hash_big.svg

Kodirniki (ciphers)

- Moderni kodirniki se glede načina enkripcije delijo v dva velika razreda
  - **Simetrično kodiranje** uporablja isti privatni ključ tako za kodiranje kot za dekodiranje sporočil. Primera simetričnega kodirnika sta standarda DES in AES.
  - **Asimetrično kodiranje** temelji na privatnem in javnem ključu. Javni ključ je poznan vsem in služi za enkripcijo podatkov, ki jih lahko dekodiramo samo s skritim privatnim ključem. Analogija z realnim svetom daje javnemu kriptografskemu ključu podobo fizične ključavnice, s katero lahko vsakdo zaklene poljuben zaboj, privatnemu ključu pa podobo fizičnega ključa, ki to ključavnico odklepa. Primer asimetričnega kodirnika je standard RSA.

Simetrični kodirniki se dodatno delijo v **pretočne** (*stream*), ki operirajo nad posameznimi zlogi podatkov, in **bločne** (*block*), ki hkrati obdelajo celoten blok podatkov (trenutno je tipična velikost bloka 128 zlogov). Meja med bločnimi in pretočnimi kodirniki ni izrazita in številni bločni kodirniki se lahko uporabijo kot pretočni kodirniki.

Dolžina kriptografskega ključa

- Obstaja široko soglasje o dolžinah simetričnih ključev in kriptografskih sekljalnih funkcij, ki so "konzervativne", torej imajo dobro možnost, da ponujajo zelo dolgoročno varnost.
- Veliko manj konsenza je o konzervativnih dolžinah asimetričnih kriptosistemov, kot je RSA.
- Po trenutnem konsenzu nudijo vsi naslednji kodirniki podobne stopnjo zaščite:
  - simetrični kodirnik s 128 bitnim ključem
  - asimetrični kodirnik s 3072 bitnim ključem,
  - asimetrični kodirnik z uporabo eliptičnih krivulj s 512 bitov dolgim ključem.
- Dolžino ključa, ki izbrani metodi enkripcije zagotavlja ustrezno stopnjo varnosti do poljubnega leta si lahko izračunamo na spletni strani: http://www.keylength.com/

Asimetrični kodirniki

(Asymmetric ciphers)

- Asimetrično kodiranje temelji na privatnem in javnem ključu.
- Javni ključ je poznan vsem in služi za enkripcijo podatkov, ki jih lahko dekodiramo samo s skritim zasebnim ključem. Običajno lahko javni ključ uporabnika A uporablja katera koli stranka za šifriranje podatkov (seveda samo za podatke, ki so namenjeni uporabniku A).
- Uporabnik A lahko podatke dešifrira samo s pomočjo zasebnega ključa. Javni in zasebni ključ vedno nastopata v parih, tako kot ključavnica in njen ključ.

- Trenutno so v uporabi trije implementacijski koncepti kodiranja z javnim/zasebnim ključem:
  - faktorizacija celih števil (*integer factorization*)
  - diskretni logaritmi (*discrete logarithm*)
  - eliptične krivulje (*elliptic curve*)

Faktorizacija celih števil

- **Osnovni problem faktorizacije celih števil**: Za dano sestavljeno celo število $n > 0$, najdi celi števili $p > 1$ in $q > 1$ tako, da velja $n = pq$.
- Pri RSA, najpogostejšem faktorizacijskem algoritmu za asimetrične kriptosisteme, vsebuje uporabnikov javni ključ celo število n, njemu ustrezen zasebni ključ pa vsebuje praštevili p in q. Število n je za vsakega uporabnika unikatno.

- **Nekaj ključnih lastnosti faktorizacije praštevil**:
  - **Število praštevil do x je sorazmerno $x/log(x)$**: torej, če je število n zgrajeno kot produkt dveh, recimo, b-mestnih praštevil, je računska kompleksnost za faktorizacijo števila n z metodo zaporednih poskusov velikostnega reda 10^b. Že za zmerno velikost b=50 je računski napor te velikosti izven dosega sodobnih klasičnih računalnikov.
  - Obstajajo algoritmi faktorizacije, ki so veliko hitrejši od metode zaporednih poskusov in **zahtevana bitna dolžina modulov $p > 1$ in $q > 1$ raste veliko hitreje kot linearna funkcija želene stopnje varnosti**.

Faktorizacija celih števil

- Algoritem *Number Field Sieve* (NFS), trenutno najhitrejša znana metoda za faktorizacijo celih števil na klasičnih računalnikih, potrebuje v povprečju **O(*exp((b· 64/9)$^{1/3}$· log (b)$^{2/3}$)*)** operacij za faktorizacijo *b*-bitnega celega števila *n*, torej 2/3 poti med eksponentnim in polinomskim časom.
  - Maja 2005 je bila oznanjena uspešna faktorizaciji RSA-200 (663-bitnega številka z 200 decimalnimi številkami). Zahtevala je več mesecev računskega časa na platformi z 80 procesorji AMD Opteron.
  - Januarja 2010 je bila oznanjena faktorizacija RSA-768. V napadu v skupni dolžini več kot dveh let je sodelovalo več sto računalnikov.
- **Dvojni Moorov zakon faktorizacije:** strošek faktorizacije poljubnega celega števila pade za faktor 2 vsakih 9 mesecev (za 2 vsakih 18 mesecev zaradi napredka kriptoanalize, in za 2 vsakih 18 mesecev zaradi napredka strojne opreme).
- **Dolžine ključev RSA, ki nudijo ustrezno varnostno zaščito**
  - Leto 2010 (λ = 75): 1112 bitov
  - Leto 2020 (λ = 82): 1387 bitov
- Obstaja **učinkovit kvantni algoritem** za faktorizacijo celih števil, ki ga je izumil Peter Shor

Diskretni logaritem

- **Osnovni matematični koncept:** Za dani element *h* končne ciklične multiplikativne grupe $(\boldsymbol{Z}_n)^\times = \{0, 1, 2, \ldots, n-1\}$ z *n* elementi, ki jo po modulu *n* napenja **multiplikativni generator *g***, najdi celo število *k,* tako da velja

  $$h = g^k \bmod n.$$

  Najmanjši nenegativni *k* se imenuje **diskretni logaritem** (*discrete logarithm*) od *h* glede na osnovo *g* in je označen z $\boldsymbol{k = \log_g(h)}$.
- Katerikoli celi števili k_1 in k_2 za kateri velja $g^{k1} \bmod n = g^{k2} \bmod n$ sta, po definiciji, **kongruenčni po modulu *n*** (t.j. imata enak ostanek po deljenju z *n*) in pripadata istemu kongruenčnemu razredu (*congruence classes*) po modulu *n*.
- **Zgled**: poiščimo rešitev enačbe $3^k \equiv 13 \pmod{17}$. Ena izmed rešitev je $k=4$. Toda to ni edina rešitev. Ker je $3^{16} \bmod 17 = 1$, velja $3^{4+16n} \bmod 17 = 13 \times 1^n \equiv 13$. Torej ima enačba neskončno mnogo rešitev, njihova splošna oblika pa je $k = 4 + 16n$. Ker je $m=16$ najmanjše pozitivno celo število, ki zadosti enačbi $3^m \bmod 17 = 1$ (t.j. 16 je red števila 3 v grupi $(\mathbf{Z}_{17})^\times$), so rešitve oblike $k = 4 + 16n$ edine rešitve diskretnega logaritma v $(\mathbf{Z}_{17})^\times$. Rešitev navadno zapišemo v obliki $k \equiv 4 \pmod{16}$.

Izmenjava ključa

(Diffie–Hellman key exchange)

Diffie-Hellmanov (D-H) algoritem za izmenjavo ključev je eden od prvih praktičnih primerov izmenjave ključev na področju kriptografije. Metoda omogoča dvema stranema, ki nimata predhodnih stikov, da skupaj vzpostavita skupni privatni ključ skozi javno odprt (kriptografsko nevaren) komunikacijski kanal. Ta ključ se lahko nato uporabi za šifriranje sporočili z uporabo simetričnega kodirnika.

Algoritem:

- Alica in Bob skupaj izbereta praštevilo p in osnovo g (ti dve vrednosti sta lahko znani vsem).
- Alica izbere naključno naravno število a in pošlje Bobu $g^a \ mod \ p$.
- Bob izbere naključno naravno število b in pošlje Alici $g^b \ mod \ p$.
- Alica izračuna $k = (g^b)^a$.
- Bob izračuna $k = (g^a)^b$.

Samo a, b in $g^{ab} \ mod \ p = g^{ba}$ mod p morajo ostati skrivni. Vse ostale računske entitete p, g, $g^a \ mod \ p$, in $g^b \ mod \ p$ so javno dostopni. Ko Alica in Bob izračunata skupni skrivni ključ, ga lahko uporabita kot ključ za simetrični kodirnik.

Seveda so za zagotovitev varnosti potrebne velike vrednosti a, b in p, saj je, na primer, enostavno preizkusiti vse možne vrednosti g^{ab} mod 23 (obstaja največ 22 takih vrednosti, četudi sta a in b velika).

Diskretni logaritem vs. faktorizacija celih števil

- Čeprav sta računanje diskretnih logaritmov in faktorizacije celih števil matematično različna problema, si delita številne skupne lastnosti:
  - Oba problema sta računsko težka (v smislu klasičnega računanja, torej mimo kvantnih računalnikov).
  - Za oba obstaja učinkovit kvantni algoritem.
  - Algoritmi za enega izmed problemov so pogosto prilagojeni še za drug problem.
  - Oba problema imata računsko zelo učinkovita inverzna problema (t.j. množenje celih števil oz. diskretno potenciranje) kar je ugodno za namene kriptografije.

Eliptične krivulje

- Zaradi nenehnega napredka strojne opreme in kriptografskega znanja se velikost kriptirnih ključev algoritma RSA nenehno povečuje, kar zmanjšuje njegovo uporabnost na platformah z omejeno zmogljivostjo.
- Alternativo predstavlja kriptografija, ki temelji na diskretnem logaritmu eliptičnih krivulj (*Elliptic Curve Discrete Logarithm*). Eliptična krivulja je v splošnem definirana z enačbo

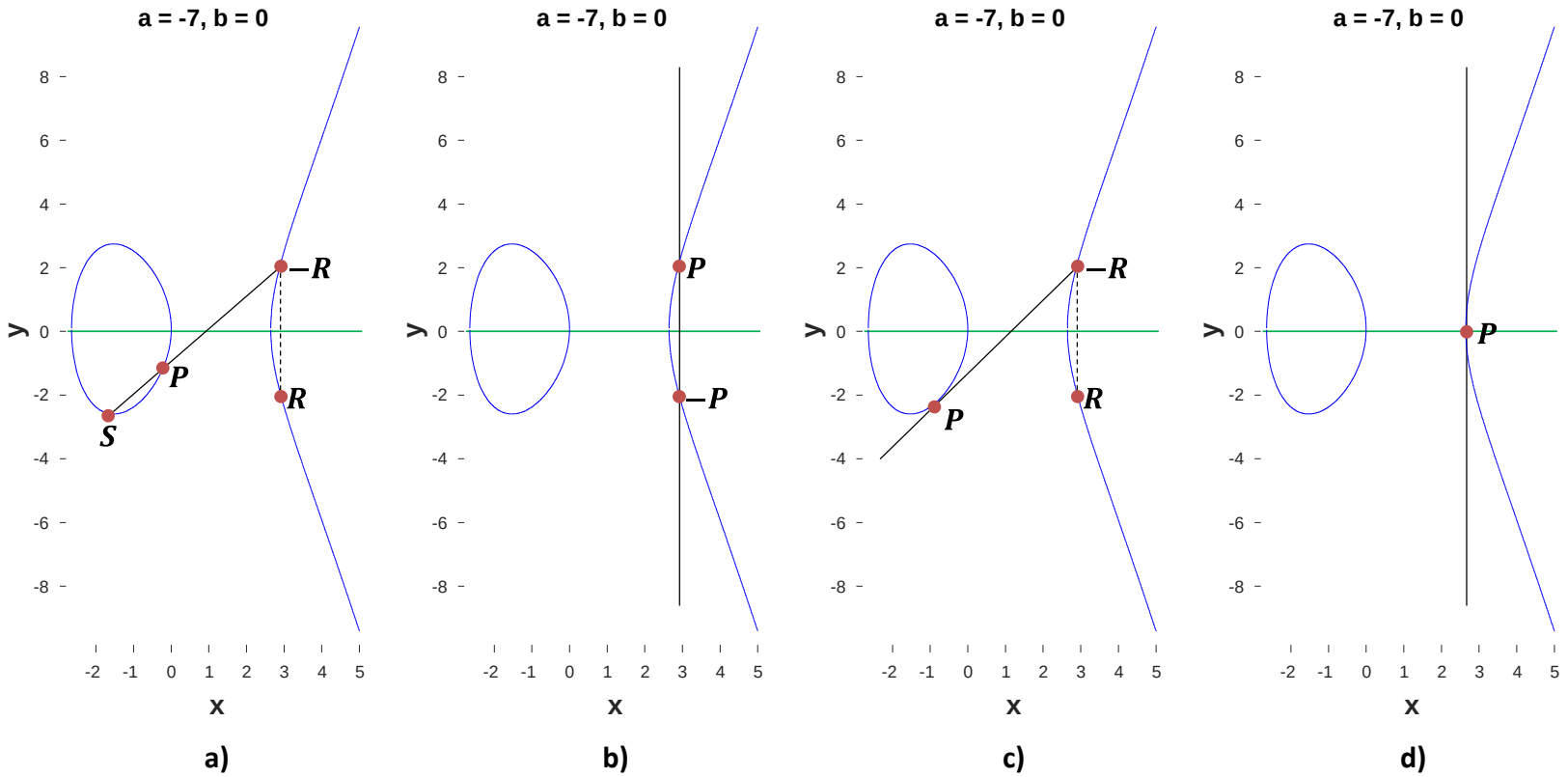
$$y^2 = x^3 + ax + b$$

kjer, zaradi varnostnih razlogov, $4a^3 + 27b^2 \neq 0$ (v nasprotnem primeru obstajajo na krivulji singularnosti, npr. presečišča krivulje same s sabo ali pa izolirane točke).

- Navadno krivuljo omejimo na končni obseg (*finite field*) $GF(p^m) = \mathbb{Z} \bmod p$, torej obseg vseh celih števil po modulu p, kjer je p veliko praštevilo.

Eliptične krivulje

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



Grafični prikaz seštevanja točk *S* in *P* na eliptični krivulji. Seštevek obeh točk lahko enoznačno opišemo s tretjo točko *R*, ki je preslikava presečišča krivulje s premico skozi *S* in *P* preko osi x: *S+P=R* (primer a) na sliki). Točko *–P* definiramo kot preslikavo točke *P* preko osi x (primer b)). Če se premica samo dotika krivulje na točki, se ta točka šteje dvakrat (primer c, na sliki*: P+P=R*). Če je premica vzporedna osi y, definiramo tretjo točko *0* kot točko "v neskončnosti" (primera b) in d) na sliki: *P+(-P)=0*). Natanko en od teh pogojev velja za vsak par točk na eliptični krivulji. Rezultat seštevanja torej vedno leži na izvorni eliptični krivulji

(vir: http://www.certicom.com/index.php/ecc-tutorial).

Eliptične krivulje

Preko operacije seštevanja točk lahko definiramo tudi množenje točke s skalarjem kot $2P = P + P$.

Za vsako točko $\boldsymbol{P}$ na eliptični krivulji v obsegu $GF(p^m)$ velja

$$\lim_{k\to\infty} k\boldsymbol{P} = \boldsymbol{0}$$

torej za vsako točko obstaja par skalarjev α in β, $\beta > \alpha$, za katere velja $\alpha\,\boldsymbol{P} = \beta\boldsymbol{P}$. Iz tega sledi $c\boldsymbol{P} = \boldsymbol{0}$ kjer je $c = \beta - \alpha$. Najmanjši c, ki izpolnjuje ta pogoj se imenuje red točke $\boldsymbol{P}$ (*order of the point*).

Za zagotovitev varnosti je potrebno izbrati takšno krivuljo in fiksno točko $\boldsymbol{F}$ na njej, da je red točke $\boldsymbol{F}$ veliko praštevilo. Namreč, če je red točke $\boldsymbol{F}$ n-bitno praštevilo, potem je za izračun faktorja k, iz $k\boldsymbol{F}$ potrebnih vsaj $2^{n/2}$ računskih operacij. Ta lastnost naredi eliptične krivulje privlačne, saj lahko z njihovo uporabo zagotovimo enako stopnjo kriptografske varnosti ključev in podpisov pri precej manjših ključih kot z RSA.

Izbira parametrov a, b, $GF(p^m)$ in c se običajno ne opravi ločeno za vsakega udeleženca v komunikaciji, saj gre za štetje števila točk na krivulji, ki je zamudno in težavno s stališča implementacije. Zaradi tega je več institucij za standarde objavilo varne domene parametrov eliptičnih krivulj:

- NIST, Priporočila za uporabo eliptičnih krivulj za potrebe vlade ZDA (http://csrc.nist.gov /groups/ST/toolkit/ documents/dss/NISTReCur.pdf)
- SECG, SEC 2: Priporočila za izbiro parametrov eliptičnih krivulj (http://www.secg.org/ download/aid-386/sec2_final.pdf)

Algoritem ECMQV

- Algoritem ECMQV [Menezes 1995] je razširitev Diffie–Hellmanovega algoritma na problem eliptičnih krivulj. Njegovo ime je sestavljeno iz akronima za eliptične krivulje (EC) in začetnic njegovih avtorjev Menezes, Qu in Vanstone.
- **Algoritem:** Alica in Bob se javno dogovorita o eliptični krivulji in o stalni točki $\boldsymbol{F}$ na tej krivulji. Alica izbere skrivno naključno celo število A_k, ki je njen skrivni ključ, in objavi točko na krivulji $A_P = A_k\boldsymbol{F}$ kot njen javni ključ. Bob sledi enakemu postopku: izbere skrivno število B_k in objavi svoj javni ključ $B_P=B_k\,\boldsymbol{F}$.
- Recimo, da želi Alica poslati sporočilo Bobu na kriptografsko varen način. V ta namen lahko preprosto izračuna $k = A_kB_P$ in rezultat uporabi kot skrivni ključ za konvencionalni simetrični kodirnik. Bob lahko izračuna isto število B_kA_P saj velja:

$$B_k\boldsymbol{A_P} = B_k\,(A_k\,\boldsymbol{F}) = A_k\,(B_k\,\boldsymbol{F}) = A_k\,\boldsymbol{B_P}$$

- Varnost izmenjave temelji na dejstvu, da je težko izračunati faktor k, če sta dani samo točki na krivulji $\boldsymbol{F}$ in $k\boldsymbol{F}$. Problem diskretnih logaritmov eliptičnih krivulj velja za NP-težek problem.

Primerjava asimetričnih kodirnikov

- **Intelektualne pravice in zaščita:** Patent algoritma RSA je potekel. Določeni koraki algoritmov z eliptičnimi krivuljami so še vedno patentno zaščiteni, čeprav nekateri avtorji zagotavljajo, da lahko izmenjavo ključev implementiramo brez kršitve patentnih pravic. Vsekakor je potrebna pri implementaciji kodirnikov, ki temeljijo na eliptičnih krivuljah dobršna mera previdnosti.
- Primerjava kriptografske varnosti ključev RSA in ključev eliptičnih krivulj (Elliptic curve cryptoghraphy – ECC) *(vir: Gura et al: Comparing Elliptic Curve Cryptography and RSA on 8-bit CPUs Cryptographic Hardware and Embedded Systems - CHES 2004)*

Pričakovani čas, potreben za uspešen napad (v MIPS-letih)	Velikost ključa RSA (v bitih)	Velikost ključa ECC (v bitih)
10^4	512	106
10^8	768	132
10^{11}	1024	160
10^{20}	2048	210
10^{78}	21000	600

Primerjava asimetričnih kodirnikov

- Kanadsko podjetje Certicom (http://www.certicom.com) je opravilo številne študije in promocije kodirnikov z eliptičnimi krivuljami (ECC). Tabela podaja nekaj njihovih primerjav časovnih zahtevnosti algoritmov ECC z algoritmom RSA.

Korak algoritma	RSA – 1024 bitni ključ (v ms)	ECC – 163 bitni ključ (v ms)
Generiranje ključa	4708,3	3,8
Podpisovanje dokumenta	228,4	2,1 (ECNRA) 3,0 (ECDSA)
Verifikacija podpisa	12,7	9,9 (ECNRA) 10,7 (ECDSA)

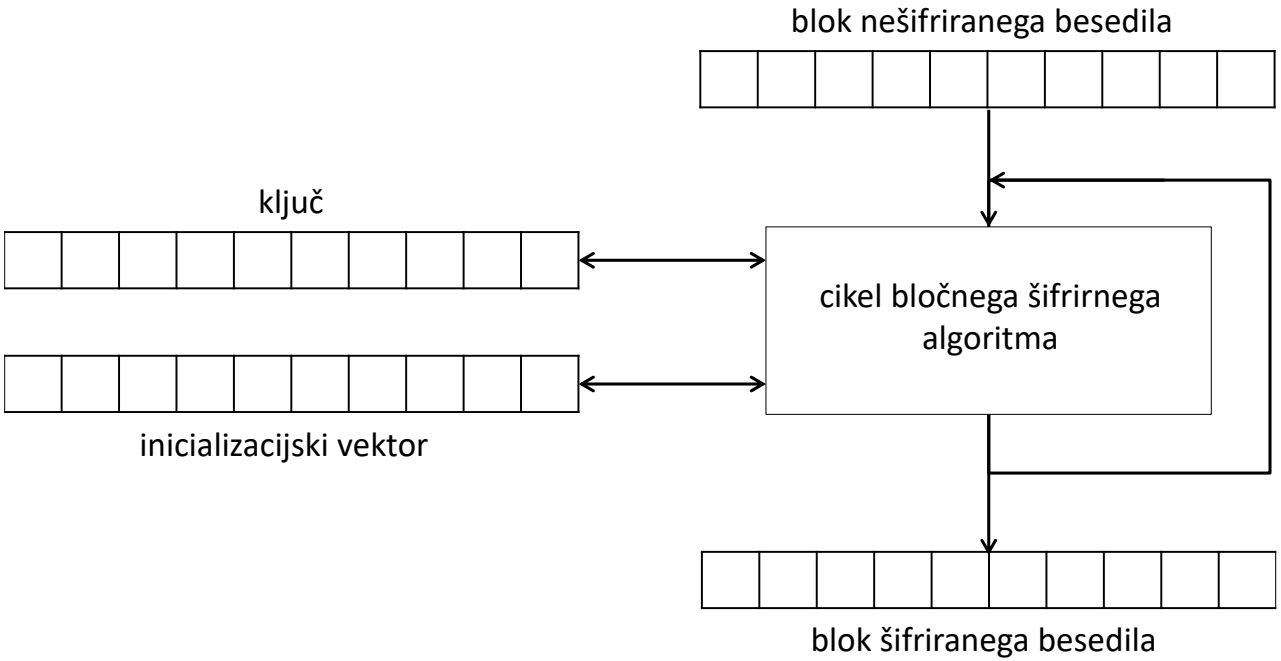
Simetrični kodirniki

Bločni simetrični kodirniki
(Block Ciphers)

- Bločni kodirnik je algoritem za šifriranje blokov podatkov (nekodiranega besedila) s **konstantno dolžino** v naključno zaporedje znakov iste dolžine (šifrirano besedilo) z uporabo drugega bloka podatkov, ki se imenuje **ključ**.
- V simetričnem načinu kodiranja se uporablja **isti ključ za enkripcijo in dekripcijo**. Ker je dešifriranje brez poznavanja ključa računsko zelo zahtevna operacija so šifrirani podatki zavarovani pred nezaželeno prisluškovanjem, vsaj dokler je ključ poznan samo oddajniku in sprejemniku.
- Bločni kodirnik je navadno sestavljen iz **več krogov preprostih kriptografskih operacij**. Šifrirni ključ se najprej razširi na več podključev (***key schedule***), podključi pa nato v več različnih ciklih premešajo in kodirajo vhodni podatkovni blok.
- Tipično so omenjena kodiranja realizirana s pomočjo bitnih operacij XOR. Zaradi svoje visoke zmogljivosti se bločni kodirniki pogosto uporabljajo v različnih aplikacijah varne komunikacije, kot so osnovno šifriranje podatkov v internetnih protokolih (IPsec in SSL / TLS), brezžične komunikacije in upravljanje z digitalnimi pravicami.

Bločni simetrični kodirniki
(Block Ciphers)

- Simetrični bločni kodirniki temeljijo na konceptu **zmede** in **difuzije vhodnih podatkov**.
- Tipični predstavniki implementacije teh konceptov so **mreže za zamenjavo in permutacijo** (*Substitution-Permutation Network - SPN*) in **mreže Feistel** (*Feistel network*).

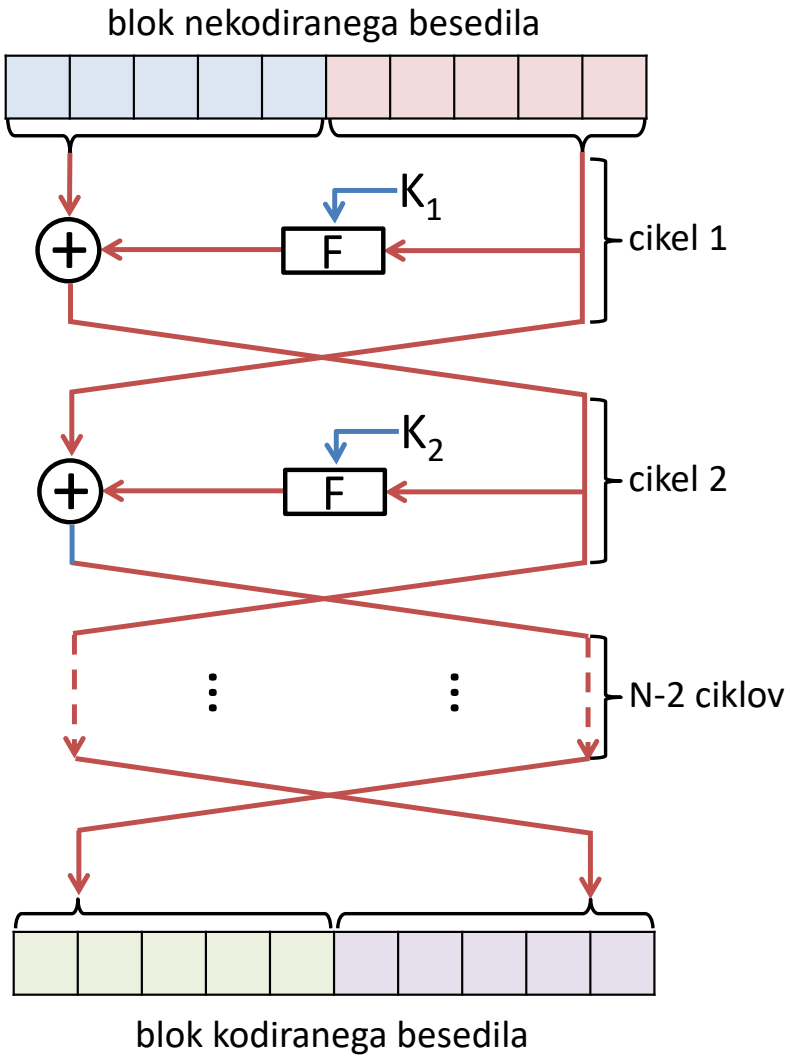


Mreže Feistel

(Feistel network)

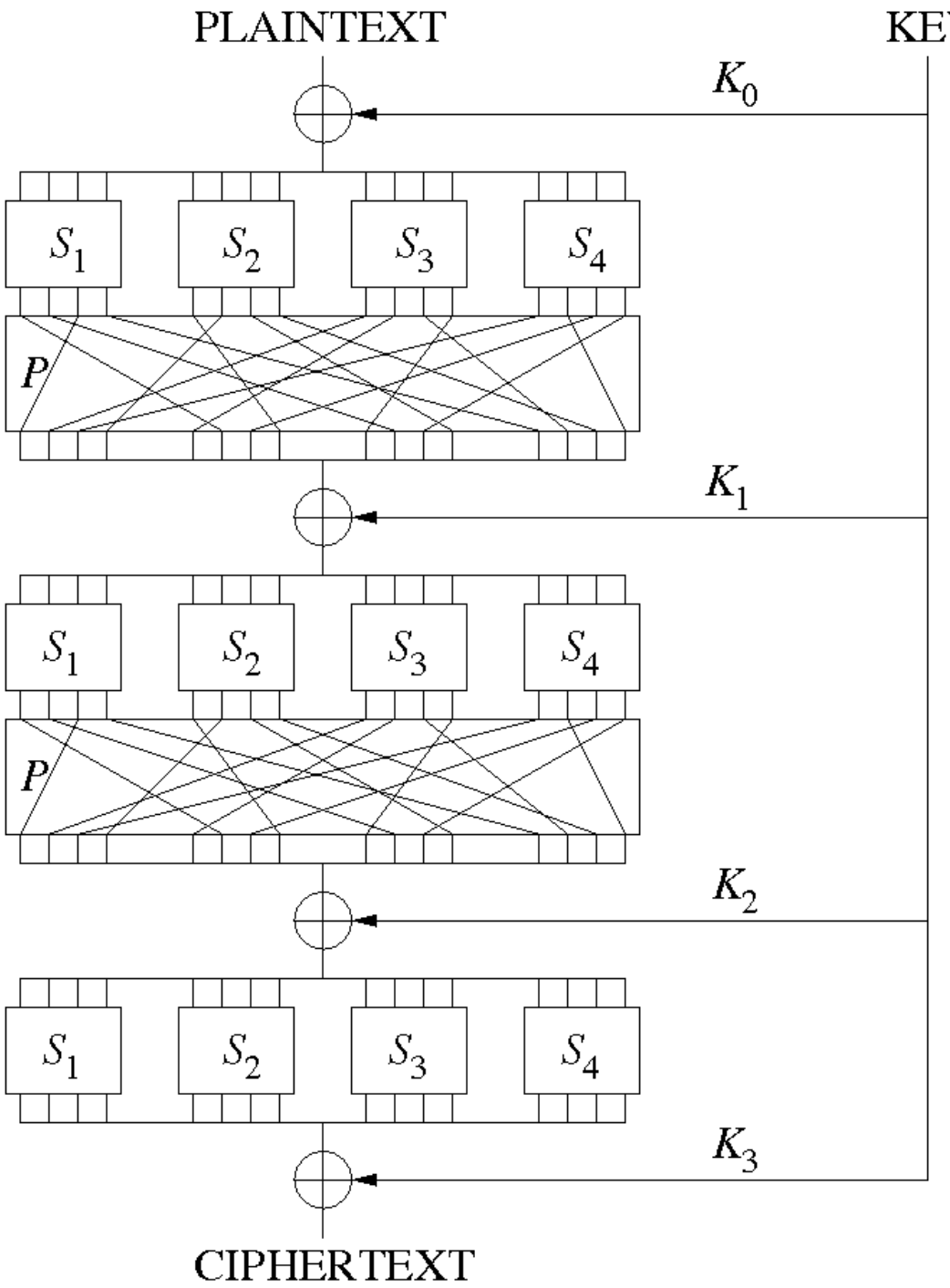
Horst Feistel

- V i-tem ciklu mreže Feistel se desna polovica vodnih podatkov (označena z X_i) posreduje na vhod funkcije *F*. Slednja prejme kot vhodni parameter tudi podključ K_i, pogosto pa je sestavljena iz korakov za mešanje ključa (**S-blokov**) in linearne transformacije.
- Izhod iz funkcije *F*, označen z Y_i, je preko bitne operacije XOR združen z levo polovico vhodnih podatkov X_{i-1}.
- Izhod cikla je nato oblikovan z zamenjavo vhodnih podatkov X_i in kodiranih podatkov $X_{i-1} \oplus Y_i$, kjer je "$\oplus$" bitni XOR.



Mreže za zamenjavo in permutacijo

(Substitution-Permutation Network - SPN)



premešanje s podključem

S-blok

difuzija (permutacija) izhodnih podatkov

https://en.wikipedia.org/wiki/File:SubstitutionPermutationNetwork2.png

Mreže za zamenjavo in permutacijo

(Substitution-Permutation Network - SPN)

- Med šifriranjem s kodirnikom SPN, se običajno vhodni podatki v vsakem ciklu premešajo s podključem, nato pa vstopijo v blok za zamenjavo (*Substitution boxes*) ali krajše **S-blok** (*S-box*).
- Izhodi S-blokov se dodatno spremenijo z linearno transformacijo, katere namen je širitev (difuzija) statističnih učinkov kodiranja v izhodnih podatkih (dobri kodirniki zagotavljajo izhodne podatke, ki navidezno in statistično ne odstopajo od naključno tvorjenjih zaporedij znakov).
- Dešifriranje je sestavljeno iz inverzne linearne transformacije, inverznega S-bloka in mešanja podključev v obratnem vrstnem redu.
- Z namenom, da se ohrani enaka pretočnost podatkov tako pri kodiranju kot pri dekodiranju se v SPN tipično izpustijo linearne transformacije zadnjega cikla.

S-blok

(S-box)

- Vsak S-blok velikosti m × n opravlja nelinearno preslikavo m hodnih bitov v n izhodnih bitov kar ustvarja navidezno zmedo v podatkih.
- Implementiran je lahko v obliki poizvedbene tabele (*lookup table*) z 2^m besedami dolgimi n bitov. Običajno so tabele fikse, kot npr. v Data Encryption Standard (DES), vendar nekateri kodirniki tabele ustvarijo dinamično, na podlagi ključa (npr. algoritma Blowfish in Twofish).
- Učni primer S-bloka predstavlja tabela z 6×4-biti kodirnika DES (S_5):

DES		Srednji štirje vhodni biti															
		0000	**0001**	**0010**	**0011**	**0100**	**0101**	**0110**	**0111**	**1000**	**1001**	**1010**	**1011**	**1100**	**1101**	**1110**	**1111**
Zunanja vhodna bita	**00**	0010	1100	0100	0001	0111	1010	1011	0110	1000	0101	0011	1111	1101	0000	1110	1001
	01	1110	1011	0010	1100	0100	0111	1101	0001	0101	0000	1111	1010	0011	1001	1000	0110
	10	0100	0010	0001	1011	1010	1101	0111	1000	1111	1001	1100	0101	0110	0011	0000	1110
	11	1011	1000	1100	0111	0001	1110	0010	1101	0110	1111	0000	1001	1010	0100	0101	0011

Glede 6-bitno vhodno zaporedje se 4-bitni izhod določi tako, da s pomočjo zunanjih dveh bitov (prvega in zadnjega bita) določi vrstica tabele, sredinski štirje biti pa določajo stolpec tabele. Na primer, vnos "100100" ima zunanja bita "10" in notranje bite "0010", kar ustreza izhodu "0001".

S-blok

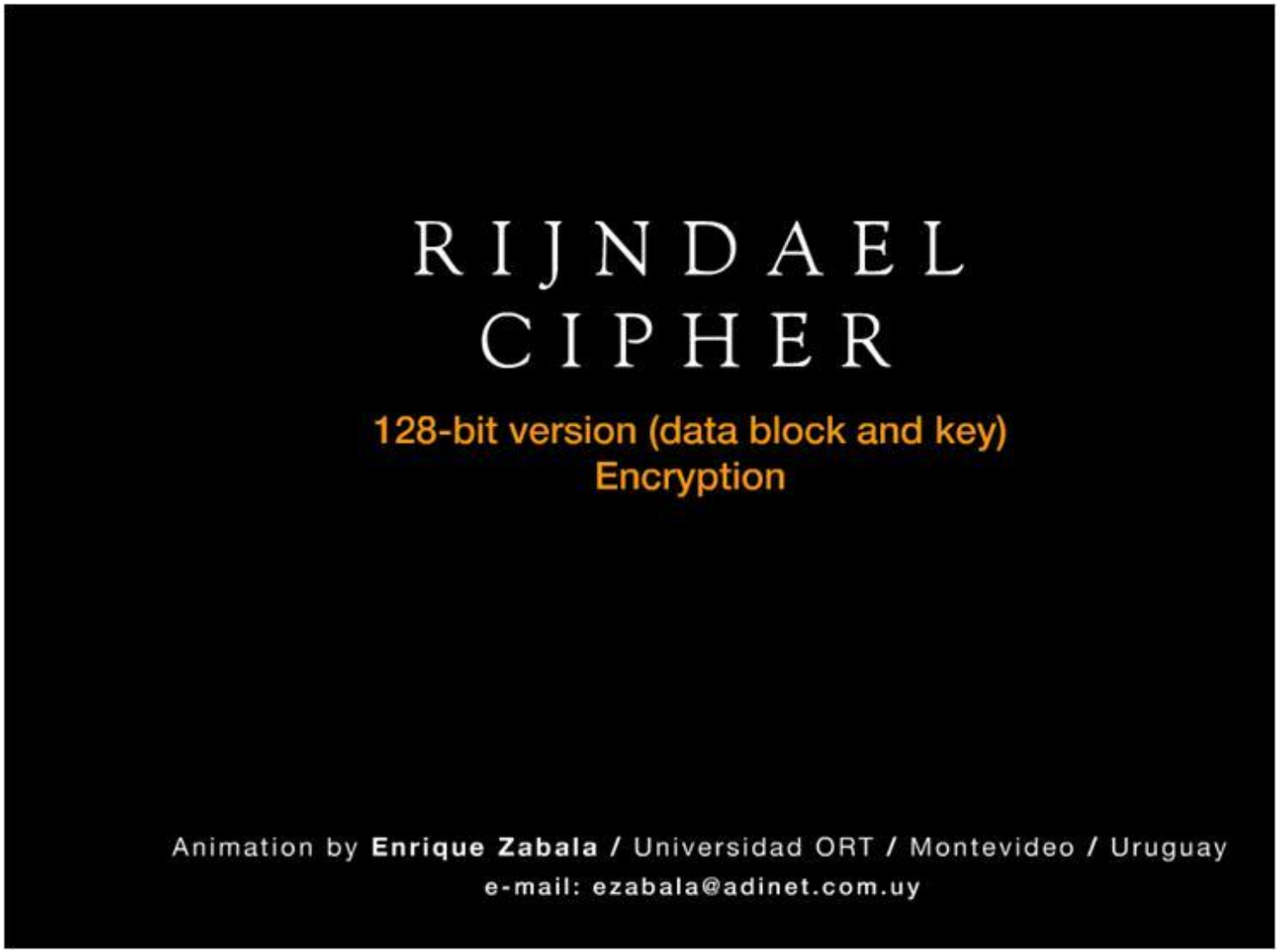
(S-box)

- Različni S-bloki v bločnem kodirniku lahko imajo različne preslikave ali pa se enaka preslikava uporablja v vseh S-blokih kodirnika.
- Kot edina nelinearna komponenta v obeh predstavljenih kodirnih arhitekturah, so bili **S-bloki predmet intenzivnih varnostnih študij** saj se je sumilo, da imajo **vgrajene ranljivosti**, ki so znane oblikovalcem kodirnih algoritmov. Pokazano je bilo, da temu ni tako in da so **S-bloki skrbno zasnovani v smeri maksimalne odpornosti na diferencialno kriptoanalizo**. Druge raziskave so še pokazale, da lahko tudi majhne spremembe S-bloka bistveno oslabijo varnost bločnih kodirnikov.
- Nedavne pobude v kriptografiji so osredotočene na razvoj novih standardov bločnih kodirnikov. Kot naslednik Data Encryption Standard (DES) se je pojavil algoritem Rijndael, ki ga je ameriški Nacionalni inštitut za standarde in tehnologijo novembra 2001 izbral za **Advanced Encryption Standard (AES)**

AES – Advanced Encryption Standard

1. **Razširitev ključa** (*KeyExpansion*): Iz kriptografskega ključa se po posebnem postopku izračunajo ključi posameznih ciklov.
2. **Začetni cikel**
   - Prištevanje ključa (*AddRoundKey*): vsak zlog v stanju algoritma AES se preko bitne operacije xor kombinira s istoležnim zlogom ključa.
3. **Osrednji cikel**
   - Substitucija zlogov (*SubBytes*): nelinearna sustitucija zlogov v kateri se vsak zlog stanja algoritma AES nadomesti z ustrezno vrednostjo v poizvedbeni tabeli (S blok).
   - Premik vrstic (ShiftRows): vsaka vrstica stanja se premakne ciklično za določeno število mest v levo. Ta korak, skupaj z mešanjem stolpcev, zagotavlja učinek difuzije podatkov.
   - Mešanje stolpcev (*MixColumns*): vsak stolpec stanja AES se s pomočjo obrnljive linearne transformacije preslika v izhoden stolpec, kjer vrednosti vseh štirih vhodnih zlogov vplivajo na vrednost vsakega izmed štirih izhodnih zlogov.
   - Prištevanje ključa (*AddRoundKey*): enako kot v začetnem ciklu.
4. **Zaključni cikel** (brez mešanja stolpcev)
   - Substitucija zlogov (*SubBytes*): enako kot v osrednjem ciklu.
   - Premik vrstic (ShiftRows): enako kot v osrednjem ciklu.
   - Prištevanje ključa (*AddRoundKey*): enako kot v začetnem in osrednjem ciklu.

AES – Advanced Encryption Standard



vir: https://www.youtube.com/watch?v=mlzxpkdXP58

Pretočni simetrični kodirniki

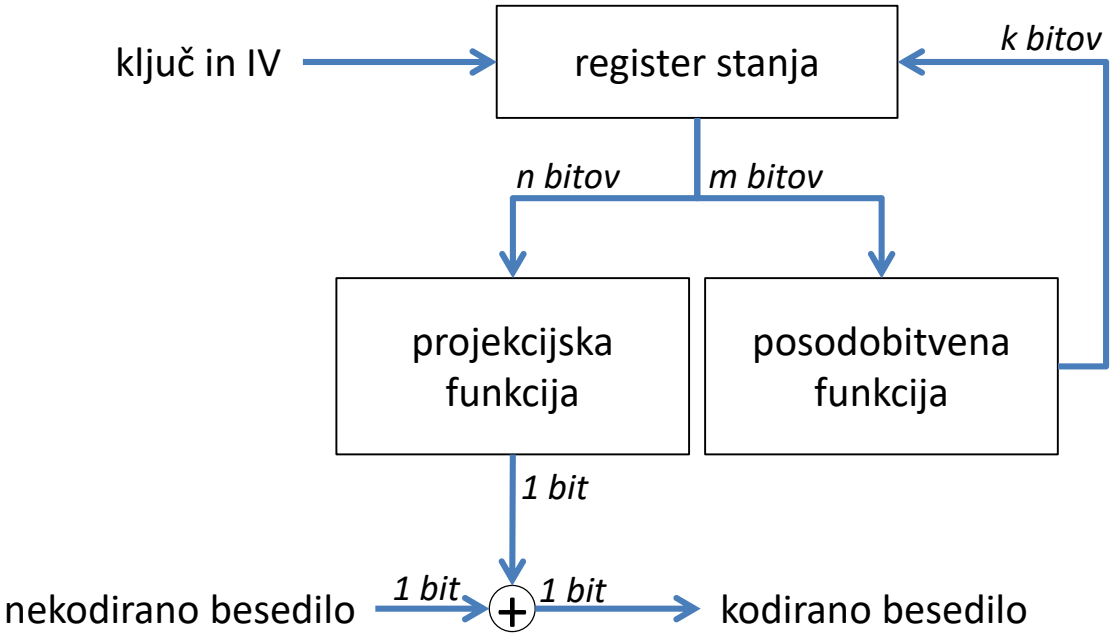
(Stream ciphers)

- Pretočni kodirnik je simetrični kodirnik, ki se uporablja v primerih, ko je količina podatkov vnaprej neznana, podatkovni tok pa je časovno nezvezen oz. heterogen.
- Kodirnik ustvarja zaporedje kriptografsko varnih bitov, imenovanih bitno zaporedje ključa (*key stream*). To zaporedje se nato na bitni ravni, z uporabo operacije xor, kombinira z nekodiranim ali kodiranim besedilom.
- Postopka enkripcije in dekripcije sta zaradi bijektivnosti operacije xor popolnoma enaka.
- Posodobitev internega stanja je lahko od prejetega besedila odvisna ali neodvisna. Glede na to delimo pretočne kodirnike v dve večji skupini:
  1. **sinhroni pretočni kodirniki** (*synchronous stream cipher*):
     - Interno stanje kodirnika se spreminja neodvisno od prejetega kodiranega oz. nekodiranega besedila
     - Pošiljatelj in prejemnik morata biti popolnoma sinhronizirana, drugače dekripcija ni mogoča.
     - Če se med pošiljanjem kodiranih podatkov podatkovnemu toku dodajo ali odvzamejo dodatni znaki se sinhronizacija izgubi.
  2. **Asinhroni oz. kodirniki s samodejno sinhronizacijo** (*asynchronous or self-synchronising stream cipher*):
     - Posodabljajo interno stanje glede na poslane/prejete znake kodiranega besedila.

Pretočni simetrični kodirniki

(Stream ciphers)

1 Osnovna topologija pretočnega kodirnika je sestavljena iz registra za shranjevanje internega stanja kodirnika (shift register), ki se na začetku določi s pomočjo ključa in inicializacijskega vektorja (IV). Vsebina registra se redno posodablja preko **funkcije za posodobitev stanja**.



2 Naslednja komponenta je **nelinearna projekcijska funkcija**, ki vzame del vsebine ali celotno vsebino registra s stanjem kodirnika in jo združi v en sam bit psevdonaključnega zaporedja kodirnika. Ta bit se nato preko operacije xor združi s trenutno obdelovanim bitom nekodiranega oz. kodiranega besedila.

3 Po operaciji xor se, preko funkcije za posodobitev, posodobi interno stanje kodirnika in postopek kodiranja se ponovi za naslednji bit nekodiranega oz. kodiranega besedila.