# Advanced Encryption Standards (AES). Algoritmid MARS, Serpent, RC6

Erika Matsak, PhD

# Advanced Encryption Standards (AES)

2. 01. 1997 teatas NIST (National Institute of Standards and Technology http://csrc.nist.gov/ ) uue standardi väljatöötamisest.

Eesmärgiks oli töötada välja uus standard, mis asendaks DES'i

Umbes üheksa kuud hiljem, 12. septembril esitati algoritmile järgmised nõudmised:

- Algoritm peab olema sümmeetriline, realiseeritud plokkšifrina, plokki suurus peab toetama 128 bitti ning võtme pikkus peab toetama 128, 192 ja 256 bitti
- 20. augustil 1998a esimesel AES konverentsil valiti 15 kandidaati.
- Teine AES konverents toimus märtsis 1999. Arutati algoritmide testimise tulemusi
- Augustis 1999 valitud 5 finalisti: MARS, RC6™, Rijndael, Serpent ja Twofish

# Advanced Encryption Standards (AES)

- Kõik viis finalisti kasutavad oma algoritmides plokkšifreerimist, iga ühel on meetod alamvõtmete tekitamiseks, lähtudes sisestatud võtmest. Raundide funktsioonid kasutavad igal raundil oma alamvõtmeid. Esimeseks ja viimaseks operatsiooniks on algoritmides mõned vormid andmete ja võtmete “segamisest”
- Olid ka mõned teised tehnilised omadused. Neli finalisti kasutasid S-box-e(AxB bitine sisend asendati B bitise väljundiga). Kolm finalisti kasutasid Feisteli võrku. Ülejäänud kaks finalisti kasutasid Feisteli võrgu asemel tervet plokki (mitte jagades seda vasakuks ja paremaks osaks) ning teostati sellega teisendusi, sh lineaarseid teisendusi

# AES finalistide lühikirjeldused

- **MARS** teostab teisendusi järgmises järjekorras: liitmine võtmega, 8 raundi permutatsioone ilma võtmeta, seejärel 8 raundi permutatsioone koos võtmega, siis 8 raundi pöördpermutatsioone koos võtmega ja lõpuks 8 viimast raundi pöördpermutatsioone ilma võtmeta. Need 16 raundi, mis on võtme kasutamisega nimetatakse krüptotuumaks. Selle sees kasutatakse 32-bitist võtme korrutamist, registrite nihutamist, ning liitmist võtmega. Raundid ilma võtmeta kasutavad kaks 8x18 bitist S-boxe, liitmist ja XOR. Kõikide raundide juures kasutatakse modifitseeritud Feisteli võrku, kus ¼ andmetest mõjutab ¾ andmeid. Nimetatud algoritm on esitatud IBM poolt.

# AES finalistide lühikirjeldused

- **RC6** on parametriseeritav šifreerimisalgoritmide pere, milles kasutatakse Feisteli võrku. AES'i jaoks oli tehtud ettepanek kasutada 20 raundi. Raundi funktsioon kasutab muutuvaid tsüklilisi registri nihutamisi, mida arvutatakse ruutfunktsiooni abil andmetest. Samuti sisaldavad raundid korrutamist (mod $2^{32}$), liitmist, XOR ning võtmega liitmist. RC6 on esitatud labori RSA poolt.

# AES finalistide lühikirjeldused

- **Rijndael** algoritm kasutab lineaarseid asendusteisendusi ning 10, 12 või 14 raundi, sõltuvalt võtme pikkusest. Andmete plokk jagatakse baitide massiivideks, ning iga operatsioon on bait-orienteerutud. Raundi funktsioon koosneb neljast kihist. Esimeses kihis rakendatakse 8x8 S-box'e. Teine ja kolmas kiht kasutavad lineaarset permutatsiooni, kus read vaadeldakse kui massiive nihutamiseks ning tulbad segatakse. Neljandas kihis teostatakse XOR alamvõtmete baitidega. Algoritm on esitatud Joan Daemen'i (Proton World International) ja Vincent Rijmen'i (Katholieke Universiteit Leuven) poolt.

# AES finalistide lühikirjeldused

- **Serpent** kasutab lineaarseid asendusteisendusi ning 32 raundi. Algoritmis on alg- ning lõpp-permutatsioonid, mis kergendavad alternatiivset režiimi *bitslice*. Raundi funktsioon koosneb kolmest kihist: XOR võtmega, paralleelne rakendus ühest kaheksast S-boxist ning lineaarne permutatsioon. Viimasel raundil on kiht XOR'ga asendatud lineaarse teisendusega. Algoritm on esitatud Ross Anderson'i (University of Cambridge), Eli Biham'i (Technion) и Lars Knudsen'i (University of California, San Diego) poolt.

# AES finalistide lühikirjeldused

- **Twofish** on Feisteli võrk 16 raundiga. Feisteli võrku on natuke modifitseeritud ühebitiste rotatsioonide kasutamisega. Raundi funktsioon mõjutab 32-bitist sõna neljast võtmest sõltuvate S-Boxiga, mille tulemus korrutatakse spetsiaalse 4x4 maatriksiga), sellele järgneb pseudo-Hadamari teisendus (Pseudo-Hadamar Transform, PHT) ning võtme lisamine. Algoritm oli esitatud järgmiste inimeste ja asutuste poolt: Bruce Schneier, John Kelsey ja Niels Ferguson (Counterpane Internet Security, Inc.), Doug Whiting (Hi/fn, Inc.), David Wagner (University of California, Berkley) ning Chris Hall (Princeton University).

# AES hindamiskriteeriumid

- Kolmas AES konverents toimus aprillis 2000. Konverentsil arutati esitatud algoritmide kohta esitatud kommentaare. Avalik arutelu lõpetati 15.mai 2000.
- Hindamiskriteeriumid olid jagatud kolmeks:
  - Turvalisus
  - Hind
  - Algoritmi ja realisatsiooni karakteristikud
- Turvalisus on väga oluline faktor, siin analüüsitakse algoritmi matematilisi aluseid, väljundandmete juhuslikust ning suhtelist turvalisust võrreldes teiste kandidaatidega.
- Hind peegeldab nõudeid litsentseerimisele, arvutuste efektiivsusele (kiirusele) erinevate riistvaraliste platvormide korral ning nõudmisi mälule. AES eesmärgiks oli leida avatud/tasuta litsentsiga algoritme. Seejuures arutati intellektuaalse omandi ning võimalike konfliktidega seotud probleeme.
- Algoritmi ja realisatsiooni karakteristikud: paindlikus, algoritmi lihtsus, riistvaraline ja tarkvaraline vastavus.

# AES hindamiskriteeriumid

Kiirus. Krüpteerimine ja dekrüpteerimine erinevatel platvormidel

| | 32 bitine protsessor (C) | 32 bitine protsessor (Java) | 64 bitine protsessor (C ja Assembler) | 8 bitine protsessor (C ja Assembler) | 32 bitine smart – kaart (ARM) | Digital signal protsessor |
|---|---|---|---|---|---|---|
| MARS | II | II | II | II | II | II |
| RC6 | I | I | II | II | I | II |
| Rijndael | II | II | I | I | I | I |
| Serpent | III | III | III | III | III | III |
| Twofish | II | III | I | II | III | I |

I- kõige kiirem, III kõige aeglasem

# AES hindamiskriteeriumid

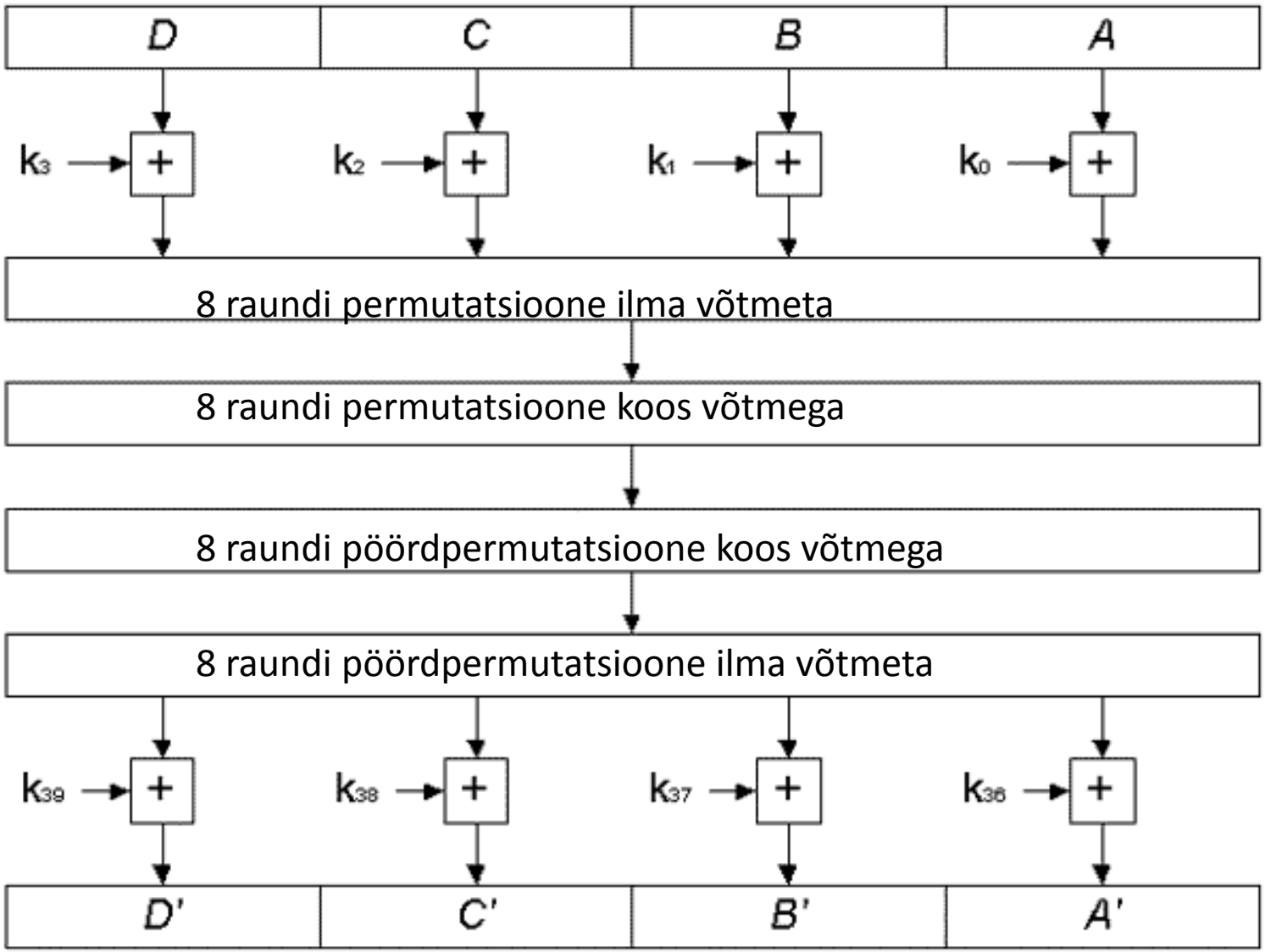
Operatsioonid võtmetega , alamvõtmete genereerimine (key derivation)

| | 32 bitine protsessor (C) | 32 bitine protsessor (Java) | 8 bitine protsessor (C ja Assembler) | 8 bitine protsessor (C ja Assembler) | Digital signal protsessor |
|---|---|---|---|---|---|
| MARS | II | II | III | II | II |
| RC6 | II | II | II | III | II |
| Rijndael | I | I | I | I | I |
| Serpent | III | II | II | III | I |
| Twofish | III | III | III | II | III |

Kiirust mõjutavad samuti võtme pikkus ja plokkšifri režiim (näiteks ECB - Electronic Codebook Mode või mõni teine)

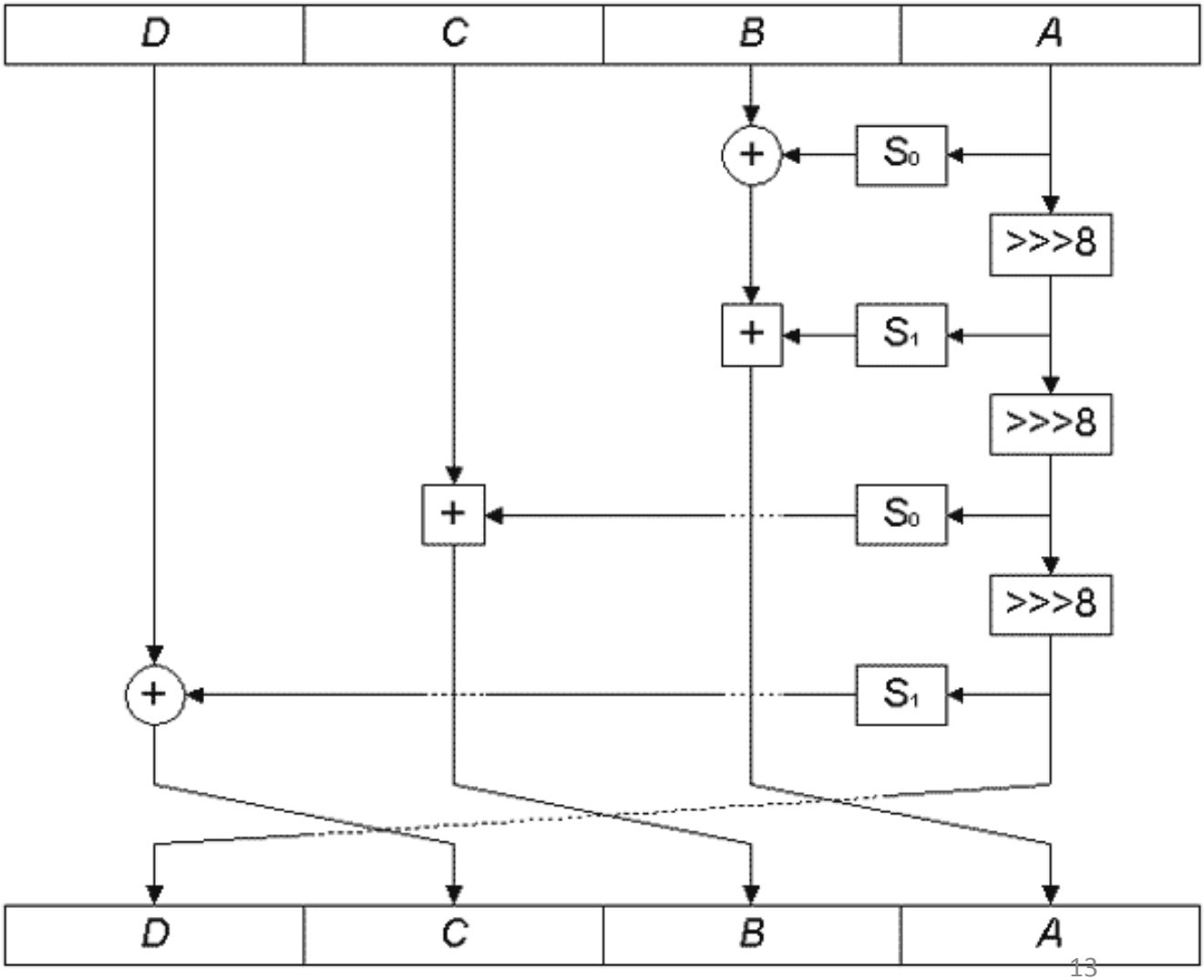
Iga plokk -32 bitti

# Algoritm MARS



# Algoritm MARS, permutatsioon ilma võtmeta

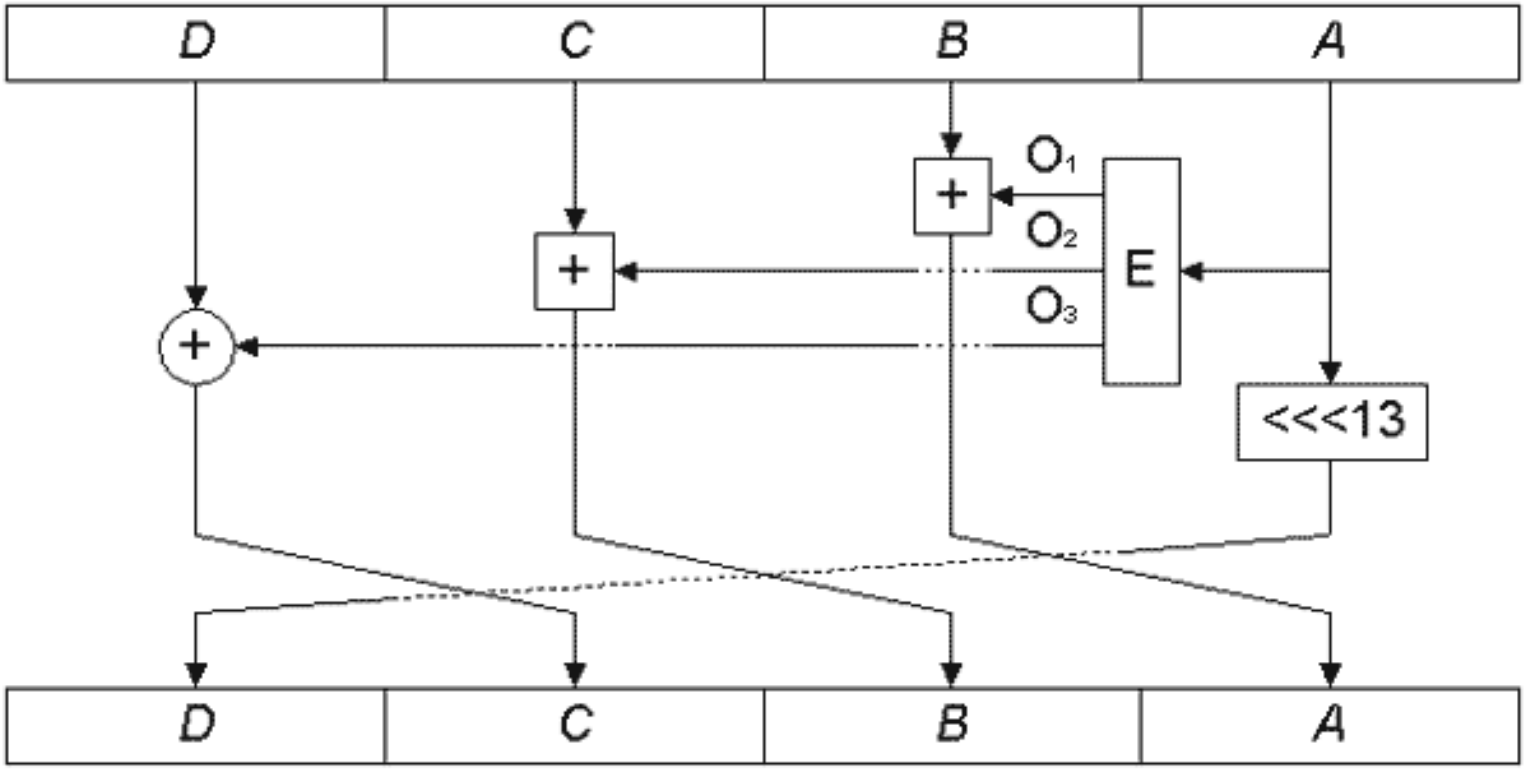
Algoritm kujutab endast laiendatud Feisteli võrku. Igal raundil töödeldakse üks alamplokk ning vastava operatsiooniga (liitmine või XOR) pannakse kokku teiste alamplokkidega, seejärel vahetatakse kohad. Konkreetsed teisendused sõltuvad raundi numbrist. Mõnede raundide vahel on ka lisaoperatsioone.



# Algoritm MARS, permutatsioon ilma võtmeta

- Raundil 0 ja 4, peale seda, kui S1 S-boxi tulemus on pandud kokku operatsiooniga XOR alamplokiga **D**, enne kirjutamist alamplokki **C,** teostatakse lisaks ka liitmist (mod $2^{32}$) alamplokiga **A**.
- Raundil 1 ja 5 alamplokk **B** analoogiliselt liidetakse alamplokiga **A (A=A+B)**.
- Algoritmi autorite sõnul muudavad nimetatud sammud algoritmi tunduvalt kindlamaks krüptoanalüüsi vastu.
- 32 bitine sõna, jagatud 8 osaks, sisestatakse S-boxi. Kui esimese osa bittidest koosneb 0-dest (vastav kümnendarv samuti 0), siis S-Boxi tulemus on arv, mis paikneb lahtris 0, ehk: 09D0C479. Kui mingile osale vastav kümnendarv arv on 1, siis tulemuseks on vastavalt teises lahtris paiknev arv, ehk: 28C8FFE0. www.tlu.ee/~matsak/crypto/MARS_S_boxid.doc

# Algoritm MARS, permutatsioon koos võtmega




Permutatsioon E laiendab sisestatud 32bitise sõna kolmeks 32 bitiseks sõnaks.
Iga saadud O-osaga teostatakse vastavaid operatsioone alamplokkidega.
Viimases alamplokis **A** nihutatakse registri bitte 13 positsiooni võrra vasakule.

# Algoritm MARS, permutatsioon E

$O_2 = I$,

$O_3 = O_2 <<< 13$,

$O_2 = O_2 + k_{2r+4} \bmod 2^{32}$,

$O_3 = O_3 * k_{2r+5} \bmod 2^{32}$,

$O_3 = O_3 <<< 5$,

$O_1 = S(O_2)$,

$O_1 = O_1 \oplus O_3$,

$O_2 = O_2 <<< O_3'$,

$O_3 = O_3 <<< 5$,

$O_1 = O_1 \oplus O_3$,

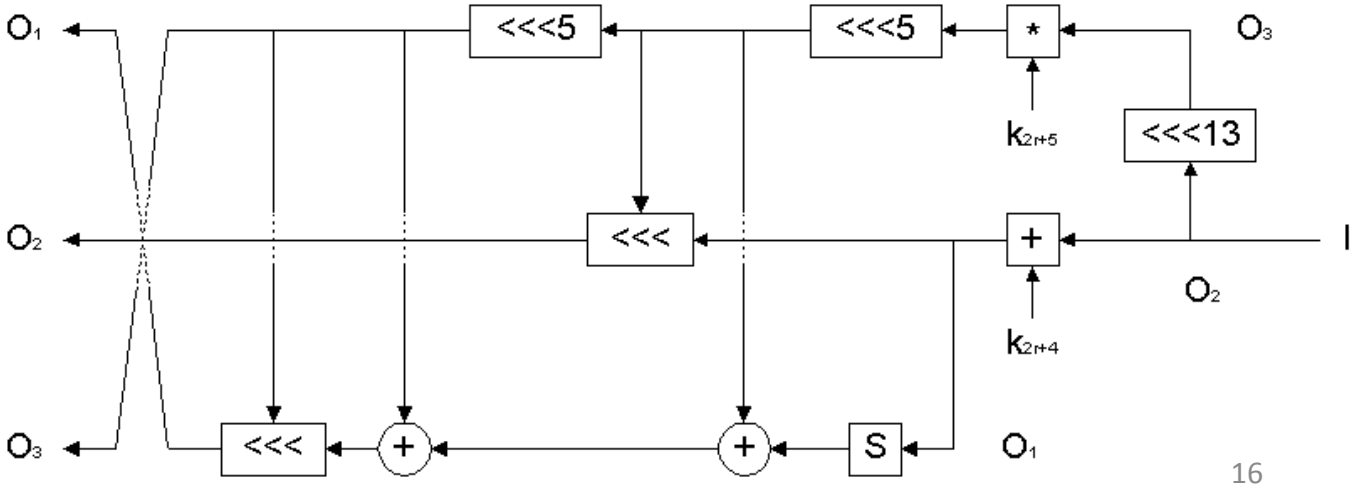
$O_1 = O_1 <<< O_3'$,

I - sisestatud 32 bitti

r – raundi number, alustades 0-st selle krüptotuuma sees

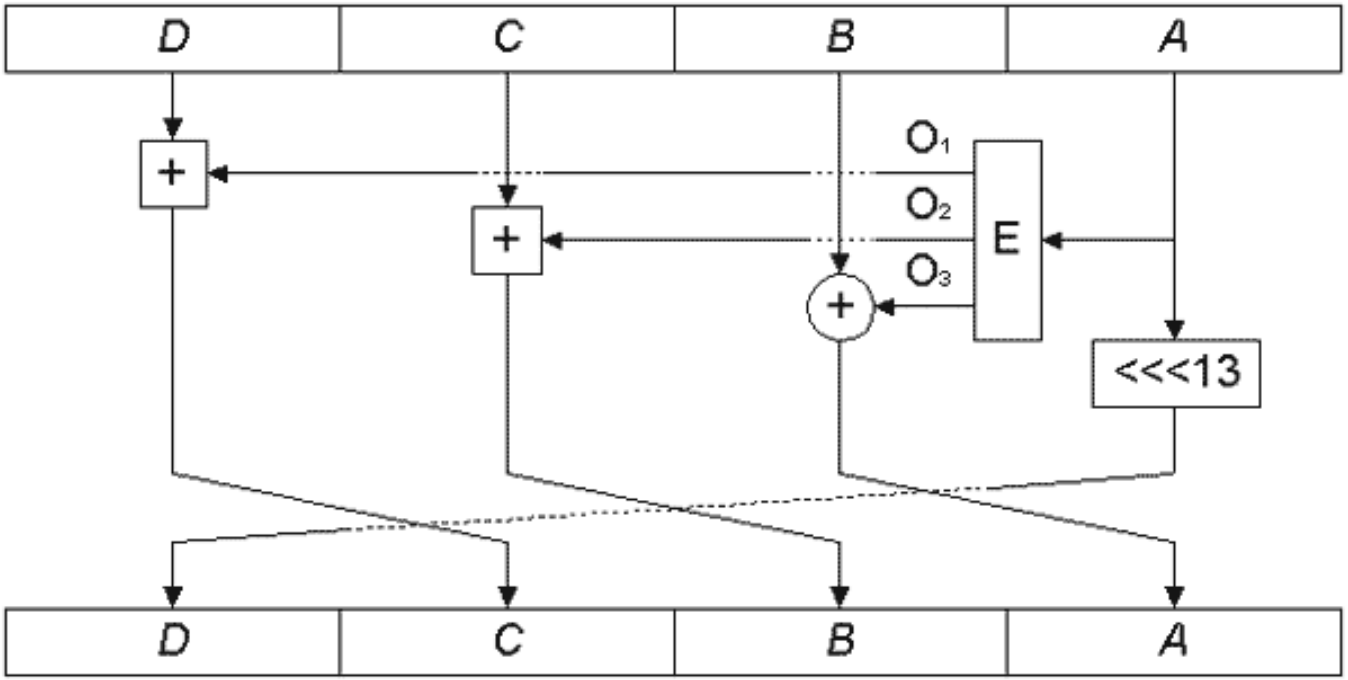
S - S-boxide $S_0$ ja $S_1$ ühendi kasutamine ($S_0 \cup S_1$ ), 512 väärtust, tulemus valitakse lähtudes sõna esimesest 9 bitist.

O' tähendab, et registri nihutamiseks kasutakse antud 5 bitiga esitatud kümnendarvu.



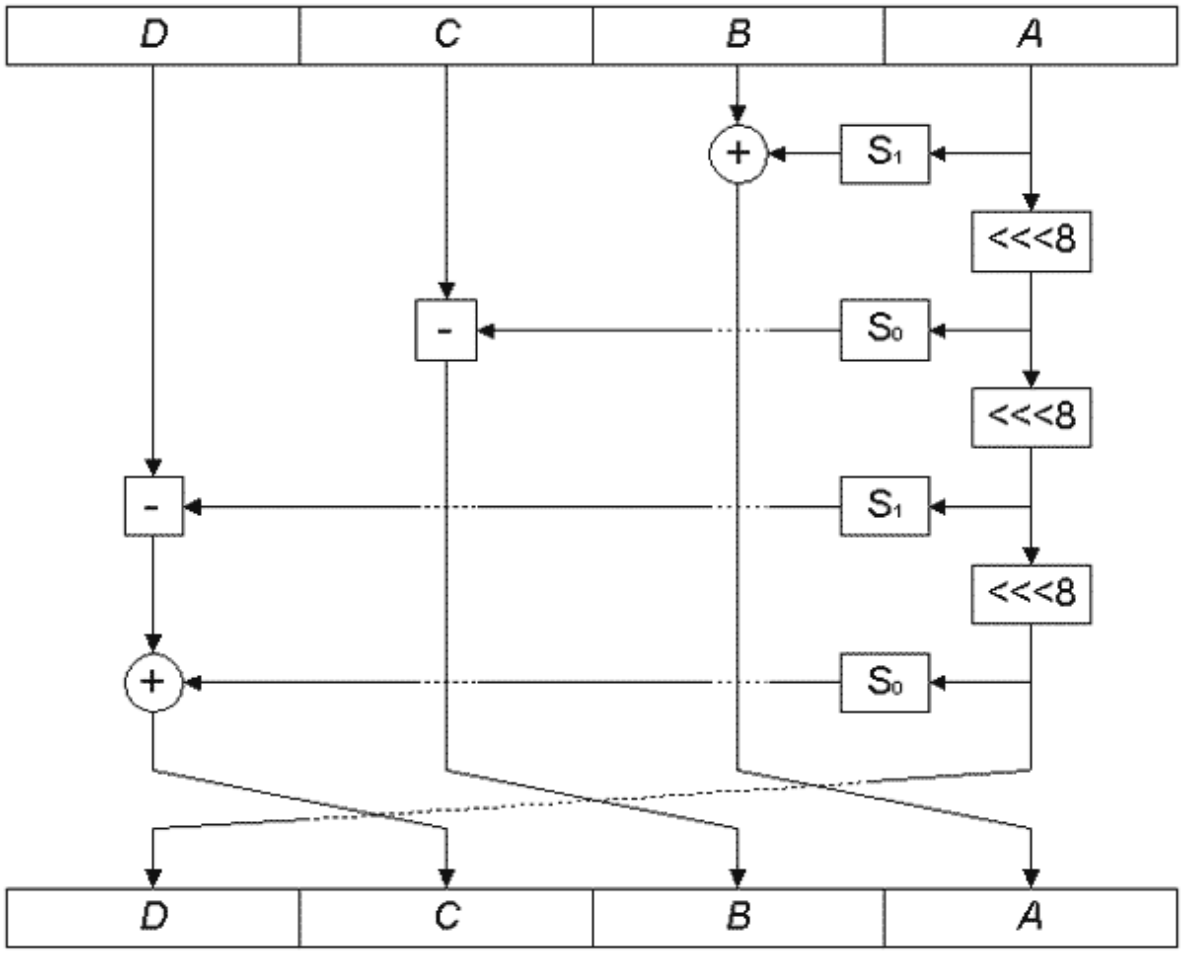
# Algoritm MARS, **pöörd**permutatsioon koos võtmega

"otse" permutatsiooni ja pöördpermutatsiooni vahe seisneb selles, et operatsioonide järjekord alamplokkide **B,C** ja **D** ja ploki **E** väljundite $\mathbf{O_1 ... O_3}$ vahel on muutunud.



# Algoritm MARS, **pöörd**permutatsioon ilma võtmeta

"Otse" permutatsiooni ja pöördpermutatsiooni vahe on siin märgatavam.
Nii nagu "otse" permutatsiooni juures on siin ka lisaoperatsioonid mõningate raundide lõpus.




Raundidel 1 ja 5, peale $S_0$ tulemuse ning alamploki **D** kokku arvutamist operatsiooniga XOR, enne kui andmed kirjutatakse uutele kohtadele, teostatakse lahutamine **A** ja **B** vahel, ehk $B=A-B \pmod{2^{32}}$

Raundidel 2 ja 6 teostatakse lahutamine alamplokkide **C** ja **B** vahel, ehk $B=C-B \pmod{2^{32}}$.

# Algoritm MARS, alamvõtmete genereerimine

- Võti saab olla erineva pikkusega 128 kuni 448 bitti, 32 biti-kordne.
- Genereeritakse 40 alamvõtit (igaüks 32 bitti) lähtudes antud võtmest
- 1. Moodustatakse ajutine massiiv $\mathbf{T_0 ... T_{14}}$:

  $\mathbf{T_0 = KI_0,}$
  $\mathbf{T_1 = KI_1,}$
  **...**
  $\mathbf{T_{n-1} = KI_{n-1},}$
  $\mathbf{T_n = n,}$
  $\mathbf{T_{n+1} = T_{n+2} = ... = T_{14} = 0,}$
  kus $\mathbf{KI_0 ... KI_{n-1}}$ on esialgne võti,
  **n** on selle võtme suurus sõnades, kus ühe sõna pikkus on 32 bitti (ehk 4 kuni 14).

# Algoritm MARS, alamvõtmete genereerimine

for (j=0 ; j<4; j++) {
    for (i=0; i< 14; i++){
    Lineaarne teisendus:
        $T_i = T_i \oplus ((T_{i-7 \bmod 15} \oplus T_{i-2 \bmod 15}) <<< 3) \oplus (4i + j)$
        kus i on iteratsiooni number
}}

!! NB!
Antud juhul
0-7 mod 15 = 8

for (j=0 ; j<4; j++) {
    for (i=0; i< 14; i++){
    Massiivi T permutatsioon:
        $T_i = (T_i \oplus S(\text{low 9 bit of } T_{i-1 \bmod 15})) <<< 9$
}}

    Valitakse 10 elementi ja paigutatakse uude, laiendatud võtmete massiivi

for (j=0 ; j<4; j++) {
        for (n=0; n< 10; n++){
            $k_{10j+n} = T_{4n \bmod 15}$
}}

# Algoritm MARS, alamvõtmete genereerimine

- Genereeritud alamvõtmetele esitatakse nõudmised:
  - Need alamvõtmed, mida rakendatakse funktsiooni E juures, ehk alamvõtmed paaritute indeksitega vahemikus $\mathbf{k_5}$ kuni $\mathbf{k_{35}}$ (kaasa-arvatud!) peavad olema paaritud arvud. Kusjuures just kaks esimest bitti peavad olema 1-d.
  - Samad alamvõtmed ei tohi sisaldada 10 järjest seisvat nulli (0) või ühte (1)

# Algoritm MARS, alamvõtmete genereerimine

Selleks, et meie alamvõtmed rahuldaksid nõudmisi, tuleb sooritada järgmisi operatsioone:

- Kaks esimest bitti alamvõtmest asendada väärtusega “1”. Vana väärtus salvestada muutujasse *j*. Tähistada uus alamvõti tervikuna sümboliga *W*.
- Arvutata *mask M,* mida kasutatakse selleks, et välistada järjest paiknevaid nulle või ühtesid:
  - Panna *maskis M* arv 1 esitama neid bitte, mis vastavad alamvõtmes kümnele ühesugusele bitile (st kümme nulli järjest või kümme ühte järjest), teistsuguseid bitte esitagu antud maskis arv 0.
  - Nullida need bitid, mida *maskis M tähistab arv* 1 ja mis seejuures vastavad ühele järgmistest tingimustest:
    i<2; i=31; $W_i \neq W_{i-1}$; $W_i \neq W_{i+1}$;
- Kasutatakse korrigeerivaid arve B (S boxi väärtused 265, 266, 267, 268), ehk $B_0$=a4a8d57b, $B_1$=5b5d193b, $B_2$=c8a8309b, $B_3$=73f9a978. Viimasena tuleb arvutada lõplikud alamvõtmed:
- $K_i = W \oplus ((B_j <<< K_{i-1}) \& M)$

# Algoritmi MARS miinused ja plussid.

- Eriliselt keeruline struktuur, mis kasutab erinevat tüüpi raunde ja teeb keeruliseks algoritmi analüüsi ning samuti selle realisatsiooni.
- Tekivad probleemid tarkvaralise realisatsiooniga nendel platvormidel, mis ei toeta 32 bitist korrutamist ning registri nihutamist muutuva bittide arvuga.
- Ei saa efektiivselt realiseerida riistvaraliselt ning juhul, kui ressursid on piiratud.
- Kehvalt toetab võimalusi laiendada võtit “jooksvas” režiimis

**Ainsaks plussiks on aga see, et šifreerimine ja dešifreerimine on praktiliselt ühtemoodi realiseeritav**

# Algoritm SERPENT

- Algoritm on loodud tuntud krüptoloogide poolt: Ross Anderson, Eli Biham, ning Lars Knudsen.
- Andmete plokk jagatakse neljaks alamplokiks, iga alamploki suurus on 32 bitti.
- Algoritm sisaldab 32 raundi. Enne esimest raundi ja peale viimast raundi on spetsiaalne permutatsioon.

Ross Anderson,
15 .09.1956

Eli Biham

Lars Knudsen
21.02.1962

# Algoritm SERPENT

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 32 | 64 | 96 | 1 | 33 | 65 | 97 | 2 | 34 | 66 | 98 | 3 | 35 | 67 | 99 |
| 4 | 36 | 68 | 100 | 5 | 37 | 69 | 101 | 6 | 38 | 70 | 102 | 7 | 39 | 71 | 103 |
| 8 | 40 | 72 | 104 | 9 | 41 | 73 | 105 | 10 | 42 | 74 | 106 | 11 | 43 | 75 | 107 |
| 12 | 44 | 76 | 108 | 13 | 45 | 77 | 109 | 14 | 46 | 78 | 110 | 15 | 47 | 79 | 111 |
| 16 | 48 | 80 | 112 | 17 | 49 | 81 | 113 | 18 | 50 | 82 | 114 | 19 | 51 | 83 | 115 |
| 20 | 52 | 84 | 116 | 21 | 53 | 85 | 117 | 22 | 54 | 86 | 118 | 23 | 55 | 87 | 119 |
| 24 | 56 | 88 | 120 | 25 | 57 | 89 | 121 | 26 | 58 | 90 | 122 | 27 | 59 | 91 | 123 |
| 28 | 60 | 92 | 124 | 29 | 61 | 93 | 125 | 30 | 62 | 94 | 126 | 31 | 63 | 95 | 127 |

Esimene permutatsioon

Lõpp-permutatsioon

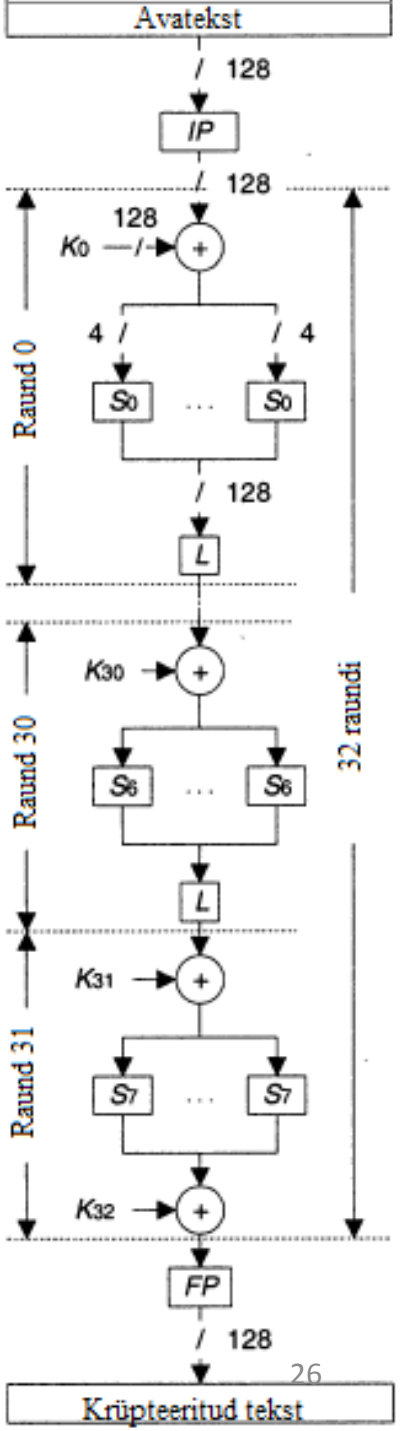
| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 4 | 8 | 12 | 16 | 20 | 24 | 28 | 32 | 36 | 40 | 44 | 48 | 52 | 56 | 60 |
| 64 | 68 | 72 | 76 | 80 | 84 | 88 | 92 | 96 | 100 | 104 | 108 | 112 | 116 | 120 | 124 |
| 1 | 5 | 9 | 13 | 17 | 21 | 25 | 29 | 33 | 37 | 41 | 45 | 49 | 53 | 57 | 61 |
| 65 | 69 | 73 | 77 | 81 | 85 | 89 | 93 | 97 | 101 | 105 | 109 | 113 | 117 | 121 | 125 |
| 2 | 6 | 10 | 14 | 18 | 22 | 26 | 30 | 34 | 38 | 42 | 46 | 50 | 54 | 58 | 62 |
| 66 | 70 | 74 | 78 | 82 | 86 | 90 | 94 | 98 | 102 | 106 | 110 | 114 | 118 | 122 | 126 |
| 3 | 7 | 11 | 15 | 19 | 23 | 27 | 31 | 35 | 39 | 43 | 47 | 51 | 55 | 59 | 63 |
| 67 | 71 | 75 | 79 | 83 | 87 | 91 | 95 | 99 | 103 | 107 | 111 | 115 | 119 | 123 | 127 |

# Algoritm SERPENT

Raundi funktsioon on suhteliselt lihtne:

1. 128 bitise võtme lisamine XOR operatsiooni abil
2. S-boxid. 128 bitine plokk jagatakse neljaks osaks, kus iga ploki suurus on 32 bitti, iga blokk töödeldakse vastava S-boxiga ning saadud tulemused pannakse kokku konkatenatsiooniga.

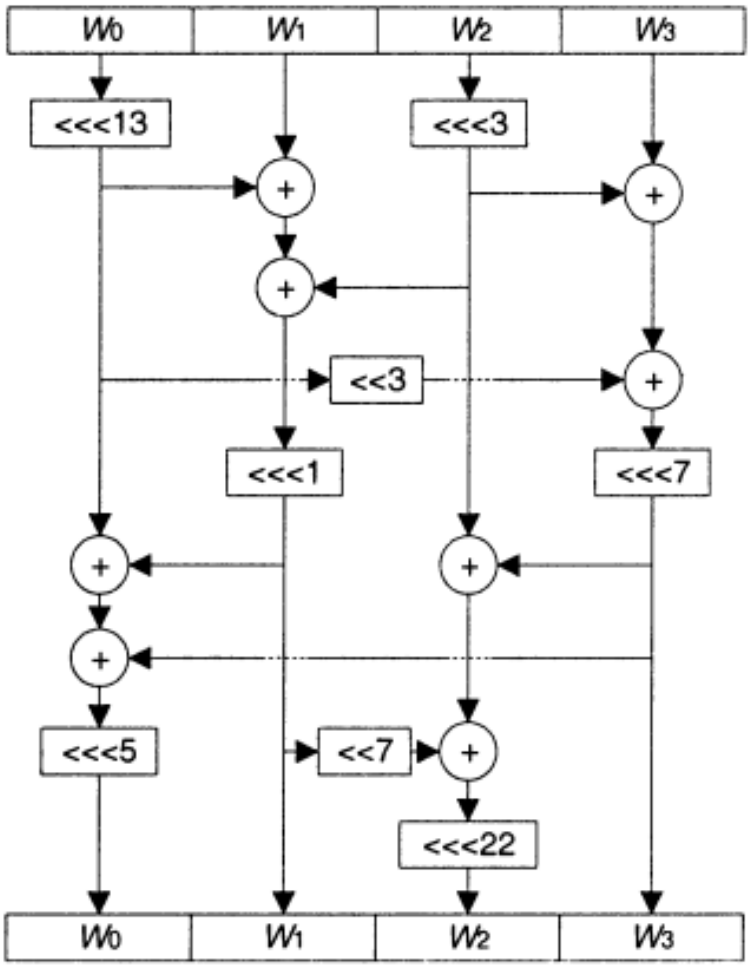
   Korraga kasutakse nelja ühesugust S-boxi, kokku on neid kaheksa, mida vahetatakse vastavalt raundile.
3. Lineaarne teisendus



# Algoritm SERPENT

Lineaarset teisendust saab teostada vastava tabeliga või valemite (algoritmi) abil.

$$W_0 = W_0 <<< 13;$$
$$W_2 = W_2 <<< 3;$$
$$W_1 = W_1 \oplus W_0 \oplus W_2;$$
$$W_3 = W_3 \oplus W_2 \oplus (W_0 << 3);$$
$$W_1 = W_1 <<< 1;$$
$$W_3 = W_3 <<< 7;$$
$$W_0 = W_0 \oplus W_1 \oplus W_3;$$
$$W_2 = W_2 \oplus W_3 \oplus (W_1 << 7);$$
$$W_0 = W_0 <<< 5;$$
$$W_2 = W_2 <<< 22,$$



# Algoritm SERPENT

Ühe raundi teisendusi saab esitada ka ühe valemi abil:

$$Y = L(S'_{(i \bmod 8)}(X \oplus K_i)),$$

kus X ja Y on vastavalt ploki sisend- ja väljundandmed
i – raundi number
$K_i$ – raundi võti
$L_i$ – lineaarne teisendus
S’ – tähendab paralleelset S-boxide kasutamist

Viimane raund erineb teistest:

$$Y = S'_7(X \oplus K_{31}) \oplus K_{32}.$$

# SERPENT. Võtmete laiendamine

- SERPENTI võti saab olla suvalise suurusega kuni 256 bitti. Raundis kasutatakse 128 bitist alamvõtit.
- Lühike võti kõigepealt "pikendatakse" järgmiste sammude abil:
  - Lisatakse bitt "1" paremale
  - Seejärel kirjutataks arvu "0" paremale, kuni võtme pikkus on 256 bitti.
  - Alustatakse võtme laienduse protseduuriga:
    - 256 bitti jagatakse kaheksaks osaks (iga osa 32 biti) $\{w_{-8}, w_{-1}\}$
    - Saadud osad ("sõnad") kasutatakse algandmetena selleks, et arvutada "sõnade " jada $w_1 \ldots w_{131}$.

$$w_i = (w_{i-8} \oplus w_{i-5} \oplus w_{i-3} \oplus w_{i-1} \oplus \phi \oplus i) <<< 11,$$

    - Lõpuks teisendatakse saadud tulemused (ehk $w_i$) S-boxide ($S_0 \ldots S_7$) abil

ümardatud kuldlõike murdosa $\phi = (\sqrt{5} - 2)/2$
ehk 9E3779B9

# SERPENT. Võtmete laiendamine

S-boxide kasutamine alamvõtmete arvutamiseks

$$K_0 = S_3(\{w_0, w_1, w_2, w_3\});$$

$$K_1 = S_2(\{w_4, w_5, w_6, w_7\});$$

$$K_2 = S_1(\{w_8, w_9, w_{10}, w_{11}\});$$

$$K_3 = S_0(\{w_{12}, w_{13}, w_{14}, w_{15}\});$$

$$K_4 = S_7(\{w_{16}, w_{17}, w_{18}, w_{19}\});$$

$$K_5 = S_6(\{w_{20}, w_{21}, w_{22}, w_{23}\});$$

$$K_6 = S_5(\{w_{24}, w_{25}, w_{26}, w_{27}\});$$

$$K_7 = S_4(\{w_{28}, w_{29}, w_{30}, w_{31}\});$$

$$\ldots$$

$$K_{31} = S_4(\{w_{124}, w_{125}, w_{126}, w_{127}\});$$

$$K_{32} = S_3(\{w_{128}, w_{129}, w_{130}, w_{131}\}).$$

# S-boxid

| | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| $S_0$ | 3 | 8 | 15 | 1 | 10 | 6 | 5 | 11 | 14 | 13 | 4 | 2 | 7 | 0 | 9 | 12 |
| $S_1$ | 15 | 12 | 2 | 7 | 9 | 0 | 5 | 10 | 1 | 11 | 14 | 8 | 6 | 13 | 3 | 4 |
| $S_2$ | 8 | 6 | 7 | 9 | 3 | 12 | 10 | 15 | 13 | 1 | 14 | 4 | 0 | 11 | 5 | 2 |
| $S_3$ | 0 | 15 | 11 | 8 | 12 | 9 | 6 | 3 | 13 | 1 | 2 | 4 | 10 | 7 | 5 | 14 |
| $S_4$ | 1 | 15 | 8 | 3 | 12 | 0 | 11 | 6 | 2 | 5 | 4 | 10 | 9 | 14 | 7 | 13 |
| $S_5$ | 15 | 5 | 2 | 11 | 4 | 10 | 9 | 12 | 0 | 3 | 14 | 8 | 13 | 6 | 7 | 1 |
| $S_6$ | 7 | 2 | 12 | 5 | 8 | 4 | 6 | 11 | 14 | 9 | 1 | 15 | 13 | 3 | 10 | 0 |
| $S_7$ | 1 | 13 | 15 | 0 | 14 | 8 | 2 | 11 | 7 | 4 | 12 | 10 | 9 | 3 | 5 | 6 |

# SERPENT. Plussid ja miinused

| plussid | miinused |
| --- | --- |
| Lihtne ülesehitus kergendab analüüsi, mille eesmärk on leida nõrgad kohad.<br><br>Efektiivselt realiseeritav riistvaraliselt ja tingimustes, kus ressursid on piiratud<br><br>On võimalik lihtsal viisil modifitseerida eesmärgiga kaitsta rünnete eest | On kõike aeglasem finalistide sees<br><br>Šifreerimine ja dešifreerimine on väga erinev ja vajab eraldi realisatsiooni<br><br>Protsesside paralleelne kasutamine on realiseeritav piirangutega |

# Algoritm RC6

- Algoritmi töötati välja aastal 1998.
- RSA Laboratories http://www.rsa.com/rsalabs/
  - Ronald Linn Rivest
  - Matt Robshaw
  - Ray Sidney
  - Yiqun Lisa Yin
- Algoritmi sisendiks on lisaks võtmele sõnad suurusega w=32 (algoritm šifreerib korraga plokki 4-st sõnast), raundide arv R=20, võtme suurus Baitides (16 Baiti  ehk 128 bitti, 24 Baiti ehk 192 bitti või 32 Baiti, ehk 256 bitti) .

Ronald L. Rivest
1947

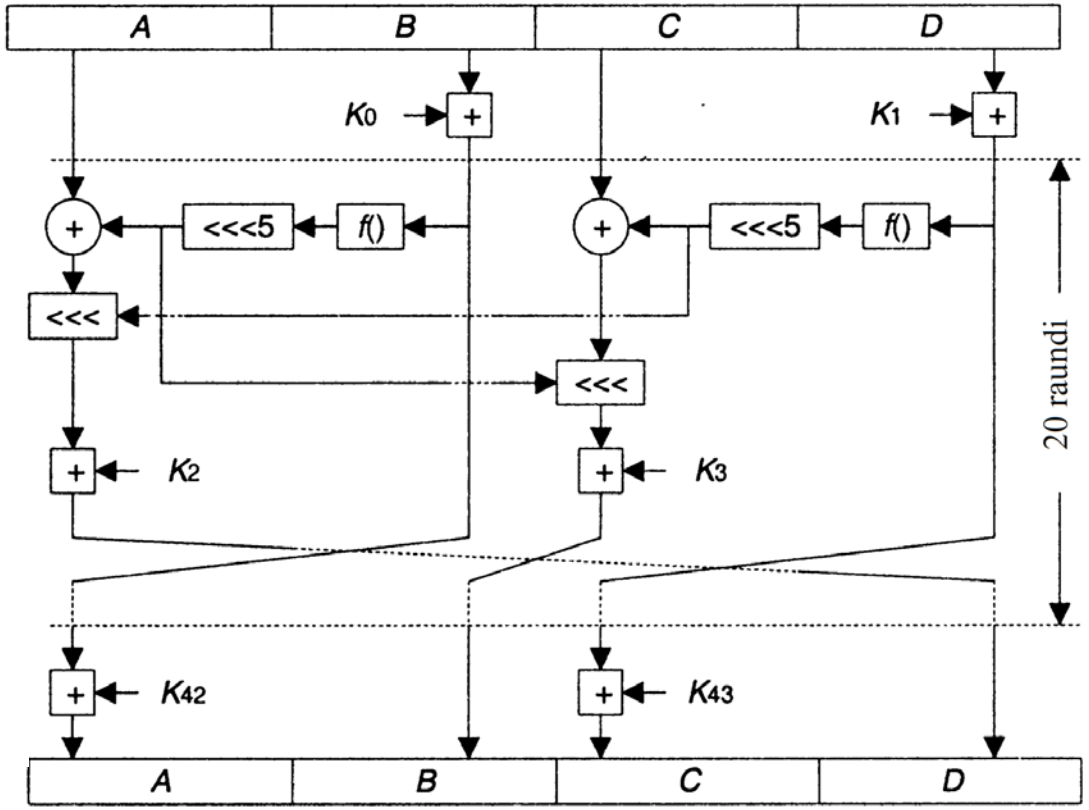
Matt Robshaw

# Algoritm RC6

$$B = B + K_0 \bmod 2^{32};$$
$$D = D + K_1 \bmod 2^{32},$$

$$A = A + K_{42} \bmod 2^{32};$$
$$C = C + K_{43} \bmod 2^{32}.$$

$$t_1 = f(B) <<< 5;$$
$$t_2 = f(D) <<< 5;$$
$$A = ((A \oplus t_1) <<< t_2) + K_{2i} \bmod 2^{32};$$
$$C = ((C \oplus t_2) <<< t_1) + K_{2i+1} \bmod 2^{32},$$




Bittide nihutamise arvu saamiseks tuleb kasutada 5 esimest bitti muutujates t2 ja t1

$$f(x) = x * (2x + 1) \bmod 2^{32}.$$

# Algoritm RC6

```
B = B + K[0]
D = D + K[1]
  for i = 1 to r do {
     t1 = (B(2B + 1)) <<< 5
     t2 = (D(2D + 1)) <<< 5
     A = ((A ^ t1) <<< t2) + K[2i]
     C = ((C ^ t2) <<< t1) + K[2i + 1]
     (A, B, C, D) = (B, C, D, A) }
A = A + K[2r + 2]
C = C + K[2r + 3]
```

# Algoritm RC6, võtmete laiendus

- Kokku on vaja 2R+4, ehk $K_0$...$K_{43}$ alamvõtmeid

$$K_0 = P_{32}; \qquad P_{32} = \text{B7E15163};$$

$$K_{i+1} = K_i + Q_{32}, \qquad Q_{32} = \text{9E3779B9}.$$

P32 ja Q32 on pseudo-juhuarvud, mis on saadud matemaatilistest konstantidest $\varepsilon$ ja $\phi$ sel moel, et nende murdosa korrutati arvuga $2^{32}$ ja seejärel ümardati lähima paaritu täisarvuni.

Tsükliliselt teostatakse järgmisi arvutusi

$$A = K_i = (K_i + A + B) <<< 3;$$

$$B = KI_j = (KI_j + A + B) <<< (A + B);$$

$$i = i + 1 \bmod 44;$$

$$j = j + 1 \bmod c,$$

i,j, A ja B on vahemuutujad ning nende väärtused on esialgu 0
Kl on algne võti, mis koosneb *c* tk 32 bitist sõnast. Teostatakse 3*c* ringi.

# Algoritm RC6, võtmete laiendus

```
K[0]=Pw
  for i=1 to 2r+3 do{
     K[i]=K[i-1]+Qw }
  A=B=i=j=0 v=3
  max{c,2r+4}
  for s=1 to v do {
     A=K[i]=(K[i]+A+B)<<<3
     B=KL[j]=(KL[j]+A+B)<<<(A+B)
     i=(i+1) mod (2r+4)
     j=(j+1) mod c }
```

# Algoritm RC6. Plussid, miinused

| Plussid | Miinused |
| --- | --- |
| Lihtne algoritmi struktuur kergendab selle analüüsi. Seejuures, antud algoritm on jätkuks algoritmile RC5, mis oli väga põhjalikult analüüsitud juba enne AES konkurssi.<br><br>Kõige kiirem algoritm finalistidest 32 bitistel platvormidel<br><br>Šifreerimine ja dešifreerimine on peaaegu identne | Šifreerimise kiirus sõltub otseselt sellest, kas platvorm toetab 32bitist korrutamist ning sellest, kas bittide ümberpaigutus toimub muutuva arvuga või mitte.<br><br>Raskesti realiseeritav riistvaraliselt piiratud tingimustes.<br><br>Rünnete eest raskesti kaitstav<br><br>Ei toeta täielikult “jooksvat” võtmete laiendust.<br><br>Protsesside paralleelne kasutamine on realiseeritav piirangutega |