Algoritmid Twofish ja Rijndael. Algoritmid ja nende matemaatilised alused.

Erika Matsak, PhD

1

Korpus

Olgu K mingi hulk, mis sisaldab vähemalt kaks elementi. Olgu K-s määratud ka kaks arvutustehet, mida tähistatakse kas või pluss- ja korrutusmärgiga (+ ja ·).

See tähendab, et "+" seab igale kahele *K* elemendile *x* ja *y* vastavusse ühe kindla *K* elemendi, mida nimetatakse *x* ja *y* summaks ning tähistatakse *x+y*-ga; samuti seab "·" neile kahele elemendile vastavusse ühe *K* elemendi, mida nimetatakse *x* ja *y* korrutiseks ning tähistatakse *x·y*-ga (või lihtsamalt *xy*-ga).

Selline hulk *K* oma kahe arvutustehtega on **korpus** ehk **kommutatiivne korpus**, kui sellel on kõik järgmised omadused:

- $x+(y+z) = (x+y)+z$ alati (+ assotsiatiivsus ehk ühenduvus)
- *K*-st leidub selline element 0, et alati kehtib $0+x = x$
- Igal elemendil *x* on *K*-s oma "vastandelement" $-x$ nii, et $-x+x = 0$
- $x+y = y+x$ alati (+ kommutatiivsus ehk vahetuvus)
- $x\cdot(y+z) = x\cdot y+x\cdot z$ alati (distributiivsus ehk jaotuvus)
- $x\cdot(y\cdot z) = (x\cdot y)\cdot z$ alati (· assotsiatiivsus ehk ühenduvus)
- *K*-st leidub teinegi eriline element 1 nii, et alati kehtib $1\cdot x = x$
- Igal elemendil *x* peale 0-i on *K*-s oma "pöördelement" x' nii, et $x'\cdot x = 1$
- $x\cdot y = y\cdot x$ alati (· kommutatiivsus ehk vahetuvus)

2

Korpus GF (2^8)

- Galois korpus ehk **Galois field,** ehk GF(q) on lõplik korpus, mille elementide arv on q.
- Iga algarvu p ja naturaalarvu n jaoks eksisteerib lõplik korpus, mille elementide arv on $q=p^n$.
- GF(2^8) on lõplik korpus, mille elementide arv on 2^8.
- Korpuse elemendid võivad olla esitatud erineval viisil.
- Bait b, mis koosneb bittidest b_7, b_6, b_5, b_4, b_3, b_2, b_1, b_0, saab olla esitatud polünoomi kujul koefitsientidega {0, 1}:
  $b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x^1 + b_0$
- Näide: Bait, mis on 16-nd süsteemis '57' (kahendkoodis 01010111) vastab polünoomile
  $x^6 + x^4 + x^2 + x + 1$

Évariste Galois (October 25, 1811 – May 31, 1832

3

Korpus GF (2^8). Liitmine

- Kahe polünoomi summa on polünoom, mille koefitsiendid on XOR liidetavate polünoomide koefitsientidest.

Näide: '57' + '83' = 'DA' :

$(x^6 + x^4 + x^2 + x + 1) + (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$

01010111
⊕10000011
= 11010100

4

Korpus GF (2^8). Korrutamine

- Polünoom $m(x) = x^8 + x^4 + x^3 + x + 1$ on selline polünoom, mis ei oma peale iseenda ning arvu 1 muid jagajaid. Sellele vastab 16-nd süsteemi arv '11B' (kümnendkoodis 283).
- Rijndael kasutab selle korrutamise juures mod arvuks.

Näide: '57' • '83' = 'C1'

$(x^6 + x^4 + x^2 + x + 1)\,(x^7 + x + 1) == x^{13} + x^{11} + x^9 + x^8 + x^7 + x^7 + x^5 + x^3 + x^2 + x + x^6 + x^4 + x^2 + x + 1 == x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$

$(x^8 + x^4 + x^3 + x + 1)\,(x^5 + x^3) + x^7 + x^6 + 1 = x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1$

Järelikult,

$x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \bmod (x^8 + x^4 + x^3 + x + 1) = x^7 + x^6 + 1$

5

Korpus GF (2^8). Liitmine koefitsientidega

$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$

$b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$

Saame liita koefitsiente XOR abil, ehk

$c(x)=(a_3 \oplus b_3)\, x^3 + (a_2 \oplus b_2)\, x^2 + (a_1 \oplus b_1)\, x^1 + (a_0 \oplus b_0)$

6

Korpus GF (2^8). Korrutamine koefitsientidega

$a(x) = a_3x^3 + a_2x^2 + a_1x + a_0$
$b(x) = b_3x^3 + b_2x^2 + b_1x + b_0$
$c(x) = a(x)\ b(x)$
$c(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0$
$c_0 = a_0 \bullet b_0$
$c_1 = a_1 \bullet b_0 \oplus a_0 \bullet b_1$
$c_2 = a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2$
$c_3 = a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3$
$c_4 = a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3$
$c_5 = a_3 \bullet b_2 \oplus a_2 \bullet b_3$
$c_6 = a_3 \bullet b_3$

Sellisel kujul ei ole esitatav 4 Baidilise vektorina.
Vajame MOD().
Kasutame selleks $M(x) = x^4 + 1$, ehk $x^j \bmod (x^4 + 1) = x^{j \bmod 4}$

7

Korpus GF (2^8). Korrutamine koefitsientidega

$d_0 = a_0 \bullet b_0 \oplus a_3 \bullet b_1 \oplus a_2 \bullet b_2 \oplus a_1 \bullet b_3$
$d_1 = a_1 \bullet b_0 \oplus a_0 \bullet b_1 \oplus a_3 \bullet b_2 \oplus a_2 \bullet b_3$
$d_2 = a_2 \bullet b_0 \oplus a_1 \bullet b_1 \oplus a_0 \bullet b_2 \oplus a_3 \bullet b_3$
$d_3 = a_3 \bullet b_0 \oplus a_2 \bullet b_1 \oplus a_1 \bullet b_2 \oplus a_0 \bullet b_3$

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

8

Algoritm Twofish

- Algoritm Twofish on töötataud välja Blowfish loojaga Bruce Schneier-i poolt koostöös kolleegidega: John Kelsey, Doug Whiting, David Wagner, Chris Hall, ning Niels Ferguson

Bruce Schneier, 1963

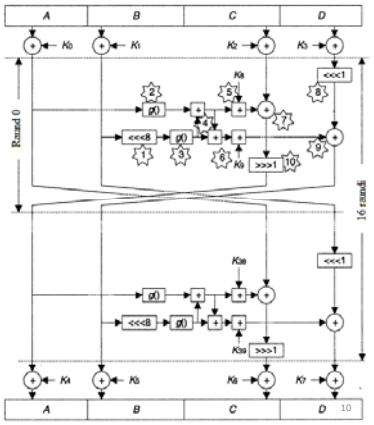
David A. Wagner, 1974

Niels Ferguson 1965

9

Algoritm Twofish

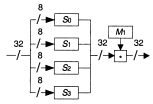
$B = B <<< 8;$
$A = g(A);$
$B = g(B);$
$A = A + B \bmod 2^{32};$
$B = A + B \bmod 2^{32};$
$A = A + K_{2r+8} \bmod 2^{32};$
$B = B + K_{2r+9} \bmod 2^{32};$
$C = C \oplus A;$
$D = D <<< 1;$
$D = D \oplus B;$
$C = C >>> 1.$



10

Algoritm Twofish, funktsioon *g*

- Funktsioon *g* töötleb 32bitist alamplokki järgmiselt:
  - Alamplokk jagatakse neljaks osaks, iga osa 8 bitti.
  - Saadud osad teisendatakse 8x8 S-boxide abil (S_0...S_3). S-boxe arvutatakse dünaamiliselt ning nad sõltuvad võtmest.
  - Eelmise sammu tulemust korrutatakse fikseeritud maatriksiga M_1. q_0..q_3 on funktsiooni *g* väljundbitid



$$\begin{pmatrix} g_0 \\ g_1 \\ g_2 \\ g_3 \end{pmatrix} = M_1 * \begin{pmatrix} s_0 \\ s_1 \\ s_2 \\ s_3 \end{pmatrix},$$

M_1:

01	EF	5B	5B
5B	EF	EF	01
EF	5B	01	EF
EF	01	EF	5B

Korrutamine toimub 2^8 korpuses, kus mod on $x^8 + x^4 + x^3 + x + 1$

11

Algoritm Twofish, funktsioon *g*

1	239	91	91	S_0	=	$1xS_0 \oplus 239xS_1 \oplus 91xS_2 \oplus 91xS_3$
91	239	239	1	S_1		$91xS_0 \oplus 239xS_1 \oplus 239xS_2 \oplus 1xS_3$
239	91	1	239	S_2		$239xS_0 \oplus 91xS_1 \oplus 1xS_2 \oplus 239xS_3$
239	1	239	91	S_3		$239xS_0 \oplus 1xS_1 \oplus 239xS_2 \oplus 91xS_3$

x

Korrutamine toimub korpuses GF(2^8)

12

Algoritm Twofish, võtmete laiendus

- Tulemuseks peame saama 40 alamvõtit, iga alamvõtme suurus on 32bitti. Seejärel peame alamvõtmete alusel arvutama S-boxe.
- Twofish kasutab võtit suvalise pikkusega kuni 256 bitti (kaasa-arvatud).
- Kõigepealt "pikendatakse" võtit 0-dega kuni standardse pikkuseni 128, 192 või 256 bitti.
- Arvutatakse k=N/64, kus N on "pikendatud" võtme suurus bittides, ehk 2,3 või 4.
- Võtit esitatakse Baitide massiivina m_0…m_{8k-1}, mille pikkus on 8·k; samuti massiivina M_0…M_{2k-1}, milles on 2·k 32bitist sõna.

$$M_i = \sum_{j=0}^{3} m_{(4i+j)} \cdot 2^{8j} \qquad i = 0, ..., 2k-1$$

13

Algoritm Twofish, võtmete laiendus

$$M_e = (M_0, M_2, ...M_{2k-2});$$
$$M_o = (M_1, M_3, ...M_{2k-1});$$
$$V = (V_{k-1}, V_{k-2}, ...V_0),$$

$$V_i = \sum_{j=0}^{3} v_{i,j} * 2^{8j};$$

$$\begin{pmatrix} v_{i,0} \\ v_{i,1} \\ v_{i,2} \\ v_{i,3} \end{pmatrix} = M_2 * \begin{pmatrix} m_{8i} \\ m_{8i+1} \\ m_{8i+2} \\ m_{8i+3} \\ m_{8i+4} \\ m_{8i+5} \\ m_{8i+6} \\ m_{8i+7} \end{pmatrix}.$$

M_2:

01	A4	55	87	5A	58	DB	9E
A4	56	82	F3	1E	C6	68	E5
02	A1	FC	C1	47	AE	3D	19
A4	55	87	5A	58	DB	9E	03

14

Algoritm Twofish, võtmete laiendus

- Alamvõtmed k_0…k_{39} arvutatakse M_0 ja M_e massiivide abil.

$$k_{2i} = A_i + B_i \bmod 2^{32};$$
$$k_{2i+1} = (A_i + 2B_i \bmod 2^{32}) <<< 9,$$

Kus i =0..19, ning A_i ja B_i on vahetulemused, mida arvutatakse järgnevalt:

$$A_i = h(2i\rho, M_e);$$
$$B_i = h((2i+1)\rho, M_o) <<< 8.$$

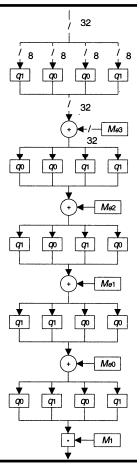
Konstant ρ:

$$\rho = 2^{24} + 2^{16} + 2^8 + 1,$$

Funktsioon *h* vajab põhjaliku kirjeldust

15

Algoritm Twofish, võtmete laiendus, funktsioon *h*



Samm 1. Sisendsõna (32-bitt) jagatakse neljaks 8-bitiseks osaks, iga osa töödeldakse teisendusega q_0 või q_1. Tulemus ühendatakse 32 bitiseks sõnaks ning liidetakse kokku massiivi M_e kolmanda sõnaga. Antud samm jääb tegemata, kui k<4, ehk k=2 või k=3.

Samm2. Eelmise sammu tulemus või sisendsõna jagatakse neljaks 8 bitiseks osaks. Analoogselt eelmisele sammule töödeldakse q_0 või q_1 (selle sammu jaoks on uued q_0 ja q_1). Seejärel tulemus liidetakse massiivi M_e teise sõnaga. Antud samm jääb tegemata kui k=2.

Samm 3 ja 4. Analoogselt sammudele 1 ja 2.

Samm 5. Peale viimast teisendust q_0 , q_1 abil, teostatakse järgmine teisendus:

$$\begin{pmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \end{pmatrix} = M_1 * \begin{pmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \end{pmatrix};$$

$$H = \sum_{i=0}^{3} h_i * 2^{8i},$$

y_i on tulemused q teisendustest

Algoritm Twofish, võtmete laiendus

Teisendus q.
Olgu x –teisenduse sisendsõna, y -väljundsõna

$$a_0 = \lfloor x/16 \rfloor;$$
$$b_0 = x \bmod 16;$$
$$a_1 = a_0 \oplus b_0;$$
$$b_1 = a_0 \oplus (b_0 >>>_4 1) \oplus 8a_0 \bmod 16;$$
$$a_2 = t_0(a_1);$$
$$b_2 = t_1(b_1);$$
$$a_3 = a_2 \oplus b_2;$$
$$b_3 = a_2 \oplus (b_2 >>>_4 1) \oplus 8a_2 \bmod 16;$$
$$a_4 = t_2(a_3);$$
$$b_4 = t_3(b_3);$$
$$y = 16b_4 + a_4,$$

Näks ehk nibble (inglise keeles nibble, nybble) on infoühik, mis sisaldab neli bitti.

17

Algoritm Twofish, võtmete laiendus

t_i arvutus q_0 jaoks

t_0	8	1	7	D	6	F	3	2	0	B	5	9	E	C	A	4
t_1	E	C	B	8	1	2	3	5	F	4	A	6	7	0	9	D
t_2	B	A	5	E	6	D	9	0	C	8	F	3	2	4	7	1
t_3	D	7	F	4	1	2	6	E	9	B	3	0	8	5	C	A

t_i arvutus q_1 jaoks

t_0	2	8	B	D	F	7	6	E	3	1	9	4	0	A	C	5
t_1	1	E	2	B	4	C	3	7	6	D	A	5	F	9	0	8
t_2	4	C	7	5	1	6	9	A	0	E	D	8	2	B	3	F
t_3	B	9	5	1	C	3	D	E	6	4	7	F	2	0	8	A

18

Algoritm Twofish, S-Boxid

- S arvutatakse samuti võtmest,
- Võib öelda, et reaalselt funktsiooni *g* asemel kasutatakse funktsiooni *h*, ehk *g(X) = h(X; S)*
- S_i osad arvutatakse samamoodi kui q_i.

19

Algoritm Twofish. Plussid ja miinused

Plussid	Miinused
Efektiivselt realiseeritav riistvaraliselt, samuti siis kui ressursid on piiratud Šifreerimine ja dešifreerimine samalaadse realisatsiooniga Parim finalistidest võtmete laiendamise "jooksva" režiimi tasemel On võimalik optimeerida konkreetsetele eesmärkidele	Keeruline ülesehitus raskendab analüüsi Keeruline ja aeglane võtme laiendus Raske kaitsta rünnete eest Protsesside paralleelne kasutamine on realiseeritav piirangutega

20

AES= Rijndael

- Algoritm on esitatud Joan Daemen'i (Proton World International) ja Vincent Rijmen'i (Katholieke Universiteit Leuven) poolt.
- Kasutab andmete plokki kui kahemõõtmelist Baitide 4x4 massiivi (vajadusel ka 4x6 ja 4x8)
- Baite vaadeldakse kui lõpliku korpuse GF (2^8) elemente .
- Kõik operatsioonid teostatakse massiivi siseselt ning samuti tulpade ja ridade siseselt.
- Kõik teisendused šifris omavad ranget matemaatilist alust.

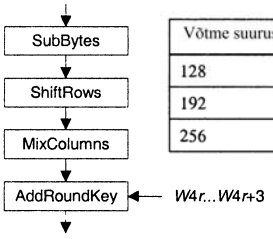
Joan Daemen, 1965

Vincent Rijmen, 1970

21

Algoritm Rijndael

Igas raundis teostatakse järgmiseid teisendusi:



Võtme suurus bittides	Raundide arv
128	10
192	12
256	14

AES-128
AES-192
AES-256

Raundide arv saab olla 10 kuni 14, sõltuvalt võtme pikkusest

22

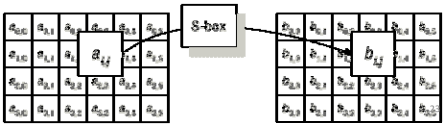
Algoritm Rijndael, SubBytes

- SubBytes on tabeli alusel teostatav teisendus, kusjuures selle asemel on võimalik teostada ka järgmiseid ekvivalentseid samme:
  - Multiplikatiivse pöördväärtuse leidmine korpuses GF (2^8) (nulli jaoks on siin pöördarvuks null ise)
  - Väljundarv *b'* arvutatakse järgmise valemi abil

$$b_i' = b_i \oplus b_{(i+4) mod 8} \oplus b_{(i+5) mod 8} \oplus b_{(i+6) mod 8} \oplus b_{(i+7) mod 8} \oplus c_i$$

Kus b_i–on i kohal seisev bitt muutuja b jaoks, ning c_i on konstandi *c* i kohal seisev bitt: arv 63, kahendkoodis ehk 01100011

$0 \le i < 8$



Algoritm Rijndael, SubBytes

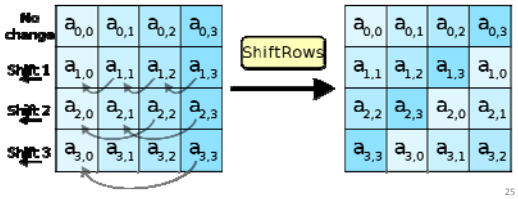
$$b_i' = b_i \oplus b_{(i+4) mod 8} \oplus b_{(i+5) mod 8} \oplus b_{(i+6) mod 8} \oplus b_{(i+7) mod 8} \oplus c_i$$

$$\begin{Vmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \\ b_4' \\ b_5' \\ b_6' \\ b_7' \end{Vmatrix} = \begin{Vmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{Vmatrix} * \begin{Vmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{Vmatrix} + \begin{Vmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{Vmatrix}$$

24

Algoritm Rijndael, ShiftRows

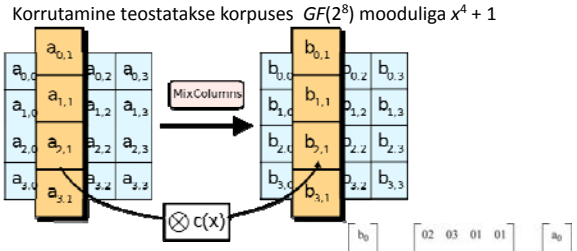
- Teisendus rea tasemel. Igal real nihutatakse vastav arv Baite. Nihutatud Baitide arv sõltub rea numbrist.



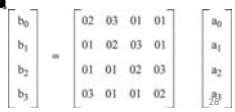
25

Algoritm Rijndael, MixColumns

Iga tulba neli Baiti segatakse lineaarteisenduse abil.
Korrutamine teostatakse korpuses $GF(2^8)$ mooduliga $x^4 + 1$

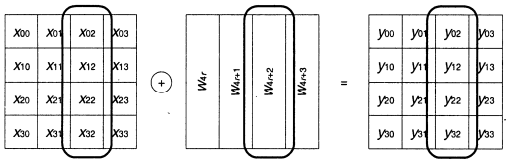


$c(x) = 3x^3 + x^2 + x + 2$



28

Algoritm Rijndael, AddRoundKey



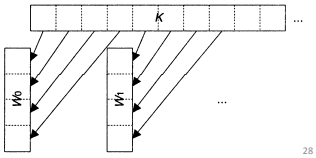
Andmete *i* tulba Baite pannakse kokku operatsiooniga XOR alamavõtme sõna Baitidega (iga sõna 4Baiti)
Siin *i* on raundi number

Enne esimest raundi teostatakse eelnevalt AddRoundKey, kasutades $W_0 \ldots W_3$
Viimasel raundil ei kasutata protseduuri MixColumns.

27

Algoritm Rijndael, KeyExpansion

- Selle protseduuri abil saadakse esialgsest võtmest kõikide raundide alamvõtmed, ehk 4*(R+1) sõna (iga sõna 4 Baiti).
- Võtme laiendus koosneb kahest etapist. Esimesel initsialiseeritakse massiiv W_i. Kõigepealt kirjutatakse N_k arv sõnu (N_k on algse võtme pikkus sõnades, ehk 4, 6 või 8) massiivi W_i, (kus i=0… (N_k -1))



28

Algoritm Rijndael, KeyExpansion

- Seejärel jätkatakse W_i arvutamisega, nüüd on *i* vahemikus N_k…(4*(R+1)-1)

1. Initsialiseeritakse ajutine muutuja T: $T=W_{i-1}$
2. See T modifitseeritakse järgnevalt:
   - Kui *i* on N_k kordne, siis $T = SubWord(RotWord(T)) \oplus RC_{i/Nk}$
   - Kui N_k =8 ja (i mod N_k)=4, siis $T = SubWord(T)$
   - Muudel juhtumitel modifitseerimist ei toimu

RC_i kujutab endast sõnu, kus kõik Baidid peale esimest on nullid, ning esimene Bait on 2^{n-1} mod 256

SubWord teostab iga Baidiga teisendust tabeli abil, mis oli kirjeldatud protseduuris SubBytes
RotWord nihutab (keerab, ehk > >) sisendsõna 1 Baidi võrra (8 bitt) vasakule

29

Algoritm Rijndael, KeyExpansion

```
KeyExpansion(byte key[4*Nk], word w[Nb*(Nr+1)], Nk)
begin
    word temp
    i = 0;

    while ( i < Nk)
        w[i] = word(key[4*i], key[4*i+1], key[4*i+2], key[4*i+3])
        i = i+1
    end while

    i = Nk

    while ( i < Nb * (Nr+1))
        temp = w[i-1]
        if (i mod Nk = 0)
            temp = SubWord(RotWord(temp)) xor Rcon[i/Nk]
        else if (Nk > 6 and i mod Nk = 4)
            temp = SubWord(temp)
        end if
        w[i] = w[i-Nk] xor temp
        i = i + 1
    end while
end
```

30

Algoritm Rijndael, Plussid ja MIinused

Plussid	Miinused
Efektiivselt realiseeritav riistvaraliselt, samuti siis, kui ressursid on piiratud Šifreerimine ja dešifreerimine samalaadse realisatsiooniga On võimalik protsesside paralleelne kasutamine On võimalik võtme laiendamine "jooksvalt"	Väiksema raundide arvuga algoritmi on võimalik rünnata (Square ründega) On arvamusi, et raundide arv ei ole piisav (pakutakse 18 ja 24). !!! Aastal 2006 on välja töötatud rünne AES-192 vastu, milles kasutatakse 10 raundi. Selleks on vaja 2^{125} avateksti ja $2^{146,7}$ operatsioone. http://eprint.iacr.org/2006/451.pdf

31