AES - Rijndael
CRIPTOSSISTEMA DE CHAVE SECRETA

AES-Advanced Encryption Standard - J. Daemen, V. Rijmen, 2000

Routo Terada
www.ime.usp.br/~rt
Depto. C. da Computação – USP

Operações básicas sobre bytes de 8 bits
(Notação na base 16: 'xy'=x*16+y, sendo x de 4 bits, y de 4 bits)

- 'A'=1010 na base 2, 'B'=1011, 'C'=1100, 'D'=1101, 'E'=1110, 'F'=1111
- soma é ou-exclusivo (xor)
  '45'+'78'='3D' pois '4' xor '7'='3', '5' xor '8'='D'
  subtrair também é ou-exclusivo pois -1=1 mod 2: '45'-'78'='3D'
- multiplicação:
  cada byte b_7. b_6... b_0 corresponde a um polinômio de grau 7,
  $b_7*x^7+b_6*x^6+..+b_0$.
  Dados dois polinômios f(x) e g(x) correspondentes a dois bytes,
  multiplica-se f(x) por g(x)
  e depois extrai-se o módulo (resto de divisão por)
  $m(x)=x^8+x^4+x^3+x+1$='11B', que é primo

veja exemplo '45'* '0A'= '94' na página seguinte

AES

$m(x) = x^8 + x^4 + x^3 + x + 1$
$f(x) = x^6 + x^2 + 1$ corresp. '45' na base 16
$g(x) = x^3 + x$ corresp. '0A'
$f(x) * g(x) = x^9 + x^7 + x^5 + x$
$h(x) = (x^9 + x^7 + x^5 + x) \bmod (m(x)) = x^7 + x^4 + x^2$ corresp. '94'

$$'45'*'0A' = (x^9 + x^7 + x^5 + x) \bmod m(x) = x^7 + x^4 + x^2 = '94'$$

Sendo m(x) primo, cada byte b(x) possui um inverso $b^{-1}(x) \bmod m(x)$ tal que $b(x)b^{-1}(x) = 1 \bmod m(x)$

AES

Operações básicas sobre bytes de 8 bits
(caso particular de multiplicação por x)

Multiplicação de byte $b(x)=b_7*x^7+b_6*x^6+..+b_0$ por x='02' na base 16:

$m(x)=x^8+x^4+x^3+x+1$='11B'=1 0001 1011

$b(x)*x \bmod m(x) = (b_7*x^8+b_6*x^7+..+b_0*x+0) \bmod m(x)$:
- se b_7=0, o resultado é $(b_6.b_5...b_0)0$, ou seja, inserir um 0 à direita, pois o polinômio acima já é de grau <8; por ex. 0100 0101*x mod m(x)=1000 1010 i.e. '45'*'02'='8A'
- se b_7=1, o resultado é $b(x)*x - m(x) = [b(x)*x]$ xor m(x) pois subtrair m(x) equivale a somar m(x) que por sua vez equivale a xor com m(x) por ex. 1100 0101*x mod m(x)=
  1 1000 1010 xor 1 0001 1011=
  0 1001 0001 i.e. 'A5'*'02'='91'

Notação: xtime(b) é multiplicação de b(x) por x; por ex., xtime('45')='8A'

AES

Operações básicas sobre **vetores de 4 bytes** (32 bits)
(Notação na base 16: 'abcd'=a*16^3+b*16^2+c*16+d)

- soma de 2 vetores: somar cada byte separadamente; como visto, soma de bytes é xor e portanto soma de vetores é xor também
- multiplicação de 2 vetores:

  $A(x)=a_3*x^3+a_2*x^2+a_1*x+a_0$

  $B(x)=b_3*x^3+b_2*x^2+b_1*x+b_0$

  o produto $D(x)=A(x)*B(x)$ é um polinômio de grau 6

  $D(x)$ mod $M(x)$ onde $M(x)=x^4+1$ é o resultado
- $M(x)$ escolhido pelos autores é tal que:

  $$x^j \bmod M(x) = x^{j \bmod 4}$$
- Como conseqüência disso, o produto $d_3.d_2.d_1.d_0$ do vetor $a_3.a_2.a_1.a_0$ por $b_3.b_2.b_1.b_0$ é dado por

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

RT AES 5

AES

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

RT AES 6

AES

Operações básicas sobre **vetores de 4 bytes** (32 bits)
Caso particular de multiplicação por x

- Multiplicação de b(x) por x:
  $(b_3*x^3+b_2*x^2+b_1*x+b_0)*x=b_3*x^4+b_2*x^3+b_1*x^2+b_0*x= c(x)$
- c(x) mod M(x)= x^4+1 é dado por

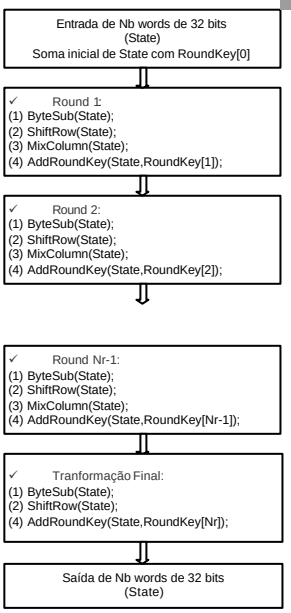
$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 00 & 00 & 00 & 01 \\ 01 & 00 & 00 & 00 \\ 00 & 01 & 00 & 00 \\ 00 & 00 & 01 & 00 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

que equivale a deslocamento circular p/ esquerda dentro do vetor

RT AES 7

AES

Esquema geral do AES-Rijndael



Nr: número de rounds

RoundKey: subchave
de comprimento Nb words
derivada da chave

SEM (3) MixColumn()

RT AES 8

AES

Comprimento de cada entrada/bloco (State)			
bits	bytes	índice do byte	Nb= número de words de 32 bits
128	16	0..15	4
192	24	0..23	6
256	32	0..31	8

Mesma tabela para Nk=comprimento da chave.

RT AES 9

AES

Número de rounds Nr (Nk=Número de words de 32-bits na chave) (Nb=Número de words de 32-bits na entrada)			
	Nb=4	Nb=6	Nb=8
Nk=4	10	12	14
Nk=6	12	12	14
Nk=8	14	14	14

RT AES 10

AES

Funções em cada round (iteração)

O bloco de entrada de comprimento Nb é chamado State;
o valor após cada round também.

(1) ByteSub(State);

(2) ShiftRow(State);

(3) MixColumn(State);

(4) AddRoundKey(State,RoundKey);

Último round: só (1), (2) e (4): SEM MixColumn()

AES

Um bloco (State) de comprimento Nb=4 words de 32-bits
em forma matricial (Nb=4 colunas, caso de 128 bits)

vetor
4 bytes

32 bits

	índice 0	1	2	3
índice 0	$a_{0,0}$	$a_{0,1}$	$a_{0,2}$	$a_{0,3}$
1	$a_{1,0}$	$a_{1,1}$	$a_{1,2}$	$a_{1,3}$
2	$a_{2,0}$	$a_{2,1}$	$a_{2,2}$	$a_{2,3}$
3	$a_{3,0}$	$a_{3,1}$	$a_{3,2}$	$a_{3,3}$

AES

Uma chave de comprimento Nk=4 words de 32-bits
em forma matricial (Nk=4 colunas, total de 128 bits)

vetor
4 bytes

32 bits

	índice 0	1	2	3
índice 0	$k_{0,0}$	$k_{0,1}$	$k_{0,2}$	$k_{0,3}$
1	$k_{1,0}$	$k_{1,1}$	$k_{1,2}$	$k_{1,3}$
2	$k_{2,0}$	$k_{2,1}$	$k_{2,2}$	$k_{2,3}$
3	$k_{3,0}$	$k_{3,1}$	$k_{3,2}$	$k_{3,3}$

RT AES 13

AES

ByteSub(State) opera sobre cada byte separadamente

(1) Inverte cada byte (mod m(x)) conforme explicado anteriormente, sendo '00' o inverso de si mesmo.
(2) A seguir, cada bit x_j de cada byte é submetido à multiplicação matricial seguinte:

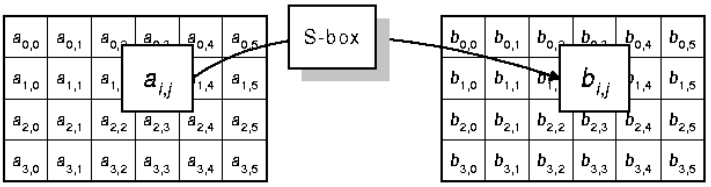
$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Trata-se de uma transformação não-linear

RT AES 14

AES

ByteSub(State) opera sobre cada byte separadamente



O inverso de ByteSub(State) consiste em aplicar o inverso do Passo (2) e depois o inverso do Passo (1) explicado anteriormente.

RT AES 15

AES

ShiftRow(State) opera sobre cada LINHA separadamente

(1) A linha 0 não é alterada;
(2) A linha 1 é deslocada circularm/ p/ esquerda de C1 bytes
(3) A linha 2 é deslocada circularm/ p/ esquerda de C2 bytes
(4) A linha 3 é deslocada circularm/ p/ esquerda de C3 bytes

Nb	C1	C2	C3
4	1	2	3
6	1	2	3
8	1	3	4

Nb=4

a	b	c	d
e	f	g	h
k	l	m	n
q	r	s	t

→

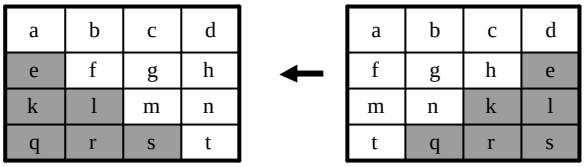
a	b	c	d
f	g	h	e
m	n	k	l
t	q	r	s

RT AES 16

AES

O inverso de ShiftRow(State) também opera sobre cada LINHA separadamente

(1) A linha 0 não é alterada;
(2) A linha 1 é deslocada circularm/ p/ esquerda de Nb-C1 bytes
(3) A linha 2 é deslocada circularm/ p/ esquerda de Nb-C2 bytes
(4) A linha 3 é deslocada circularm/ p/ esquerda de Nb-C3 bytes



Nb=4

RT AES 17

AES

MixColumn (State) opera sobre cada COLUNA separadamente, ou seja, sobre cada **vetor de 4 bytes** (32 bits)

(1) Vetor $A(x)=a_3*x^3+a_2*x^2+a_1*x+a_0$ é multiplicado pelo vetor fixo $c(x)=‘03’*x^3+‘01’*x^2+‘01’*x+‘02’$
(obs.: c(x) e M(x) são co-primos → c(x) possui inversa modM(x).)
(2) Aplicar mod x^4+1 (x^4+1 é M(x))

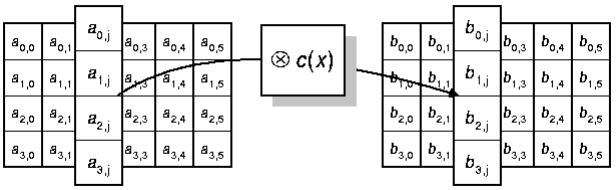
Matricialmente, o resultado (b_3,b_2,b_1,b_0) é dado por:

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

RT AES 18

AES

MixColumn (State) opera sobre cada COLUNA separadamente, ou seja, sobre cada **vetor de 4 bytes** (32 bits)



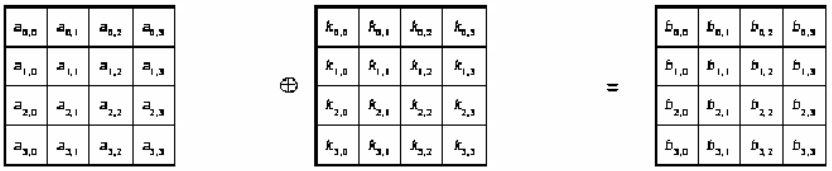
O inverso de MixColumn (State) também opera sobre cada COLUNA **de 4 bytes** (32 bits).

(1) Vetor $A(x)=a_3*x^3+a_2*x^2+a_1*x+a_0$ é multiplicado pelo vetor fixo $c^{-1}(x)=$'0B'$*x^3+$'0D'$*x^2+$'09'$*x+$'0E'
(obs.: $c^{-1}(x)$ é a inversa de c(x) mod M(x): $c(x)*c^{-1}(x)=1 \bmod M(x)$.)
(2) Aplicar mod x^4+1 (x^4+1 é chamado M(x))

AES

AddRoundKey(State,RoundKey) opera sobre State de comprimento Nb words de 32-bits:

(1) RoundKey é a subchave de comprimento Nb words de 32-bits, derivada da chave principal (veremos mais tarde)
(2) Esta operação consiste em efetuar ou-exclusivo entre State e Roundkey

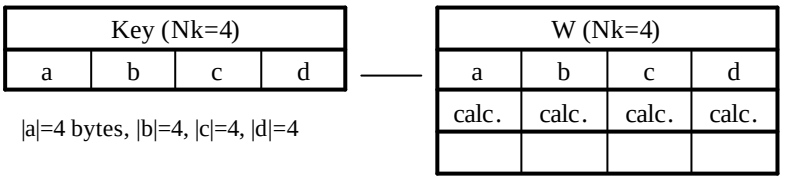


AddRoundKey é a inversa de si mesma.

AES

Geração de subchaves (key schedule)
(1) Chave principal Key de comprim/ Nk words de 32-bits
(2) Subchave RoundKey de comprim/ Nb words de 32-bits é derivada de Key, para cada round=1...Nr

(1) Key é expandido para um ExpandedKey W[] com Nb*(Nr+1) words de 32 bits; por ex., Nb=4 (128 bits), Nr=10, Nb*(Nr+1)=44 (1408 bits)
(2) Os primeiros Nk words de W[] contêm os words de Key
(3) Os elementos seguintes de W[] são calculados de elementos de W[] com índices menores.
(4) Este cálculo depende do valor de Nk



RT AES 21

AES

Geração de subchaves (key schedule):
fase de expansão de Key para gerar W[]

Operações básicas:
(1) SubByte(V) aplica S-box do Rijndael sobre cada byte do vetor V de 4 bytes, conforme visto na operação ByteSub(State)
(2) RotByte(V) aplica deslocamento circular p/ esquerda de um byte sobre o vetor V de 4 bytes
(a,b,c,d) → (b,c,d,a)
(3) sendo i um índice inteiro múltiplo de Nk (comprim.de Key), ^Rcon[] é a operação de ou-exclusivo com uma constante de 4 bytes Rcon[] (Round constant) definida a seguir:
Rcon[i]=(RC[i], '00', '00', '00') onde
RC[1]= '01' = polinômio 1
RC[i] = '02'*RC[i-1] = x*RC[i-1]=$x^{(i-1)}$

RT AES 22

AES

Para Nk<7, a geração de W é como segue:

```
KeyExpansion(byte Key[4*Nk] word W[Nb*(Nr+1)])
{
      for(i = 0; i < Nk; i++)
            W[i] = (Key[4*i],Key[4*i+1],Key[4*i+2],Key[4*i+3]);

      for(i = Nk; i < Nb * (Nr + 1); i++)
      {
            temp = W[i - 1];
            if (i % Nk == 0)
                  temp = SubByte(RotByte(temp)) ^ Rcon[i / Nk];
            W[i] = W[i - Nk] ^ temp;
      }
}
```

RT AES 23

AES

Geração de subchaves (key schedule):
fase de geração de RoundKey, após a fase de geração de W[]

Subchave RoundKey de comprim/ Nb words de 32-bits

Para i=1...Nr,
RoundKey[i] é igual aos words W[Nb*i],W[Nb*i+1],...W[Nb*(i+1)]

Para Nb=4 e Nk=4, tem-se a ilustração a seguir:

RoundKey[0] é igual aos words W[0],W[1],...W[3]
RoundKey[1] é igual aos words W[4],W[5],...W[7]
....

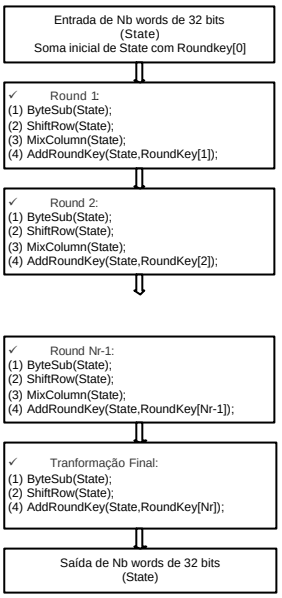
W[0]	W[1]	W[2]	W[3]	W[4]	W[5]	W[6]	W[7]	W[8]	W[9]	W[10]	W[11]	W[12]	W[13]	W[14]	.

RoundKey[0]	RoundKey[1]	…

RT AES 24

AES

Esquema geral



RT AES 25

AES

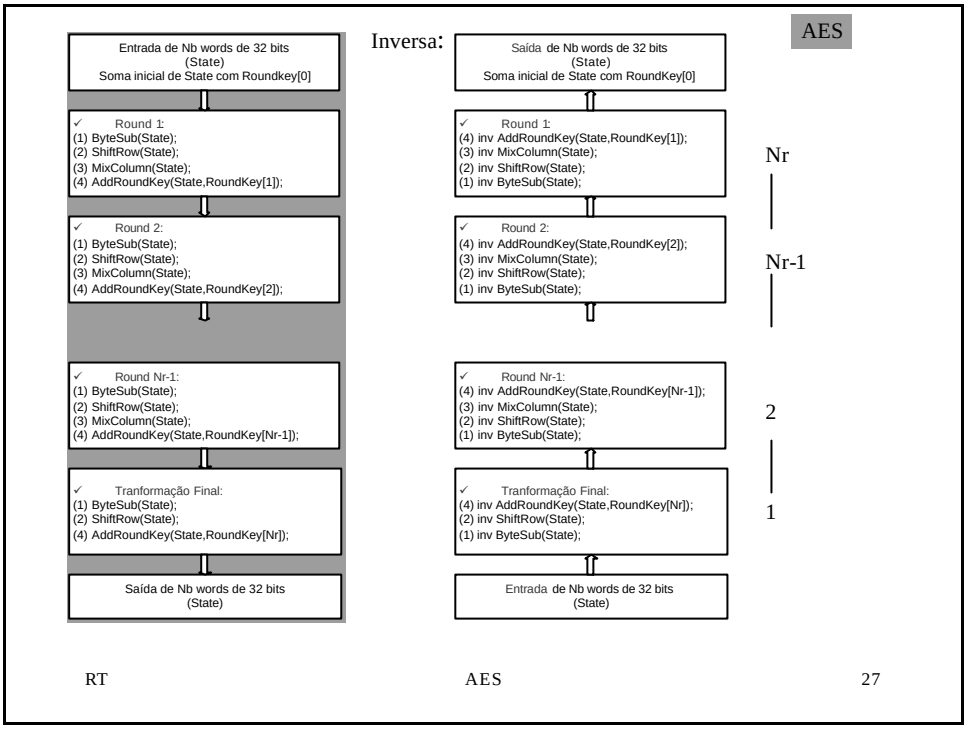
AES/Rijndael em linguagem pseudo C:

```
Rijndael(State,CipherKey)
{
KeyExpansion(CipherKey,ExpandedKey) ;
AddRoundKey(State,ExpandedKey);
For( i=1 ; i<Nr ; i++ ) Round(State,ExpandedKey + Nb*i) ;
FinalRound(State,ExpandedKey + Nb*Nr);
}
```

Se a ExpandedKey for calculada previamente:

```
Rijndael(State,ExpandedKey)
{
AddRoundKey(State,ExpandedKey);
For( i=1 ; i<Nr ; i++ ) Round(State,ExpandedKey + Nb*i) ;
FinalRound(State,ExpandedKey + Nb*Nr);
}
```

RT AES 26


AES
Entrada de Nb words de 32 bits
(State)
Soma inicial de State com Roundkey[0]
✓ Round 1:
(1) ByteSub(State);
(2) ShiftRow(State);
(3) MixColumn(State);
(4) AddRoundKey(State,RoundKey[1]);
✓ Round 2:
(1) ByteSub(State);
(2) ShiftRow(State);
(3) MixColumn(State);
(4) AddRoundKey(State,RoundKey[2]);
✓ Round Nr-1:
(1) ByteSub(State);
(2) ShiftRow(State);
(3) MixColumn(State);
(4) AddRoundKey(State,RoundKey[Nr-1]);
✓ Tranformação Final:
(1) ByteSub(State);
(2) ShiftRow(State);
(4) AddRoundKey(State,RoundKey[Nr]);
Saída de Nb words de 32 bits
(State)
Inversa:
Saída de Nb words de 32 bits
(State)
Soma inicial de State com RoundKey[0]
✓ Round 1:
(4) inv AddRoundKey(State,RoundKey[1]);
(3) inv MixColumn(State);
(2) inv ShiftRow(State);
(1) inv ByteSub(State);
✓ Round 2:
(4) inv AddRoundKey(State,RoundKey[2]);
(3) inv MixColumn(State);
(2) inv ShiftRow(State);
(1) inv ByteSub(State);
✓ Round Nr-1:
(4) inv AddRoundKey(State,RoundKey[Nr-1]);
(3) inv MixColumn(State);
(2) inv ShiftRow(State);
(1) inv ByteSub(State);
✓ Tranformação Final:
(4) inv AddRoundKey(State,RoundKey[Nr]);
(2) inv ShiftRow(State);
(1) inv ByteSub(State);
Entrada de Nb words de 32 bits
(State)
Nr
Nr-1
2
1
RT
AES
27