Advanced Encryption Standard (AES)

History

- AES is the result of an open competition organized by NIST, US Department of Commerce.
- NIST issued call for a standard cipher in 1997
  - 15 candidates (out of 21) accepted in Jun 98
- Five candidates are sort listed.
- NIST continued to study all the available information and analyses about candidate algorithms and selected one of the algorithm, Rijndael algorithm, to propose for the AES.
- AES resists well all known cryptographic attacks and has already now achieved a high level of acceptance.

AES Overview

- AES specifies Rijndael algorithm, a symmetric block cipher that can process data blocks of 128 bits, using cipher keys with lengths of 128, 192 and 256 bits.
- It may be referred to as AES-128, AES-192 and AES-256.
- AES operates on a 4×4 array of bytes, termed the **state** (versions of Rijndael with a larger block size have additional columns in the state).
- For Encryption, each round except last round of AES consists of four stages.
  - SubBytes
  - ShiftRows
  - MixColumns
  - AddRoundKey
- The final round omits the MixColumns stage.

AES Overview (contd)

- A number of AES parameters depend on the key length.

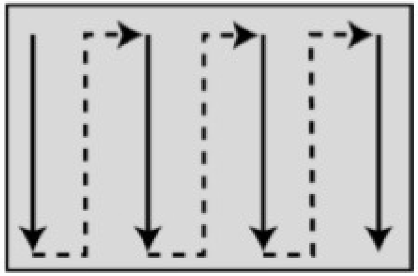
Key size (words/bytes/bits)	4/16/128	6/24/192	8/32/256
Plaintext block size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Number of rounds	10	12	14
Round key size (words/bytes/bits)	4/16/128	4/16/128	4/16/128
Expanded key size (words/bytes)	44/176	52/208	60/240

b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}

Block

$$s_{i \bmod 4,\, i/4} \longleftarrow \text{block}_i$$

$$\text{State} \quad \begin{bmatrix} s_{0,0} = b_0 & s_{0,1} = b_4 & s_{0,2} = b_8 & s_{0,3} = b_{12} \\ s_{1,0} = b_1 & s_{1,1} = b_5 & s_{1,2} = b_9 & s_{1,3} = b_{13} \\ s_{2,0} = b_2 & s_{2,1} = b_6 & s_{2,2} = b_{10} & s_{2,3} = b_{14} \\ s_{3,0} = b_3 & s_{3,1} = b_7 & s_{3,2} = b_{11} & s_{3,3} = b_{15} \end{bmatrix}$$



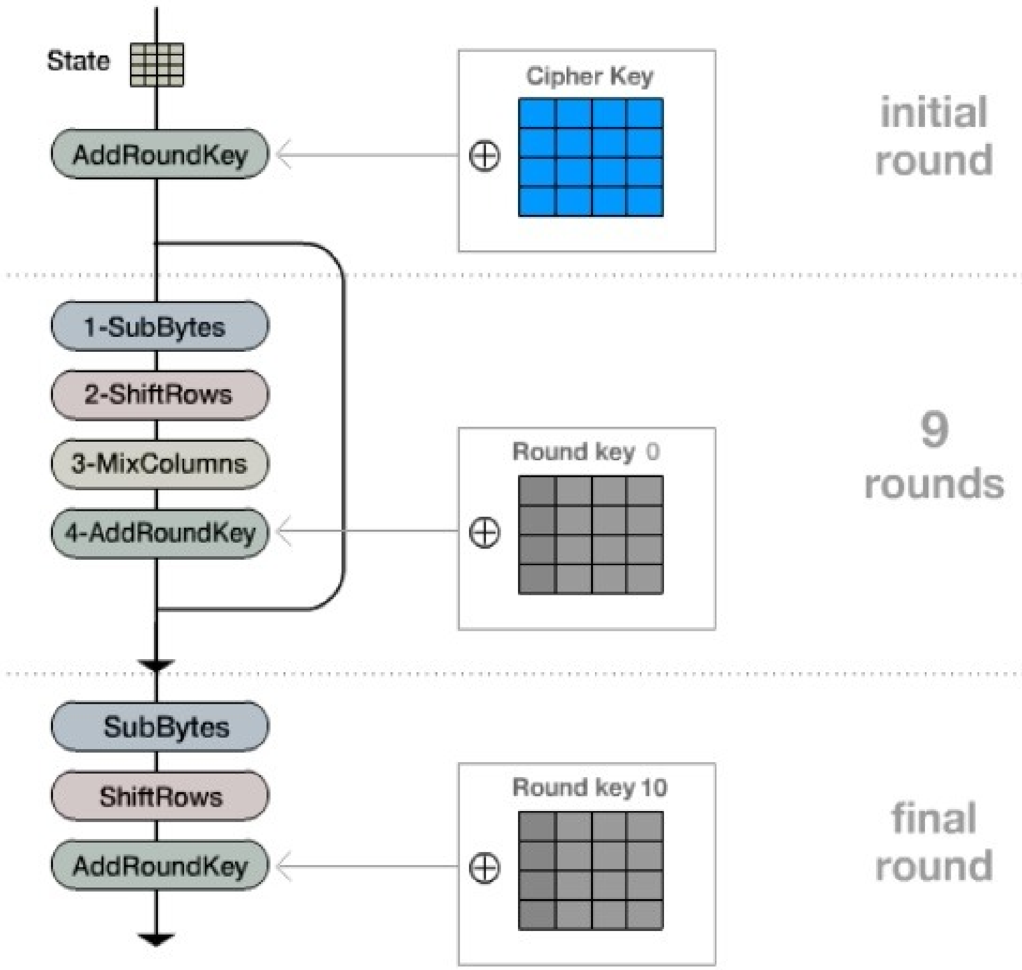
Insertion and extraction flow

$$\text{block}_{i+4j} \longleftarrow s_{i,j}$$

Block

b_0	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	b_9	b_{10}	b_{11}	b_{12}	b_{13}	b_{14}	b_{15}

AES Structure



Algorithm Parameters, Symbols and Functions used in Description of AES

- N_k Number of 32-bit words comprising the Cipher key.
  - For AES, $N_k = 4, 6$ or 8 (i.e. key length 128, 192 or 256 bits)
- N_r Number of rounds, which is a function of N_k and N_b.
  - For AES $N_r = 10$, 12 or 14.
- Rcon [ ] The round constant word array.
- RotWord( )
  - Function used in the Key Expansion routine that takes a four- byte word and perform a cyclic permutation.
- ShiftRows( )
  - Transformation in the Cipher that processes the State by cyclically shifting the last three rows of the State by different offsets.
- SubBytes( )
  - Transformation in the Cipher that processes the State using a nonlinear byte substitution table (S–box) that operates on each of the State bytes independently.
- Finite field multiplication.

Algorithm Parameters, Symbols and Functions used in Description of AES

- N_k Number of 32-bit words comprising the Cipher key.
  - For AES, N_k = 4, 6 or 8 (i.e. key length 128, 192 or 256 bits)
- N_r Number of rounds, which is a function of N_k and N_b.
  - For AES N_r = 10, 12 or 14.
- Rcon [ ] The round constant word array.
- RotWord( )
  - Function used in the Key Expansion routine that takes a four- byte word and perform a cyclic permutation.
- ShiftRows( )
  - Transformation in the Cipher that processes the State by cyclically shifting the last three rows of the State by different offsets.
- SubBytes( )
  - Transformation in the Cipher that processes the State using a nonlinear byte substitution table (S–box) that operates on each of the State bytes independently.
- Finite field multiplication.

State

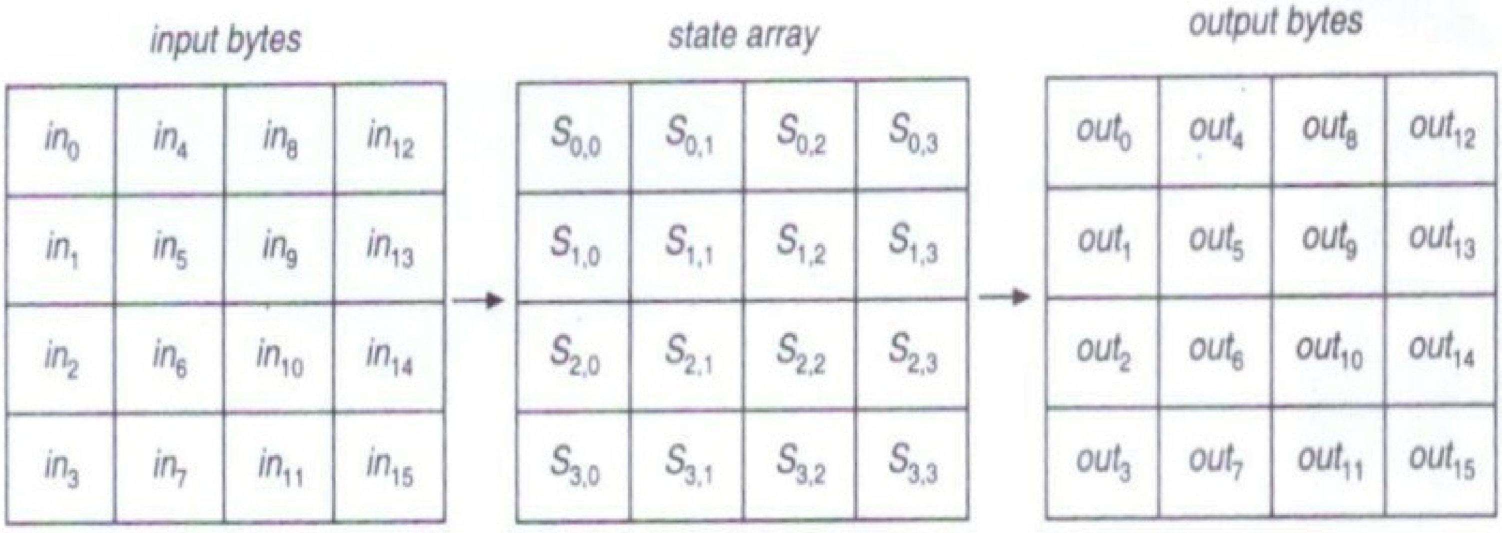
- Definition:
  - AES algorithm's operations are performed on a two-dimensional array of bytes called the ***State***.

- The State consists of four rows of bytes, each containing N_b bytes, where N_b is the block length divided by 32.

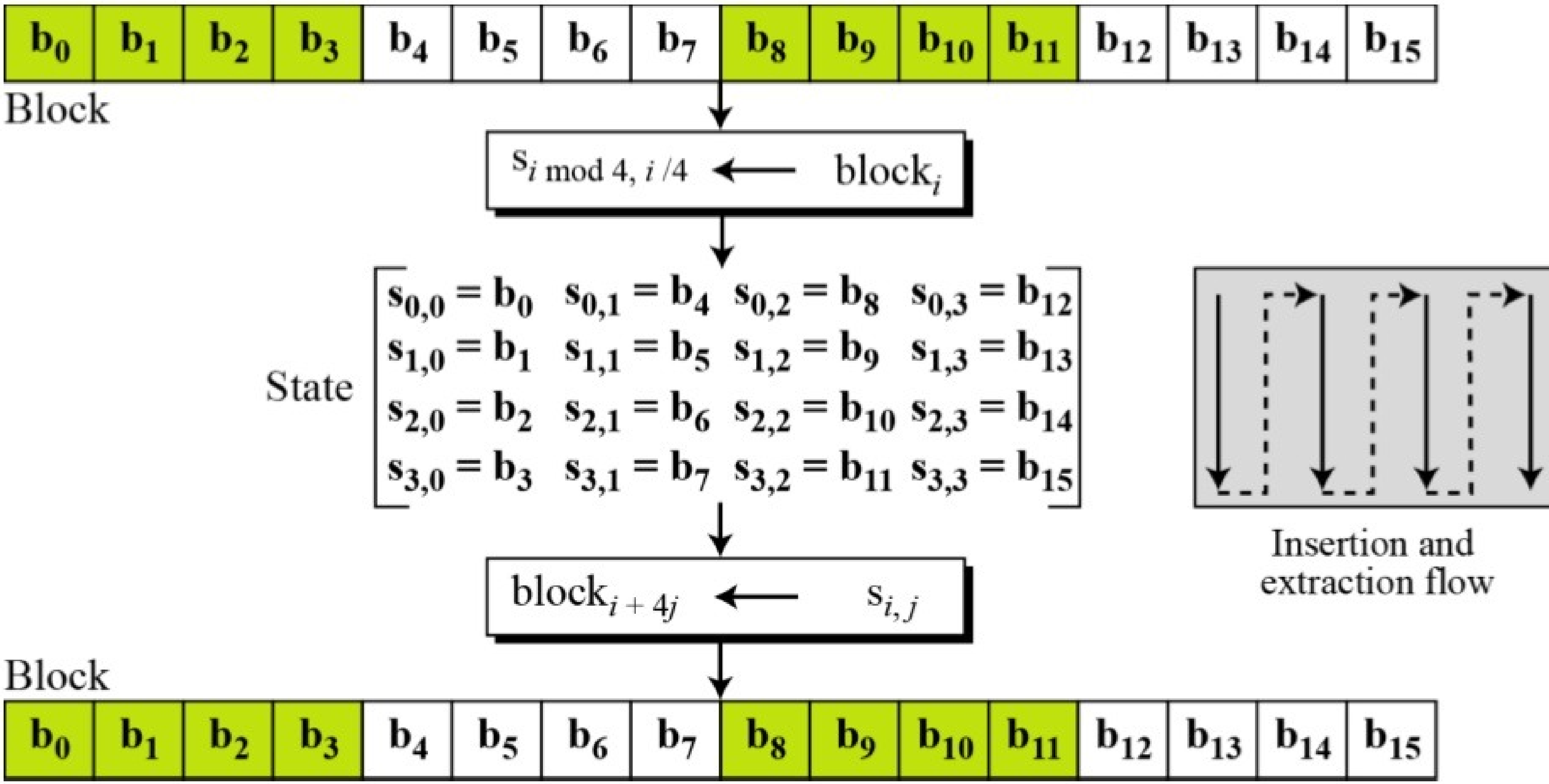
- State array denoted by symbol S.
  - Each individual byte has two indices.
    - Row number r, $0 \le r < 4$
    - Column number c, $0 \le c < N_b$

State (contd)

- The input, array of bytes in_0, in_1, … in_{15}, is copied into the State array as shown in figure.



- The cipher operations are then conducted on this State array, after which its final value is copied to the output – the array of bytes out_0,out_1,…,out_{15}.


b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 b10 b11 b12 b13 b14 b15
Block
$s_{i \bmod 4,\, i/4} \leftarrow \text{block}_i$
State
$s_{0,0} = b_0$ $s_{0,1} = b_4$ $s_{0,2} = b_8$ $s_{0,3} = b_{12}$
$s_{1,0} = b_1$ $s_{1,1} = b_5$ $s_{1,2} = b_9$ $s_{1,3} = b_{13}$
$s_{2,0} = b_2$ $s_{2,1} = b_6$ $s_{2,2} = b_{10}$ $s_{2,3} = b_{14}$
$s_{3,0} = b_3$ $s_{3,1} = b_7$ $s_{3,2} = b_{11}$ $s_{3,3} = b_{15}$
$\text{block}_{i+4j} \leftarrow s_{i,j}$
Block
b0 b1 b2 b3 b4 b5 b6 b7 b8 b9 b10 b11 b12 b13 b14 b15
Insertion and extraction flow

GF(2^8)

- Byte $b_7b_6b_5b_4b_3b_2b_1b_0$ will have the representation as
  $b(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$
- Therefore, 01010111 would have the representation as 01010111
  $x^6 + x^4 + x^2 + x + 1$

Addition in Finite field

- Addition in a finite field achieved by adding the coefficients for the corresponding powers in the polynomials for the two elements
- Addition performed by EX-OR operation
  - denoted as <F, $\oplus$> modulo 2
- Alternatively, addition of finite field elements done by modulo 2 addition of the corresponding bits in the byte.

Example:

- $(x^6 + x^4 + x^2 + x + 1) \oplus (x^7 + x + 1) = x^7 + x^6 + x^4 + x^2$ (polynomial notation)
- {01010111} $\oplus$ {10000011} = {11010100} (binary notation)
- {57} $\oplus$ {83} = {d4} (hexadecimal notation)

Multiplication

- denoted by <F{0}, • > or < F{0},
- multiplication in $GF(2^8)$ corresponds to
  - multiplication of polynomials modulo an irreducible polynomial of degree 8
  - irreducible polynomial is the one whose divisors are one and itself only
  - for AES, the irreducible polynomial is $m(x) = x^8 + x^4 + x^3 + x + 1$ (i.e. {01}{1B})

Example:

{57} • {83} = {c1}.

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

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

$$= x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1 \qquad \text{------(1)}$$

Multiplication (contd)

$(x^{13} + x^{11} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + 1)$ modulo $(x^8 + x^4 + x^3 + x + 1)$
$= x7 + x6 + 1 = \{c1\}$

Sol^n : $x^8 + x^4 + x^3 + x + 1$ is a m(x) means irreducible polynomial.

Multiply this m(x) with x^5 (because highest power in eq. (1) is 13.)

$(x^8 + x^4 + x^3 + x + 1)(x^5) = x^{13} + x^9 + x^8 + x^6 + x^5$ ----(2)

- Now addition of eq. (1) and (2),

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

  $= x^{11} + x^4 + x^3 + 1$ ----- (3)
- Multiply m(x) with x^3 ((because highest power in eq. (3) is 11.)

  $(x^8 + x^4 + x^3 + x + 1)(x^3) = x^{11} + x^7 + x^6 + x^4 + x^3$ ----(4)
- Now addition of eq. (3) and (4),

  $(x^{11} + x^7 + x^6 + x^4 + x^3) \oplus (x^{11} + x^4 + x^3 + 1) = x^7 + x^6 + 1 = \{11000001\}$

  $= \{c1\}$

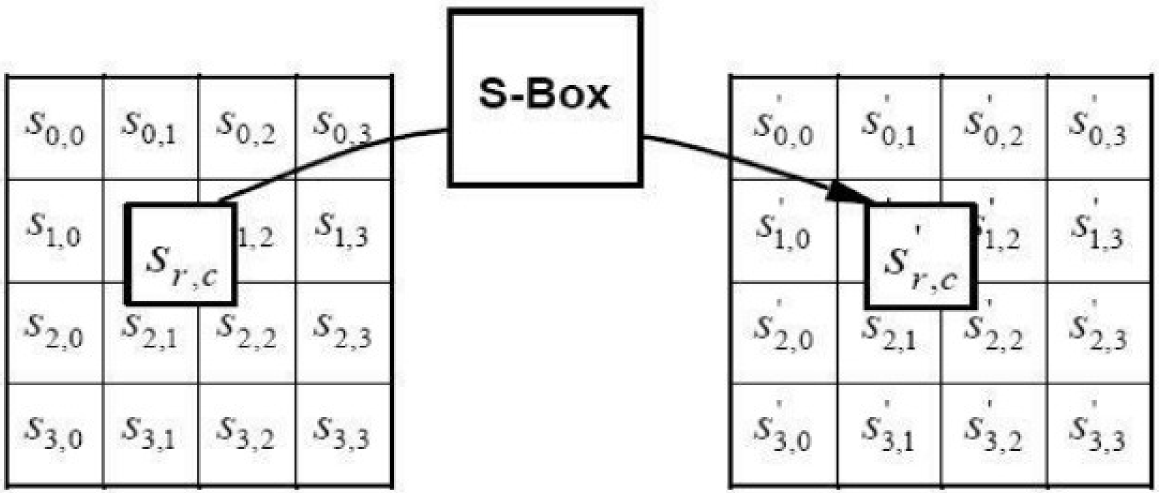
Pseudo code for AES

- N_b = 4 for block size 128 bits

```
Cipher(byte in[4*Nb],byte out[4*Nb],word w[Nb*(Nr+1)])
begin
     byte state[4,Nb]
     state = in
     AddRoundKey(state, w[0, Nb-1])
     for round = 1 step 1 to Nr-1
          SubBytes(state)
          ShiftRows(state)
          MixColumns(state)
          AddRoundKey(state,w[round*Nb,(round+1)*Nb-1])
     end for
     SubBytes(state)
     ShiftRows(state)
     AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])
     out = state
end
```

SubBytes( ) Transformation

- It is a non-linear byte substitution that operates independently on each byte of the State using a substitution table (S- box).
- Each byte of state is replaced by byte indexed by row (left 4-bits) & column (right 4-bits)
  - eg. byte {95} is replaced by byte in row 9 column 5
  - which has value {2A}



SubBytes( ) Transformation (contd)

- AES S-box

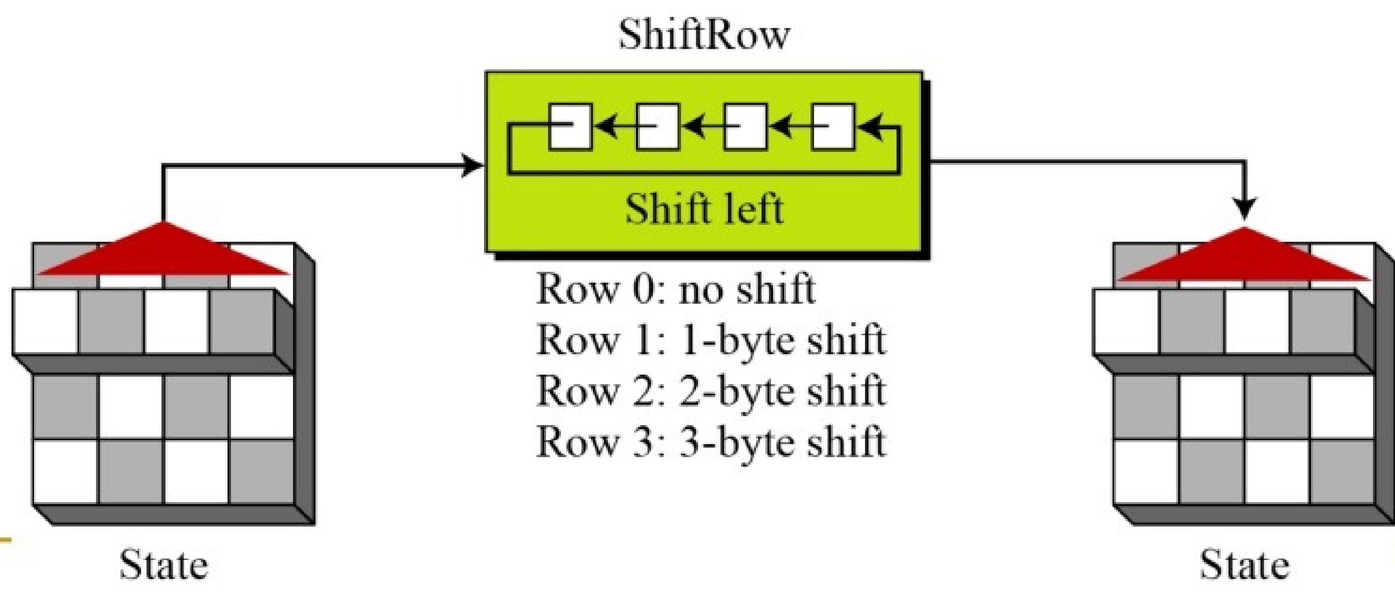
		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

- Inverse S-box used in the **InvSubBytes()** transformation

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
	0	52	09	6a	d5	30	36	a5	38	bf	40	a3	9e	81	f3	d7	fb
	1	7c	e3	39	82	9b	2f	ff	87	34	8e	43	44	c4	de	e9	cb
	2	54	7b	94	32	a6	c2	23	3d	ee	4c	95	0b	42	fa	c3	4e
	3	08	2e	a1	66	28	d9	24	b2	76	5b	a2	49	6d	8b	d1	25
	4	72	f8	f6	64	86	68	98	16	d4	a4	5c	cc	5d	65	b6	92
	5	6c	70	48	50	fd	ed	b9	da	5e	15	46	57	a7	8d	9d	84
	6	90	d8	ab	00	8c	bc	d3	0a	f7	e4	58	05	b8	b3	45	06
	7	d0	2c	1e	8f	ca	3f	0f	02	c1	af	bd	03	01	13	8a	6b
x	8	3a	91	11	41	4f	67	dc	ea	97	f2	cf	ce	f0	b4	e6	73
	9	96	ac	74	22	e7	ad	35	85	e2	f9	37	e8	1c	75	df	6e
	a	47	f1	1a	71	1d	29	c5	89	6f	b7	62	0e	aa	18	be	1b
	b	fc	56	3e	4b	c6	d2	79	20	9a	db	c0	fe	78	cd	5a	f4
	c	1f	dd	a8	33	88	07	c7	31	b1	12	10	59	27	80	ec	5f
	d	60	51	7f	a9	19	b5	4a	0d	2d	e5	7a	9f	93	c9	9c	ef
	e	a0	e0	3b	4d	ae	2a	f5	b0	c8	eb	bb	3c	83	53	99	61
	f	17	2b	04	7e	ba	77	d6	26	e1	69	14	63	55	21	0c	7d

ShiftRows( ) Transformation

- The bytes in the last three rows of the State are cyclically shifted over different numbers of bytes.
  - 1st row is unchanged
  - 2nd row does 1 byte circular shift to left
  - 3rd row does 2 byte circular shift to left
  - 4th row does 3 byte circular shift to left
- Decrypt inverts using shifts to right



MixColumns( ) Transformation

- The transformation operates on the State column-by-column, treating each column as a four-term polynomial.
- The MixColumns stage is a substitution that makes use of arithmetic over GF(2^8).
- Constant matrices used by MixColumns and InvMixColumns

$$\underset{C}{\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}} \xleftrightarrow{\text{Inverse}} \underset{C^{-1}}{\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}}$$

MixColumns( ) Transformation (contd)

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \qquad \text{for } 0 \le c < \boldsymbol{Nb}.$$

As a result of this multiplication, the four bytes in a column are replaced by the following:

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$

$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c}).$$

MixColumns( ) Transformation (contd)

$$\begin{bmatrix} s'_{0,c} \\ s'_{1,c} \\ s'_{2,c} \\ s'_{3,c} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,c} \\ s_{1,c} \\ s_{2,c} \\ s_{3,c} \end{bmatrix} \quad \text{for } 0 \le c < \boldsymbol{Nb}.$$

As a result of this multiplication, the four bytes in a column are replaced by the following:

$$s'_{0,c} = (\{02\} \bullet s_{0,c}) \oplus (\{03\} \bullet s_{1,c}) \oplus s_{2,c} \oplus s_{3,c}$$

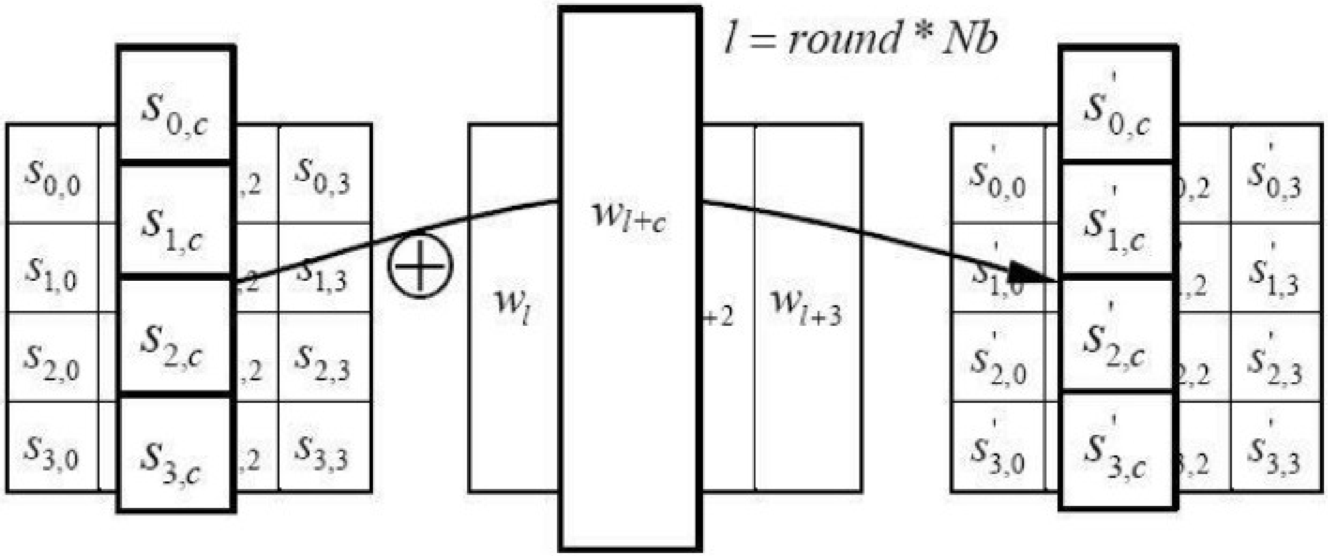
$$s'_{1,c} = s_{0,c} \oplus (\{02\} \bullet s_{1,c}) \oplus (\{03\} \bullet s_{2,c}) \oplus s_{3,c}$$

$$s'_{2,c} = s_{0,c} \oplus s_{1,c} \oplus (\{02\} \bullet s_{2,c}) \oplus (\{03\} \bullet s_{3,c})$$

$$s'_{3,c} = (\{03\} \bullet s_{0,c}) \oplus s_{1,c} \oplus s_{2,c} \oplus (\{02\} \bullet s_{3,c}).$$

AddRoundKey( ) Transformation

- A Round key is added to the State by a simple bitwise XOR operation.
- Each Round Key consists of N_b words from the key schedule.
- Inverse for decryption is identical.
  - Since XOR own inverse, with reversed keys



AES Key Expansion

- Algorithm takes the Cipher Key K and performs a Key Expansion routine to generate a key schedule.
- The Key Expansion generates a total of N_b (N_r +1) words
  - Algorithm requires an initial set of N_b words
  - Each of the N_r rounds requires N_b words of key data.
- proceeds as per
  - subword()
    - input – 4-byte word and S-box
    - output – 4 byte word after substitution
  - RotWord()
    - input – 4-byte word [a0a1a2a3]
    - output - 4-byte rotated word [a1a2a3a0]

Pseudo code for Key Expansion

```
KeyExpansion(byte key[4 * Nk], word w[Nb * (Nr + 1)], Nk)
begin
    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))
        word temp = w[i - 1]
        if (i mod Nk = 0)
            temp = SubWord(RotWord(temp)) xor Rcon[i / Nk]
        else if (Nk = 8 and i mod Nk = 4)
            temp = SubWord(temp)
        end if
    w[i] = w[i - Nk] xor temp
    i = i + 1
    end while
end
```

AES Cipher Example

- Input = 32 43 f6 a8 88 5a 30 8d 31 31 98 a2 e0 37 07 34
- Cipher Key = 2b 7e 15 16 28 ae d2 a6 ab f7 15 88 09 cf 4f 3c

AES Key Expansion Example

2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

. . .

How to calculate Rcon?

Rcon table

01	02	04	08	10	20	40	80	1b	36
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00

Rcon

AES Key Expansion Example (contd)

Cipher Key				Round key 1				Round key 2				Round key 3					Round key 10			
2b	28	ab	09	a0	88	23	2a	f2	7a	23	73	3d	47	1e	6d		d0	c9	e1	b6
7e	ae	f7	cf	fa	54	a3	6c	c2	96	a3	59	80	16	23	7a	…	14	ee	3f	63
15	d2	15	4f	fe	2c	39	76	95	b9	39	f6	47	fe	7e	88		f9	25	0c	0c
16	a6	88	3c	17	b1	39	05	f2	43	39	7f	7d	3e	44	3b		a8	89	c8	a6

AES Key Expansion Example (contd)

Cipher Key				Round key 1				Round key 2				Round key 3					Round key 10			
2b	28	ab	09	a0	88	23	2a	f2	7a	23	73	3d	47	1e	6d		d0	c9	e1	b6
7e	ae	f7	cf	fa	54	a3	6c	c2	96	a3	59	80	16	23	7a	...	14	ee	3f	63
15	d2	15	4f	fe	2c	39	76	95	b9	39	f6	47	fe	7e	88		f9	25	0c	0c
16	a6	88	3c	17	b1	39	05	f2	43	39	7f	7d	3e	44	3b		a8	89	c8	a6

AES Key Expansion Example (contd)

Cipher Key				Round key 1				Round key 2				Round key 3					Round key 10			
2b	28	ab	09	a0	88	23	2a	f2	7a	23	73	3d	47	1e	6d		d0	c9	e1	b6
7e	ae	f7	cf	fa	54	a3	6c	c2	96	a3	59	80	16	23	7a	…	14	ee	3f	63
15	d2	15	4f	fe	2c	39	76	95	b9	39	f6	47	fe	7e	88		f9	25	0c	0c
16	a6	88	3c	17	b1	39	05	f2	43	39	7f	7d	3e	44	3b		a8	89	c8	a6

AES Key Expansion Example (contd)

Cipher Key

2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

Round key 1

a0	88	23	2a
fa	54	a3	6c
fe	2c	39	76
17	b1	39	05

Round key 2

f2	7a	23	73
c2	96	a3	59
95	b9	39	f6
f2	43	39	7f

Round key 3

3d	47	1e	6d
80	16	23	7a
47	fe	7e	88
7d	3e	44	3b

…

Round key 10

d0	c9	e1	b6
14	ee	3f	63
f9	25	0c	0c
a8	89	c8	a6

AES Key Expansion Example (contd)

2b	28	ab	09	a0	88	23	2a	f2	7a	23	73	3d	47	1e	6d		d0	c9	e1	b6
7e	ae	f7	cf	fa	54	a3	6c	c2	96	a3	59	80	16	23	7a	...	14	ee	3f	63
15	d2	15	4f	fe	2c	39	76	95	b9	39	f6	47	fe	7e	88		f9	25	0c	0c
16	a6	88	3c	17	b1	39	05	f2	43	39	7f	7d	3e	44	3b		a8	89	c8	a6
Cipher Key				Round key 1				Round key 2				Round key 3					Round key 10			

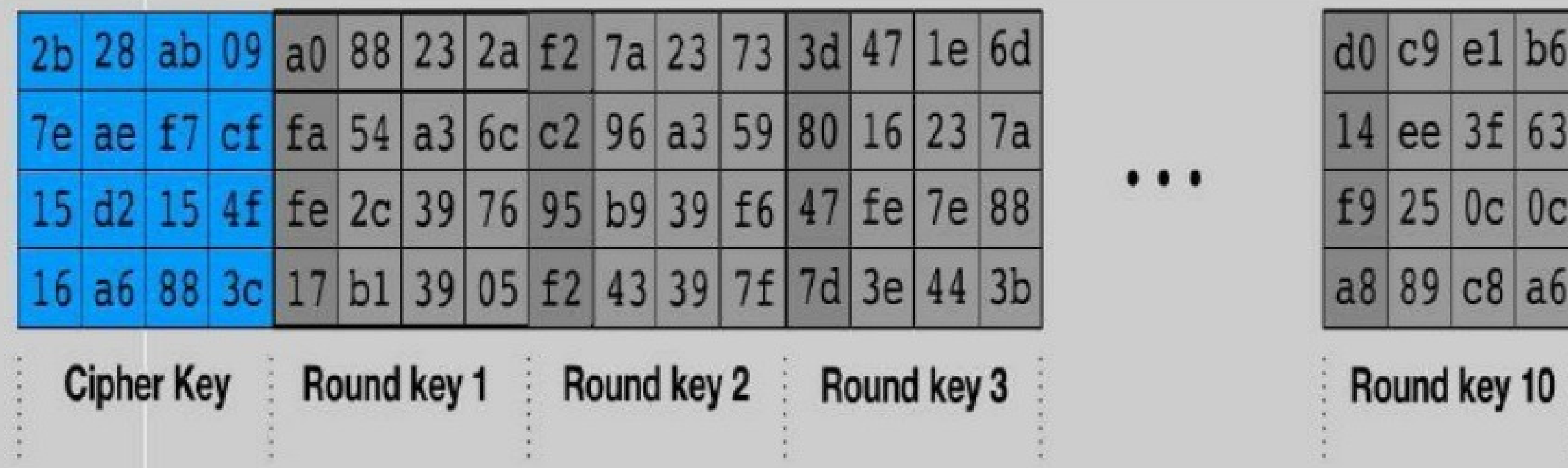
AES Key Expansion Example (contd)

Cipher Key				Round key 1				Round key 2				Round key 3					Round key 10			
2b	28	ab	09	a0	88	23	2a	f2	7a	23	73	3d	47	1e	6d		d0	c9	e1	b6
7e	ae	f7	cf	fa	54	a3	6c	c2	96	a3	59	80	16	23	7a	...	14	ee	3f	63
15	d2	15	4f	fe	2c	39	76	95	b9	39	f6	47	fe	7e	88		f9	25	0c	0c
16	a6	88	3c	17	b1	39	05	f2	43	39	7f	7d	3e	44	3b		a8	89	c8	a6

AES Key Expansion Example (contd)

Cipher Key				Round key 1				Round key 2				Round key 3					Round key 10			
2b	28	ab	09	a0	88	23	2a	f2	7a	23	73	3d	47	1e	6d		d0	c9	e1	b6
7e	ae	f7	cf	fa	54	a3	6c	c2	96	a3	59	80	16	23	7a	• • •	14	ee	3f	63
15	d2	15	4f	fe	2c	39	76	95	b9	39	f6	47	fe	7e	88		f9	25	0c	0c
16	a6	88	3c	17	b1	39	05	f2	43	39	7f	7d	3e	44	3b		a8	89	c8	a6

AES Key Expansion Example (contd)



AES Key Expansion Example (contd)

Cipher Key				Round key 1				Round key 2				Round key 3				…	Round key 10			
2b	28	ab	09	a0	88	23	2a	f2	7a	23	73	3d	47	1e	6d		d0	c9	e1	b6
7e	ae	f7	cf	fa	54	a3	6c	c2	96	a3	59	80	16	23	7a	• • •	14	ee	3f	63
15	d2	15	4f	fe	2c	39	76	95	b9	39	f6	47	fe	7e	88		f9	25	0c	0c
16	a6	88	3c	17	b1	39	05	f2	43	39	7f	7d	3e	44	3b		a8	89	c8	a6

Initial Round for Encryption

Input Data Block

32	88	31	e0
43	5a	31	37
f6	30	98	07
a8	8d	a2	34

$\oplus$

Cipher Key

2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

=

Input for 1st Round

19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

1stRound - SubBytes transformation

19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

hex		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

S-BOX

1^{st} Round - SubBytes transformation (contd)

Result after SubBytes stage

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

hex		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

S-BOX

1st Round - SubBytes transformation (contd)

Result after SubBytes stage

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

hex		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

S-BOX

1st Round - SubBytes transformation (contd)

Result after SubBytes stage

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

hex		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

S-BOX

1st Round - SubBytes transformation (contd)

Result after SubBytes stage

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

hex		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
	0	63	7c	77	7b	f2	6b	6f	c5	30	01	67	2b	fe	d7	ab	76
	1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
	2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
	3	04	c7	23	c3	18	96	05	9a	07	12	80	e2	eb	27	b2	75
	4	09	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
	5	53	d1	00	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
	6	d0	ef	aa	fb	43	4d	33	85	45	f9	02	7f	50	3c	9f	a8
	7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
x	8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
	9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
	a	e0	32	3a	0a	49	06	24	5c	c2	d3	ac	62	91	95	e4	79
	b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	08
	c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
	d	70	3e	b5	66	48	03	f6	0e	61	35	57	b9	86	c1	1d	9e
	e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
	f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

S-BOX

1^{st} Round – ShiftRows transformation

d4	e0	b8	1e
27	bf	b4	41
11	98	5d	52
ae	f1	e5	30

rotate over 1 byte

1st Round – ShiftRows transformation (contd)

Result after ShiftRows stage

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

rotate over 3 bytes

1st Round – ShiftRows transformation (contd)

Result after ShiftRows stage

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

rotate over 3 bytes

1st Round – ShiftRows transformation (contd)

Result after ShiftRows stage

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

rotate over 3 bytes

1st Round – ShiftRows transformation (contd)

Result after ShiftRows stage

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

rotate over 3 bytes

1st Round – ShiftRows transformation (contd)

Result after ShiftRows stage

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

rotate over 3 bytes

1st Round – ShiftRows transformation (contd)

Result after ShiftRows stage

d4	e0	b8	1e
bf	b4	41	27
5d	52	11	98
30	ae	f1	e5

rotate over 3 bytes

1^{st} Round – MixColumns transformation

Result after MixColumns transforamtion

04	e0	48	28
66	cb	f8	06
81	19	d3	26
e5	9a	7a	4c

1^{st} Round – MixColumns transformation

Result after MixColumns transforamtion

04	e0	48	28
66	cb	f8	06
81	19	d3	26
e5	9a	7a	4c

1^{st} Round – MixColumns transformation

Result after MixColumns transforamtion

04	e0	48	28
66	cb	f8	06
81	19	d3	26
e5	9a	7a	4c

1^{st} Round – AddRound Key

Input for this stage

04	e0	48	28
66	cb	f8	06
81	19	d3	26
e5	9a	7a	4c

1st Round – AddRound Key (contd)

04	e0	48	28
66	cb	f8	06
81	19	d3	26
e5	9a	7a	4c

a0	88	23	2a
fa	54	a3	6c
fe	2c	39	76
17	b1	39	05

Round key

1st Round – AddRound Key Example (contd)

a4	68	6b	02
9c	9f	5b	6a
7f	35	ea	50
f2	2b	43	49

Round key

1st Round – AddRound Key Example (contd)

a4	68	6b	02
9c	9f	5b	6a
7f	35	ea	50
f2	2b	43	49

Round key

Results for remaining rounds

After SubBytes

Round 7				Round 8				Round 9				Round 10			
f7	27	9b	54	be	d4	0a	da	87	f2	4d	97	e9	cb	3d	af
ab	83	43	b5	83	3b	e1	64	ec	6e	4c	90	09	31	32	2e
31	a9	40	3d	2c	86	d4	f2	4a	c3	46	e7	89	07	7d	2c
f0	ff	d3	3f	c8	c0	4d	fe	8c	d8	95	a6	72	5f	94	b5

After ShiftRows

Round 7				Round 8				Round 9				Round 10			
f7	27	9b	54	be	d4	0a	da	87	f2	4d	97	e9	cb	3d	af
83	43	b5	ab	3b	e1	64	83	6e	4c	90	ec	31	32	2e	09
40	3d	31	a9	d4	f2	2c	86	46	e7	4a	c3	7d	2c	89	07
3f	f0	ff	d3	fe	c8	c0	4d	a6	8c	d8	95	b5	72	5f	94

After MixColumns

Round 7				Round 8				Round 9			
14	46	27	34	00	b1	54	fa	47	40	a3	4c
15	16	46	2a	51	c8	76	1b	37	d4	70	9f
b5	15	56	d8	2f	89	6d	99	94	e4	3a	42
bf	ec	d7	43	d1	ff	cd	ea	ed	a5	a6	bc

⊕

Round Key

Round 7				Round 8				Round 9				Round 10			
4e	5f	84	4e	ea	b5	31	7f	ac	19	28	57	d0	c9	e1	b6
54	5f	a6	a6	d2	8d	2b	8d	77	fa	d1	5c	14	ee	3f	63
f7	c9	4f	dc	73	ba	f5	29	66	dc	29	00	f9	25	0c	0c
0e	f3	b2	4f	21	d2	60	2f	f3	21	41	6e	a8	89	c8	a6

=

After AddRoundKey

Round 7				Round 8				Round 9				Round 10 (Ciphertext)			
5a	19	a3	7a	ea	04	65	85	eb	59	8b	1b	39	02	dc	19
41	49	e0	8c	83	45	5d	96	40	2e	a1	c3	25	dc	11	6a
42	dc	19	04	5c	33	98	b0	f2	38	13	42	84	09	85	0b
b1	1f	65	0c	f0	2d	ad	c5	1e	84	e7	d2	1d	fb	97	32

Results for remaining rounds

After SubBytes

Round 7				Round 8				Round 9				Round 10			
f7	27	9b	54	be	d4	0a	da	87	f2	4d	97	e9	cb	3d	af
ab	83	43	b5	83	3b	e1	64	ec	6e	4c	90	09	31	32	2e
31	a9	40	3d	2c	86	d4	f2	4a	c3	46	e7	89	07	7d	2c
f0	ff	d3	3f	c8	c0	4d	fe	8c	d8	95	a6	72	5f	94	b5

After ShiftRows

Round 7				Round 8				Round 9				Round 10			
f7	27	9b	54	be	d4	0a	da	87	f2	4d	97	e9	cb	3d	af
83	43	b5	ab	3b	e1	64	83	6e	4c	90	ec	31	32	2e	09
40	3d	31	a9	d4	f2	2c	86	46	e7	4a	c3	7d	2c	89	07
3f	f0	ff	d3	fe	c8	c0	4d	a6	8c	d8	95	b5	72	5f	94

After MixColumns

Round 7				Round 8				Round 9			
14	46	27	34	00	b1	54	fa	47	40	a3	4c
15	16	46	2a	51	c8	76	1b	37	d4	70	9f
b5	15	56	d8	2f	89	6d	99	94	e4	3a	42
bf	ec	d7	43	d1	ff	cd	ea	ed	a5	a6	bc

$\oplus$

Round Key

Round 7				Round 8				Round 9				Round 10			
4e	5f	84	4e	ea	b5	31	7f	ac	19	28	57	d0	c9	e1	b6
54	5f	a6	a6	d2	8d	2b	8d	77	fa	d1	5c	14	ee	3f	63
f7	c9	4f	dc	73	ba	f5	29	66	dc	29	00	f9	25	0c	0c
0e	f3	b2	4f	21	d2	60	2f	f3	21	41	6e	a8	89	c8	a6

=

After AddRoundKey

Round 7				Round 8				Round 9				Round 10 (Ciphertext)			
5a	19	a3	7a	ea	04	65	85	eb	59	8b	1b	39	02	dc	19
41	49	e0	8c	83	45	5d	96	40	2e	a1	c3	25	dc	11	6a
42	dc	19	04	5c	33	98	b0	f2	38	13	42	84	09	85	0b
b1	1f	65	0c	f0	2d	ad	c5	1e	84	e7	d2	1d	fb	97	32

Pseudo code for the Inverse Cipher

```
EqInvCipher(byte in[4 * Nb], byte out[4 * Nb], word dw[Nb * (Nr + 1)])
begin
   byte  state[4,Nb]

   state = in

   AddRoundKey(state, dw + Nr * Nb)

   for round = Nr - 1 step -1 to 1
      InvSubBytes(state)
      InvShiftRows(state)
      InvMixColumns(state)
      AddRoundKey(state, dw + round * Nb)
   end for

   InvSubBytes(state)
   InvShiftRows(state)
   AddRoundKey(state, dw)

   out = state
end
```

Implementation Issues

- **Key Length Requirements**
  - An implementation of the AES algorithm shall support *at least one* of the three key lengths: 128, 192, or 256 bits (i.e., ***Nk*** = 4, 6, or 8, respectively).
  - Implementations may optionally support two or three key lengths, which may promote the interoperability of algorithm implementations.
- **Keying Restrictions**
  - No weak or semi-weak keys have been identified for the AES algorithm, and there is no restriction on key selection.
- **Parameterization of Key Length, Block Size, and Round Number**
  - This standard explicitly defines the allowed values for the key length (N_k), block size (N_b), and number of rounds (N_r).
  - However, future reaffirmations of this standard could include changes or additions to the allowed values for those parameters. Therefore, implementers may choose to design their AES implementations with future flexibility in mind.

Calculation of Rcon

- Rcon(i) = 02 • Rcon(i-1)   where i is round number
- Rcon(1) = 01
  - So, Rcon used for 1st round is [01 00 00 00] word.
- Rcon(2) = 02 • Rcon(1)
  = 02 • 01 = 02
  - So, Rcon used for 2nd round is [02 00 00 00] word.
- Rcon(3) = 02 • Rcon(2)
  = 02 • 02 = 04
  - So, Rcon used for 3rd round is [02 00 00 00] word.
- Similarly Rcon(4) = 08
  Rcon(5) = 10
  Rcon(6) = 20
  Rcon(7) = 40
  Rcon(8) = 80
  Rcon(9) = 1B
  Rcon(10) = 36

Back