Advanced Encryption Standard (AES)

Dr. B C Dhara

Department of Information Technology

Jadavpur University

Objectives

- Define the basic structure of AES
- Define the transformations used by AES
- Define the key expansion process
- Discuss different implementations

Introduction

- *The Advanced Encryption Standard (AES) is a modern symmetric-key block cipher that may replace DES*
- *This standard is published by the National Institute of Standards and Technology (NIST) in December 2001*

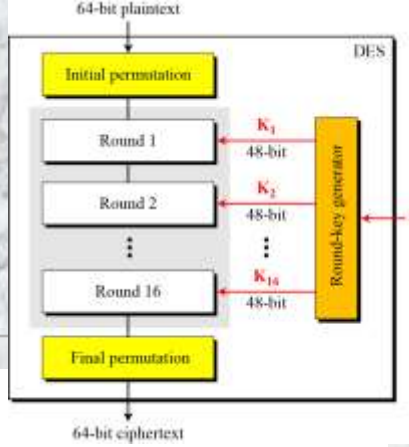
Criteria

- *The criteria for selecting AES fall into three areas:*
  - *Security: main emphasis was on the security*
  - *Cost: computational efficiency and storage requirement*
  - *Implementation: algorithm must have flexibility and simplicity*

General design of AES encryption cipher

- AES is a non-Feistel cipher that encrypts and decrypts a data block of 128 bits
- It uses 10, 12, or 14 rounds
- The key size, which can be 128, 192, or 256 bits, depends on the number of rounds
  - They are referred as AES-128, AES-192 and AES-256
  - Round keys are always 128 bits
- Number of round keys is $N_r + 1$
  - $K_0, K_1, \ldots, K_{N_r}$

General design of AES encryption cipher (contd…)



128-bit plaintext

AES

Round keys (128 bits)

Pre-round transformation

K_0

Round 1

K_1

Round 2

K_2

Round N_r (slightly different)

K_{Nr}

Key expansion

Cipher key (128, 192, or 256 bits)

Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

128-bit ciphertext

Data units

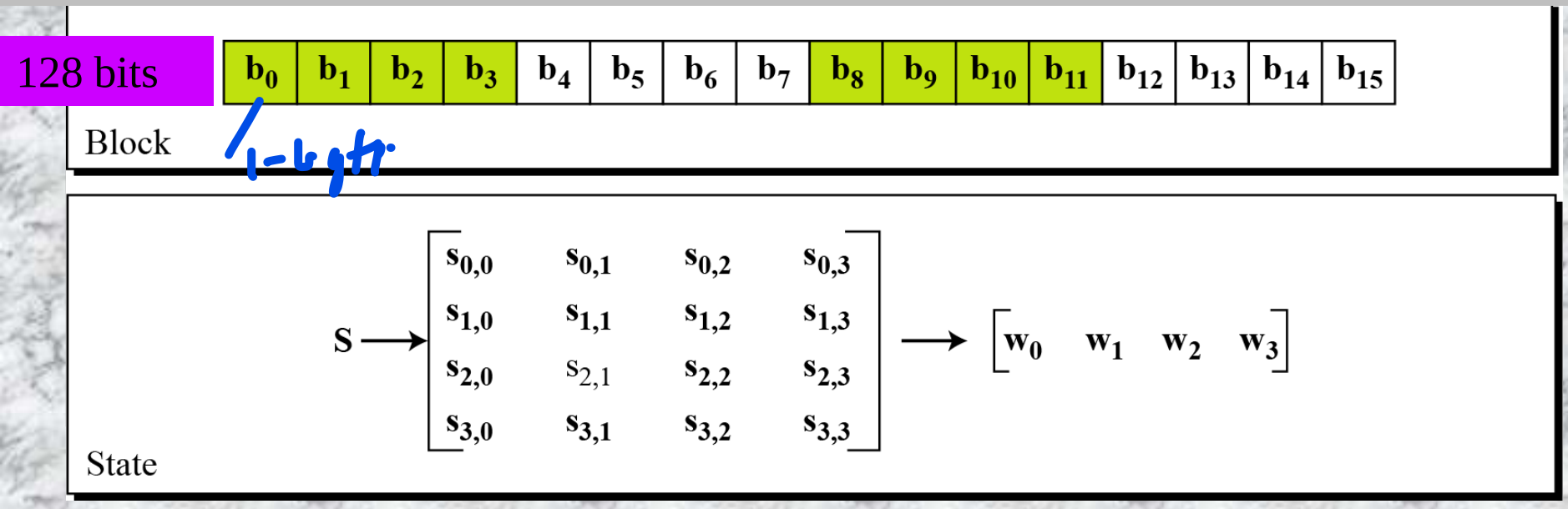
- Byte, word, block and state

AES uses several rounds and each round is made of stages

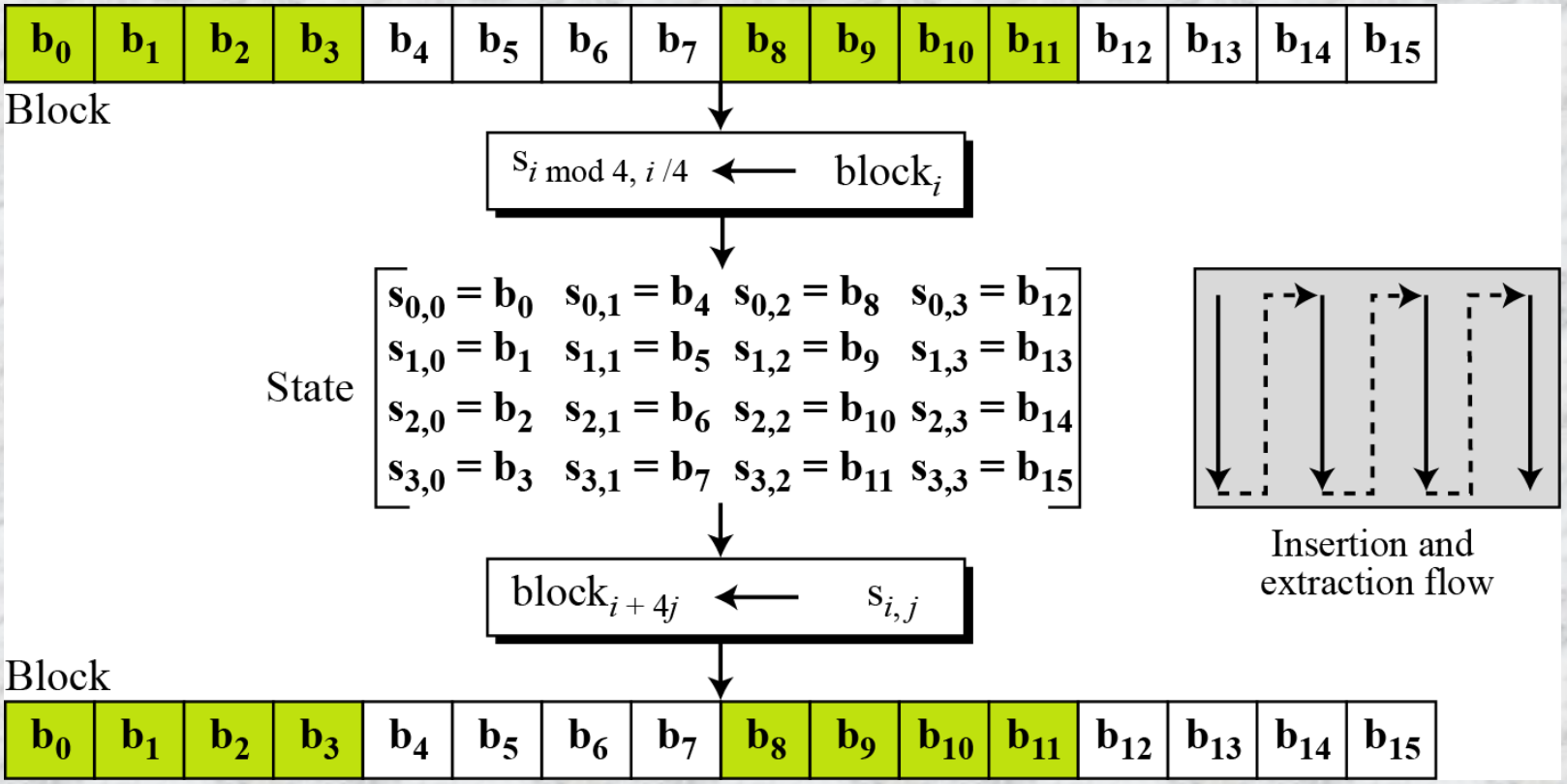
At the beginning and end the term data block is used

Each stage of the rounds transform the data block from one state to another

State is represented by 16 bytes (like data block)



Block-to-state and state-to-block transformation



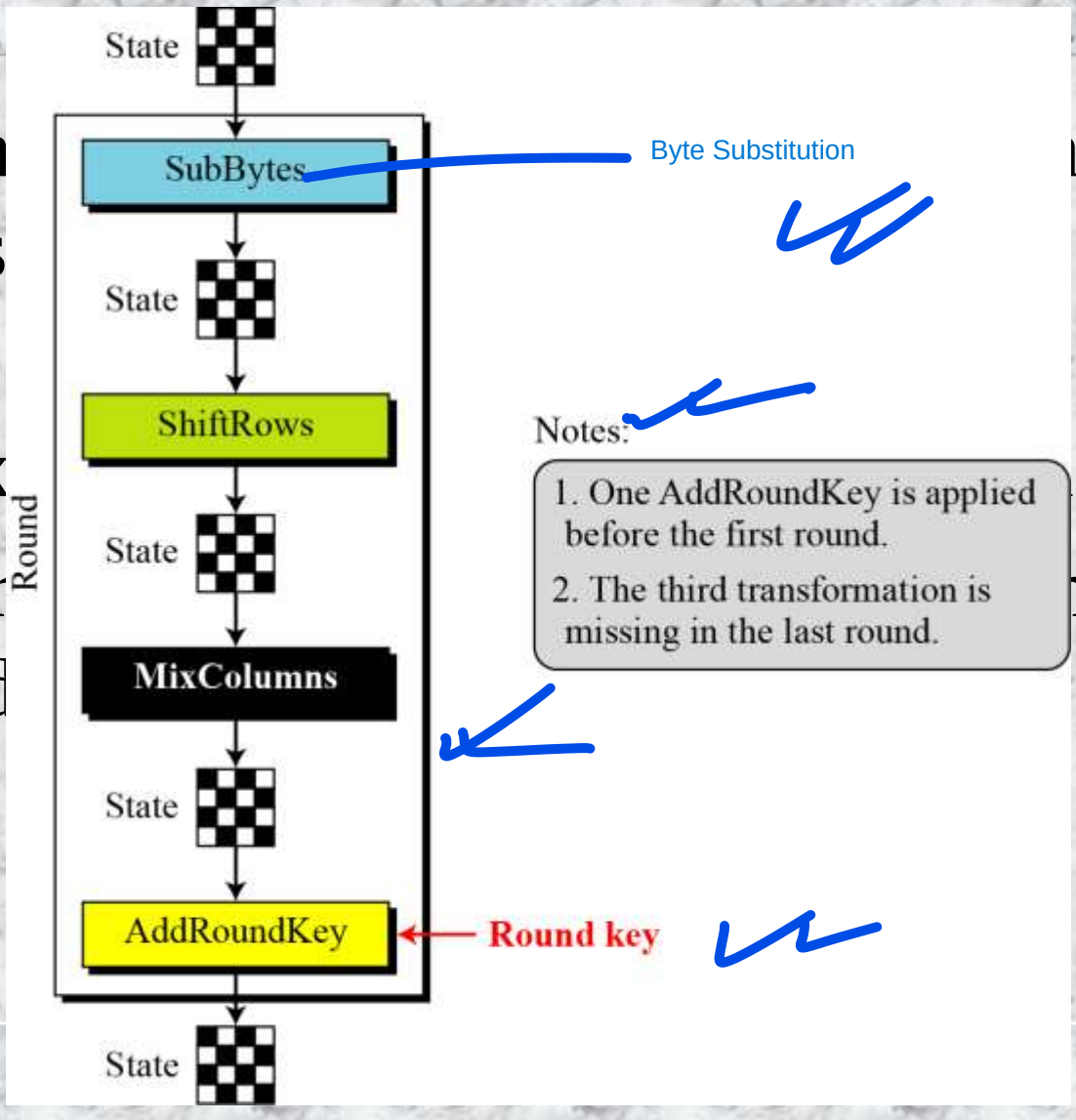
Structure of Each Round

- Each
  trans
- Last
  (Mix
- Pre-r
  (Add

s four

ng)

mation



TRANSFORMATIONS

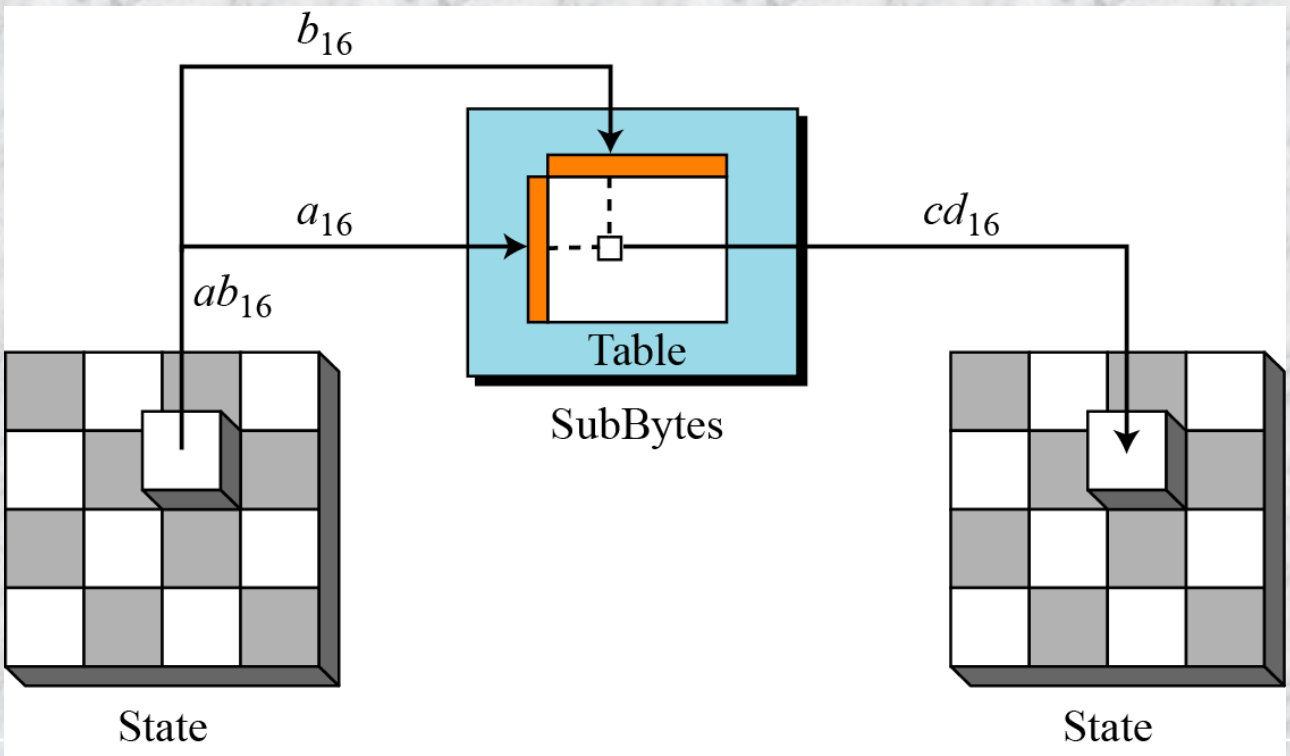
- *To provide security, AES uses four types of transformations:*
  - *substitution,*
  - *permutation,*
  - *mixing, and*
  - *key-adding*

Substitution

- *AES, like DES, uses substitution*
- *Transformation is done for each byte*
- If two bytes are same, transformation is also same
- Transformation is defined either by table lookup or mathematical computation in GF(2^8)

Substitution (contd…)

- *To substitute a byte, a byte is interpreted as two hexadecimal digits*



1-byte

2^4 x 2^4 (matrix)

Table 5.2 *SubBytes transformation table*

	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
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	CB	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

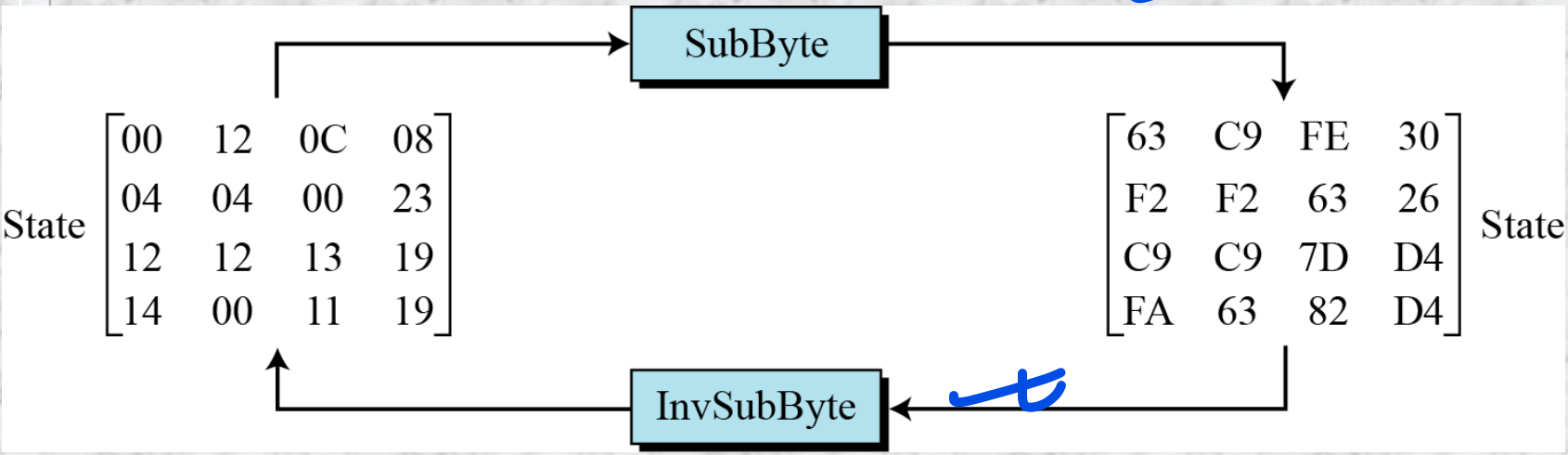
Table 7.2 *InvSubBytes transformation table*

Inverse S-1

	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
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

Substitution example

- How a state is transformed using the SubBytes transformation is shown below.



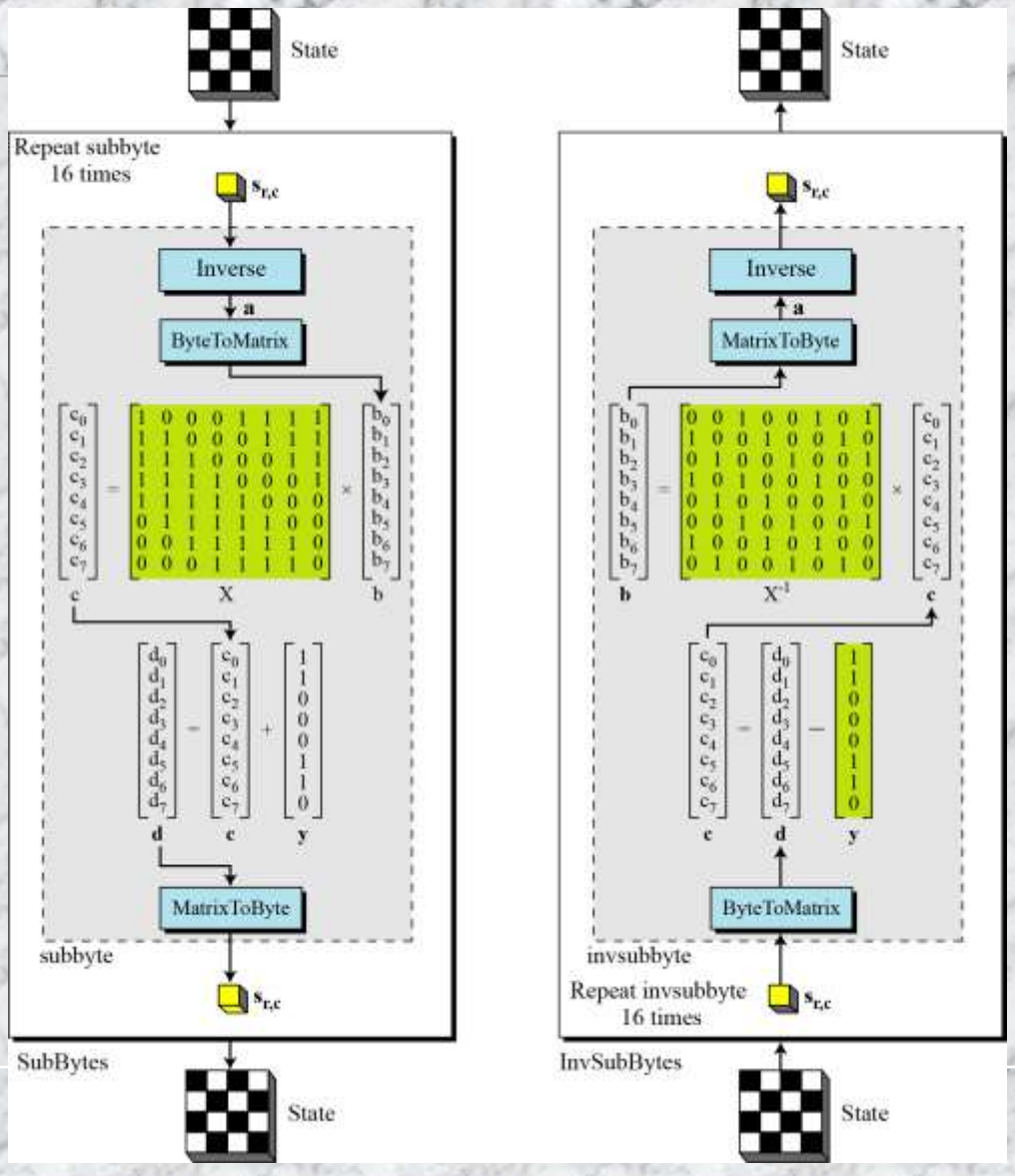
Substitution in GF(2^8)

- *Transformation Using the GF(2^8) Field*
- *AES also defines the transformation algebraically using the GF(2^8) field with the irreducible polynomials ($x^8 + x^4 + x^3 + x + 1$)*

subbyte: $\rightarrow \quad \mathbf{d} = \mathbf{X}\,(s_{r,c})^{-1} \oplus \mathbf{y}$

invsubbyte: $\rightarrow \quad [\mathbf{X}^{-1}(\mathbf{d} \oplus \mathbf{y})]^{-1} = [\mathbf{X}^{-1}(\mathbf{X}\,(s_{r,c})^{-1} \oplus \mathbf{y} \oplus \mathbf{y})]^{-1} = [(s_{r,c})^{-1}]^{-1} = s_{r,c}$

SubBytes and InvSubBytes processes



SubBytes and InvSubBytes processes: example

- Let us show how the byte 0C is transformed to FE by subbyte routine and transformed back to 0C by the invsubbyte routine.

1. *subbyte:*
   a. The multiplicative inverse of 0C in GF(2^8) field is B0, which means **b** is (10110000).
   b. Multiplying matrix **X** by this matrix results in **c** = (10011101)
   c. The result of XOR operation is **d** = (11111110), which is FE in hexadecimal.

2. *invsubbyte:*
   a. The result of XOR operation is **c** = (10011101)
   b. The result of multiplying by matrix $\mathbf{X}^{-1}$ is (11010000) or B0
   c. The multiplicative inverse of B0 is 0C.

Pseudocode for SubBytes

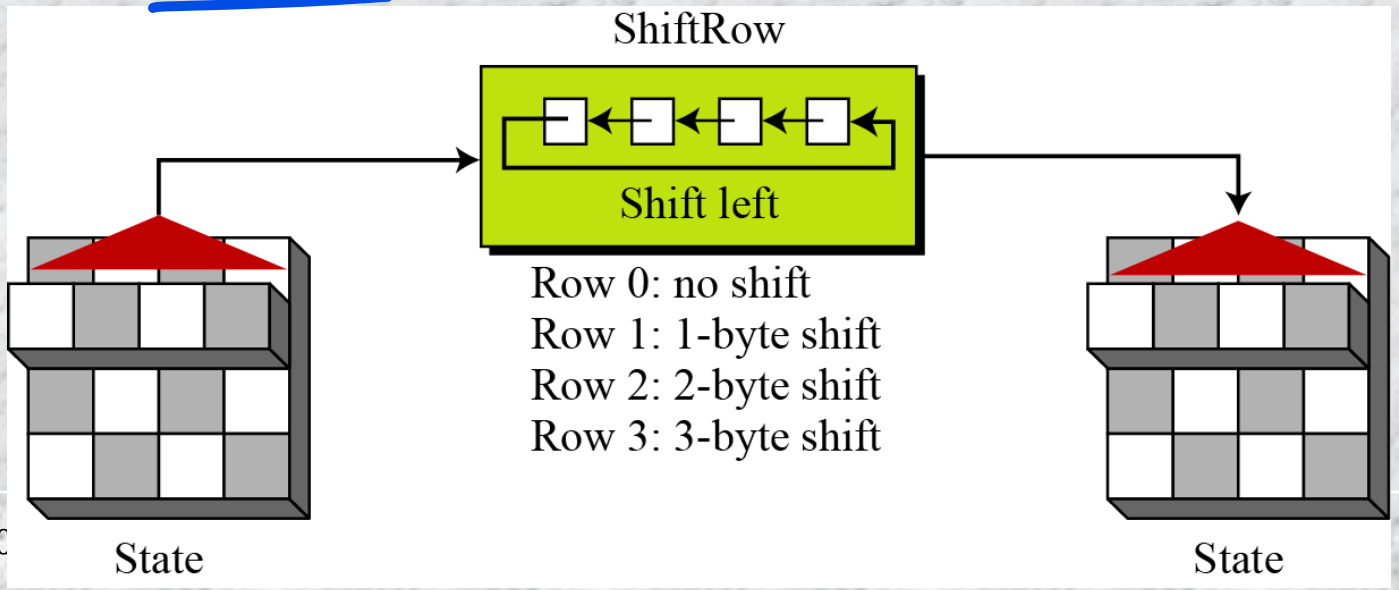
Algorithm 7.1 *Pseudocode for SubBytes transformation*

```
SubBytes (S)
{
   for (r = 0 to 3)
     for (c = 0 to 3)
           S_r,c = subbyte (S_r,c)
}
```

```
subbyte (byte)
{
   a ← byte^-1                 // Multiplicative inverse in GF(2^8) with inverse of 00 to be 00
    ByteToMatrix (a, b)
    for (i = 0 to 7)
    {
        c_i ← b_i ⊕ b_(i+4)mod 8 ⊕ b_(i+5)mod 8 ⊕ b_(i+6)mod 8 ⊕ b_(i+7)mod 8
        d_i ← c_i ⊕ ByteToMatrix (0x63)
   }
    MatrixToByte (d, d)
    byte ← d
}
```

Transformation: permutation

- *Another transformation found in a round is shifting, which permutes the bytes.*
- *ShiftRows*
  - *In the encryption, the transformation is called ShiftRows*



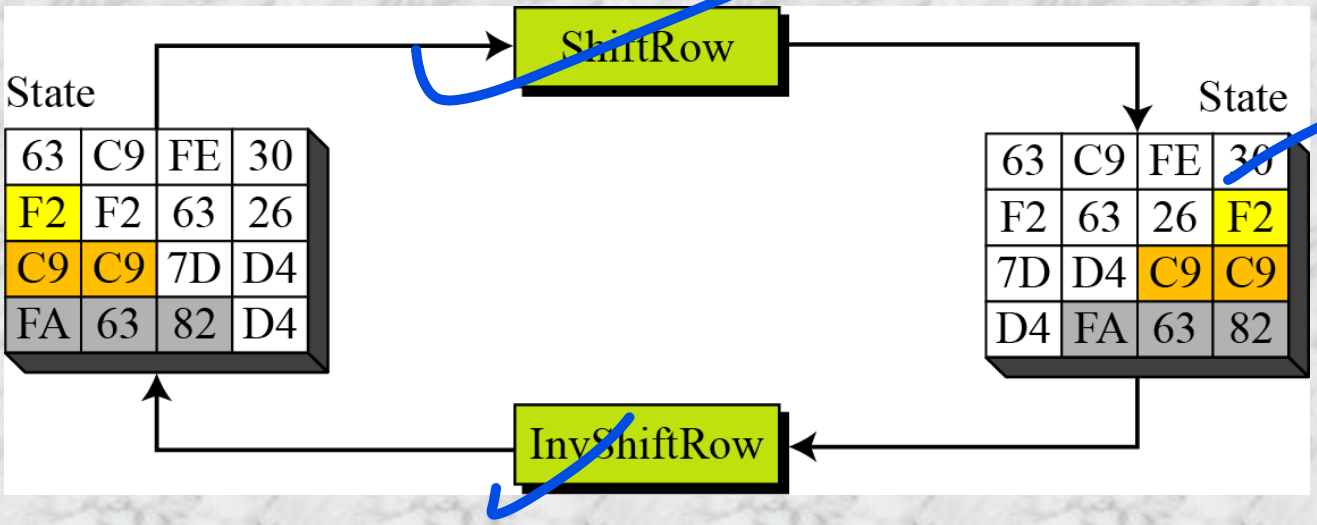
Pseudocode for shiftRows

Algorithm 7.2 *Pseudocode for ShiftRows transformation*

```
ShiftRows (S)
{
    for (r = 1 to 3)
        shiftrow (s_r, r)                    // s_r is the rth row
}

shiftrow (row, n)                            // n is the number of bytes to be shifted
{
   CopyRow (row, t)                          // t is a temporary row
   for (c = 0 to 3)
        row_((c – n) mod 4) ← t_c
}
```

ShiftRows: example

- How a state is transformed using ShiftRows transformation



Mixing

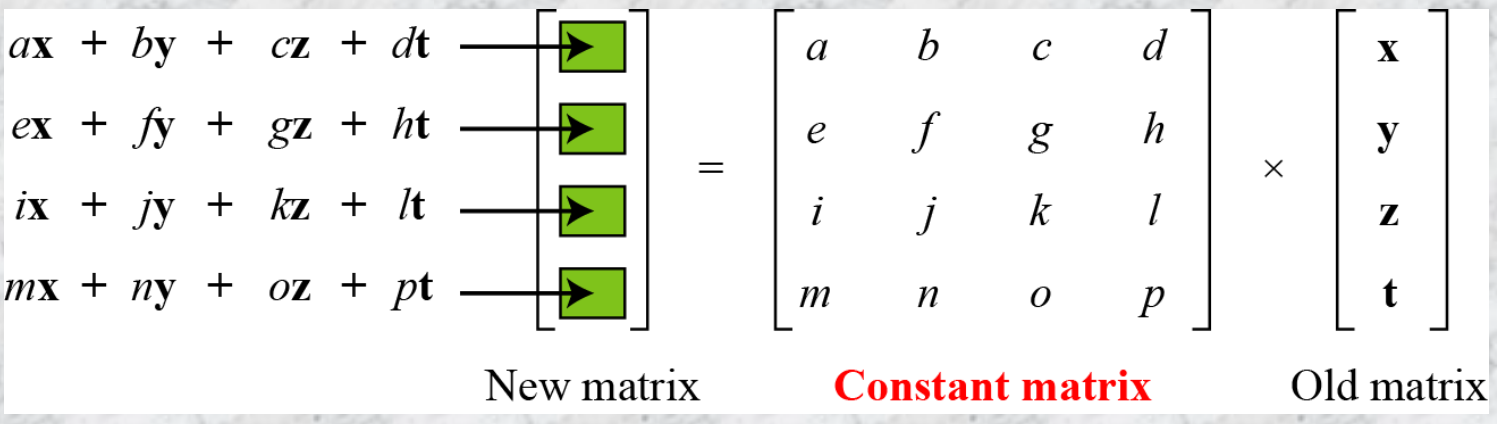
- *The* Subbyte transformation is an intrabyte transformation
  - *It depends only on the value of the byte and table, not depends on the neighbor bytes*
- *ShiftRows exchanges bytes without permuting bits inside the bytes*
  - *This is a byte-exchange transformation*

Mixing (contd…)

- *We need an interbyte transformation that changes the bits inside a byte, based on the bits inside the neighboring bytes*
- *We need to mix bytes to provide diffusion at the bit level*
- *The mixing transformation changes the contents of each bytes*
  - *Takes four bytes at a time and recreate four new bytes*
  - *New bytes are different (even all four bytes are same)*

Mixing (contd…)

- Multiplies each byte with different constant and then mix them
  - Multiply a square matrix (4x4) with a column matrix (4x1), which results a new column matrix



Mixing (contd…)

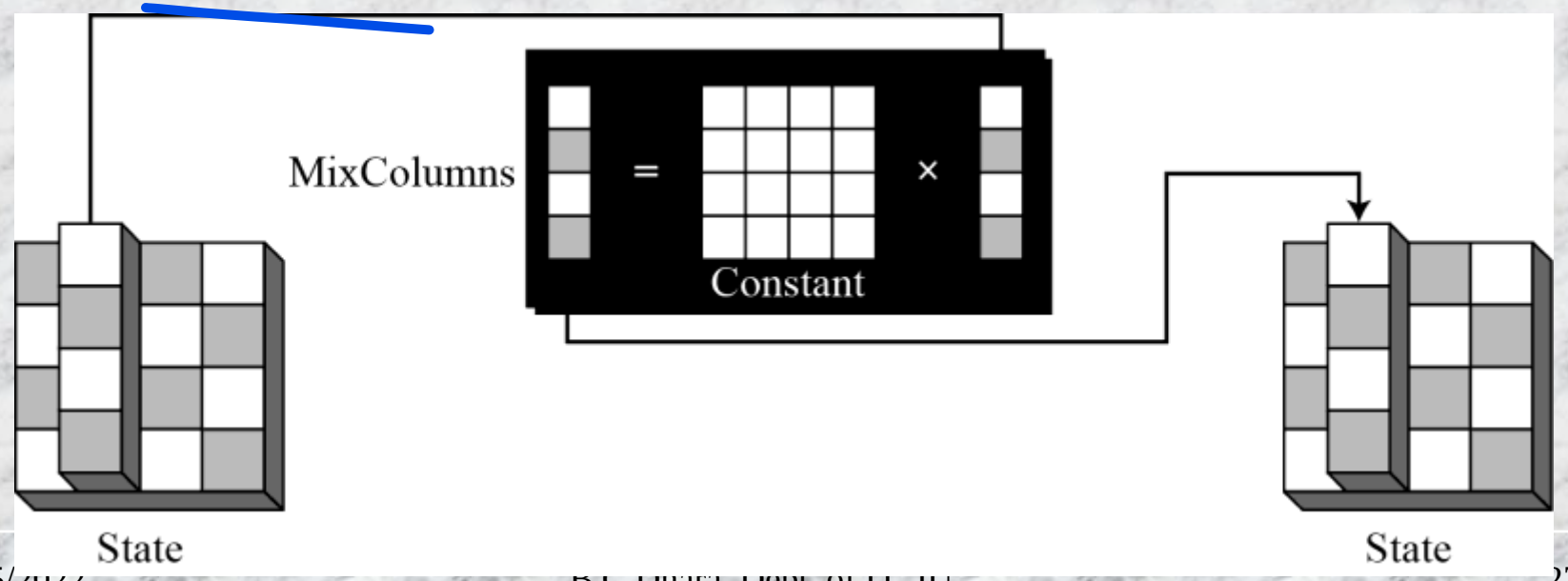
- MixColumns operation is used to achieve the goal
  - This is invertible
  - Elements are interpreted as 8-bit words (polynomial) with coefficients in GF(2^8) respect to the irreducible polynomial $x^8 + x^4 + x^3 + x + 1$

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \quad \text{Inverse} \quad \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix}$$

C $\qquad$ C^{-1}

MixColumns

- *The MixColumns transformation operates at the column level*
  - *it transforms each column of the state to a new column.*



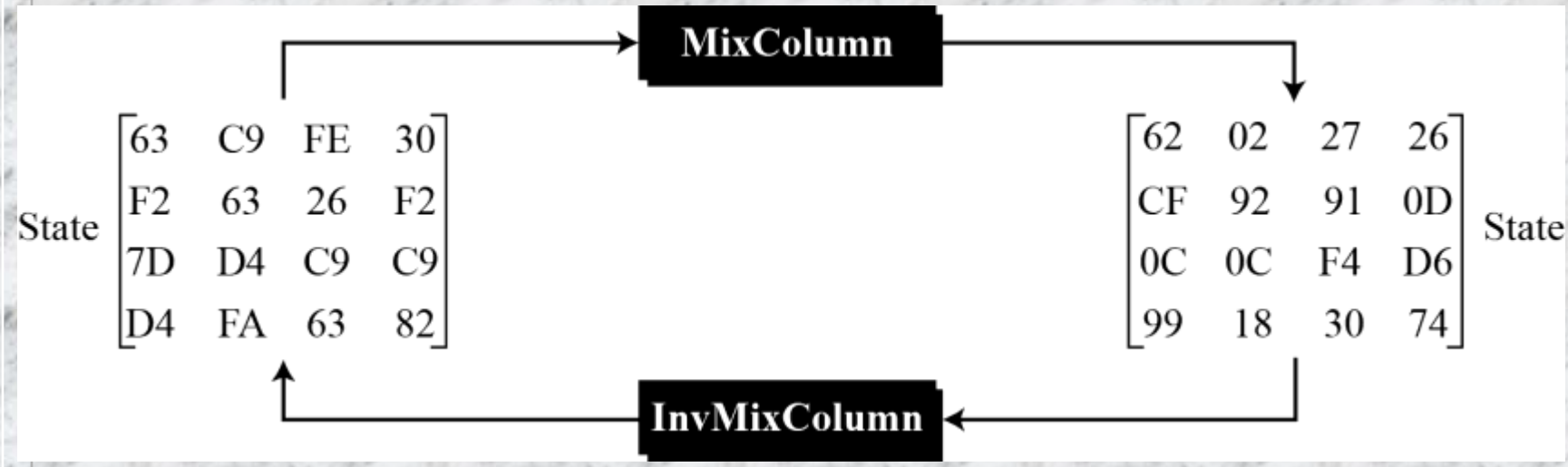
Pseudocode for MixColumns

Algorithm 7.3 *Pseudocode for MixColumns transformation*

```
MixColumns (S)
{
    for (c = 0 to 3)
        mixcolumn (s_c)
}

mixcolumn (col)
{
   CopyColumn (col, t)                    // t is a temporary column

   col_0 ← (0x02) • t_0 ⊕ (0x03 • t_1) ⊕ t_2 ⊕ t_3

   col_1 ← t_0 ⊕ (0x02) • t_1 ⊕ (0x03) • t_2 ⊕ t_3

   col_2 ← t_0 ⊕ t_1 ⊕ (0x02) • t_2 ⊕ (0x03) • t_3

   col_3 ← (0x03 • t_0) ⊕ t_1 ⊕ t_2 ⊕ (0x02) • t_3
}
```

MixColumns: example

- How a state is transformed using the MixColumns transformation

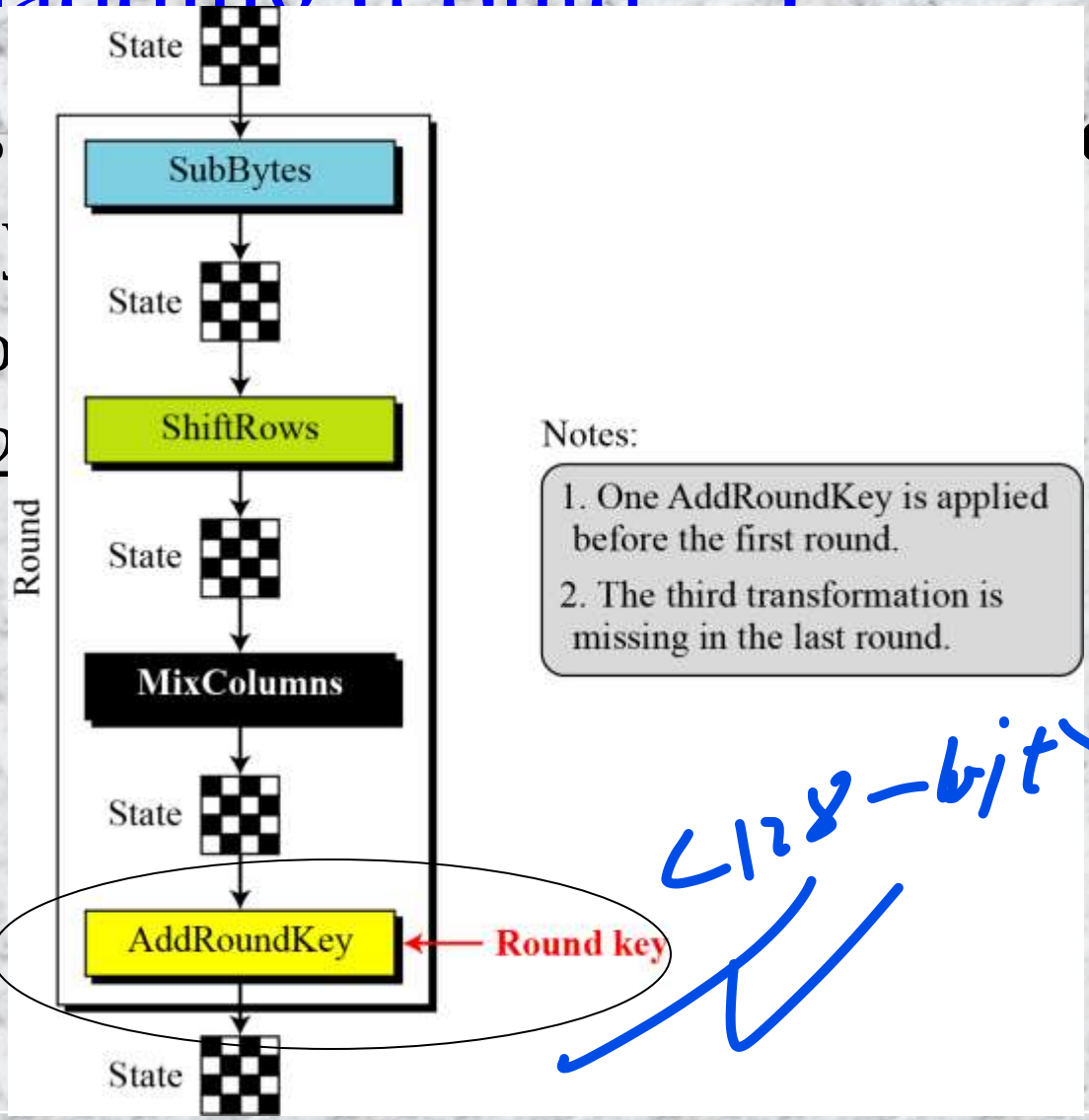


Key-adding

- All the previous transformation (*substitution, permutation, mixing) are invertible*
- *if cipher key is not added to the state at each round*
  - *It is very easy to the adversary to find the plaintext, given the ciphertext*
  - *Key is the only secret*

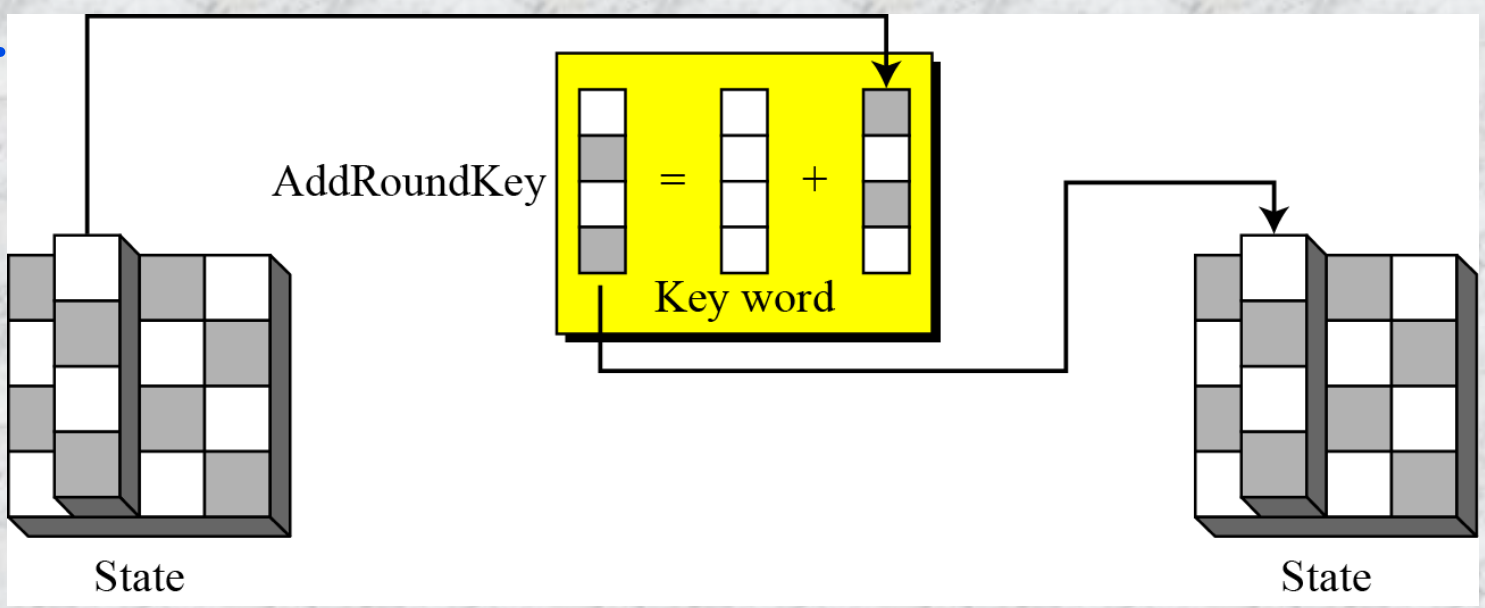
Key-adding (contd...)

- A proces … es N_r+1 round ke…
  - Each ro…
  - Four 32…



AddRoundKey

- *AddRoundKey proceeds one column of the state at a time.*
- *AddRoundKey adds a round key word with each state column matrix; the operation in AddRoundKey is matrix addition.*



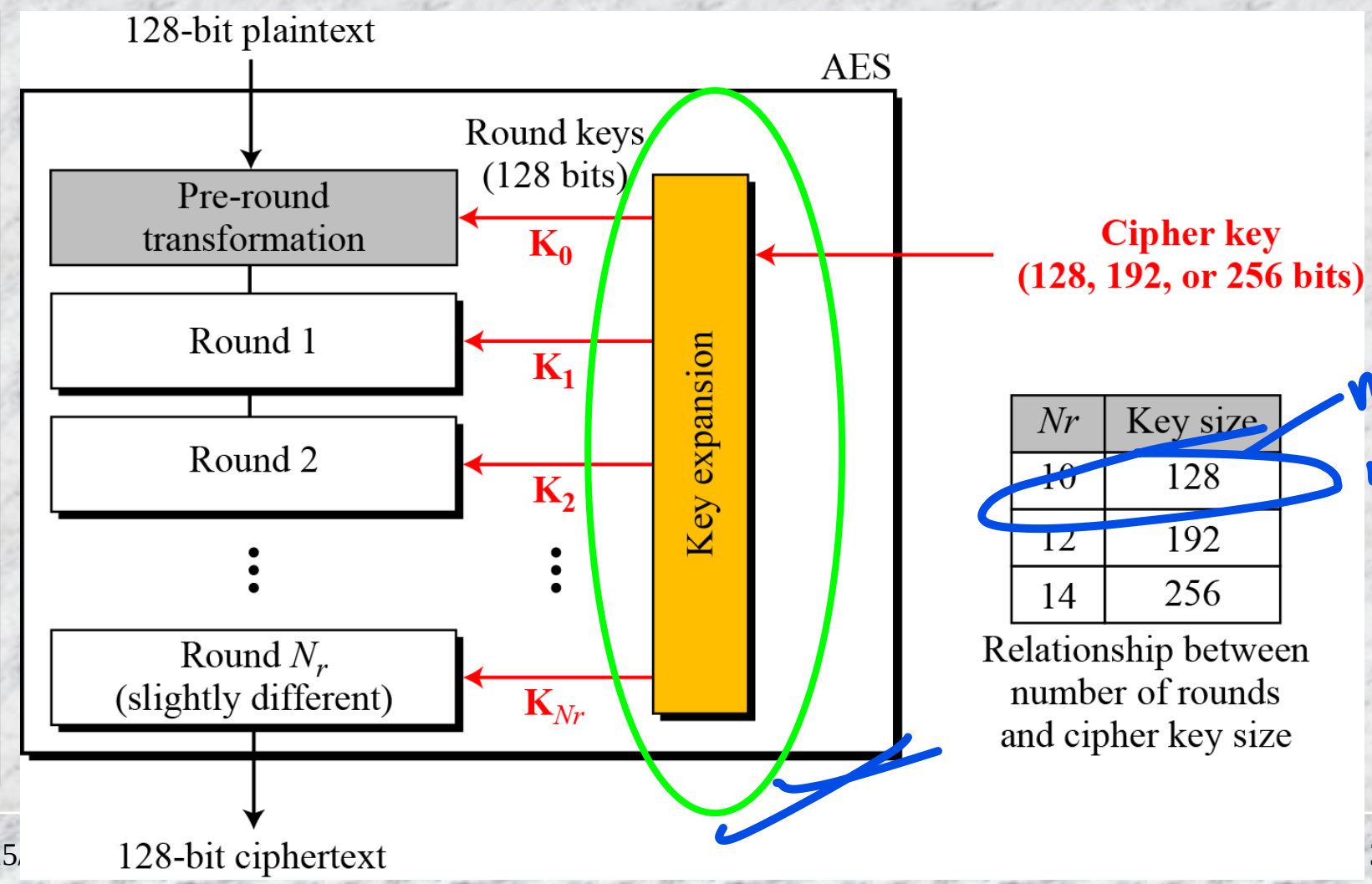
The AddRoundKey transformation is the inverse of itself.

Pseudocode for AddRoundKey

Algorithm 7.4 *Pseudocode for AddRoundKey transformation*

```
AddRoundKey (S)
{
    for (c = 0 to 3)
        s_c ← s_c ⊕ w_round + 4c
}
```

KEY EXPANSION



Nr	Key size
10	128
12	192
14	256

Relationship between number of rounds and cipher key size

KEY EXPANSION (contd…)

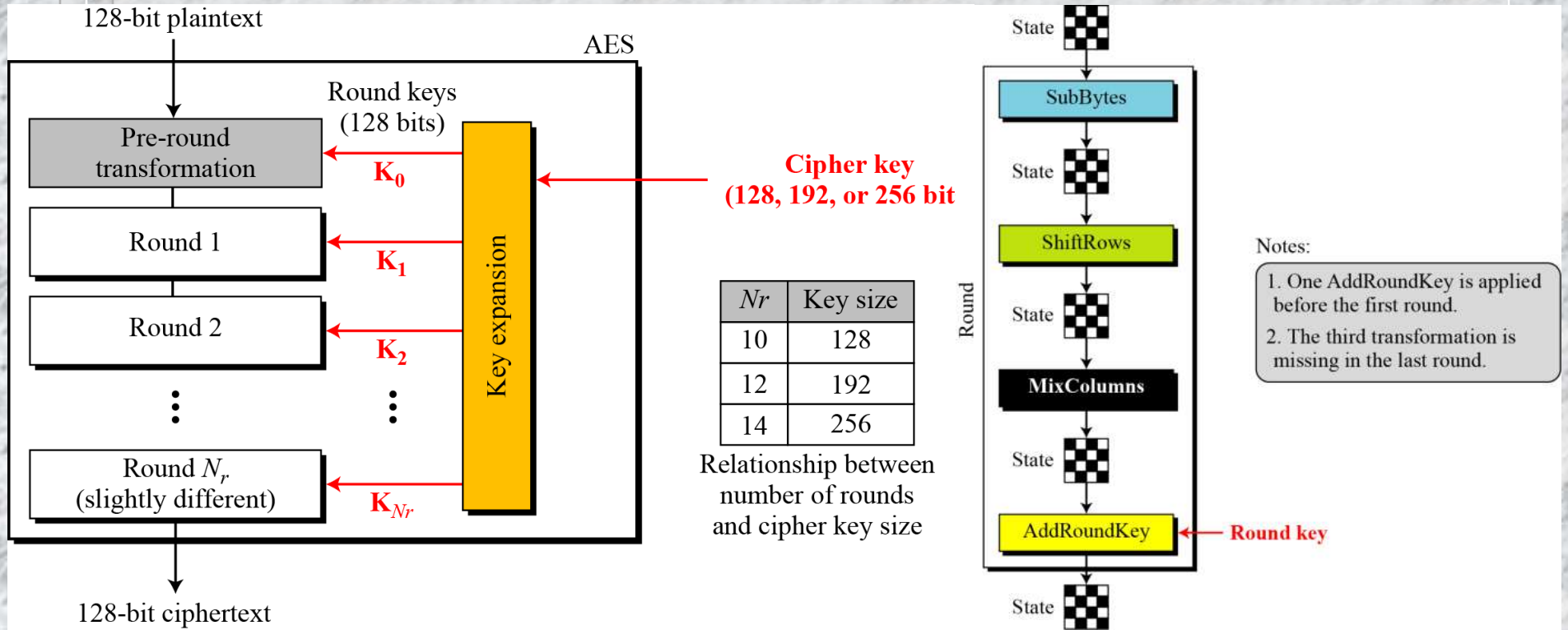
- *To create round keys for each round, AES uses a key-expansion process*
- *If the number of rounds is N_r , the key-expansion routine creates $N_r + 1$ 128-bit round keys from one single cipher key*

  - Key Expansion in AES-128
  - Key Expansion in AES-192 and AES-256
  - Key-Expansion Analysis

KEY EXPANSION (contd…

First round key is used for pre-round transformation

Remaining round keys are used for at the end of each round



Nr	Key size
10	128
12	192
14	256

KEY EXPANSION (contd…)

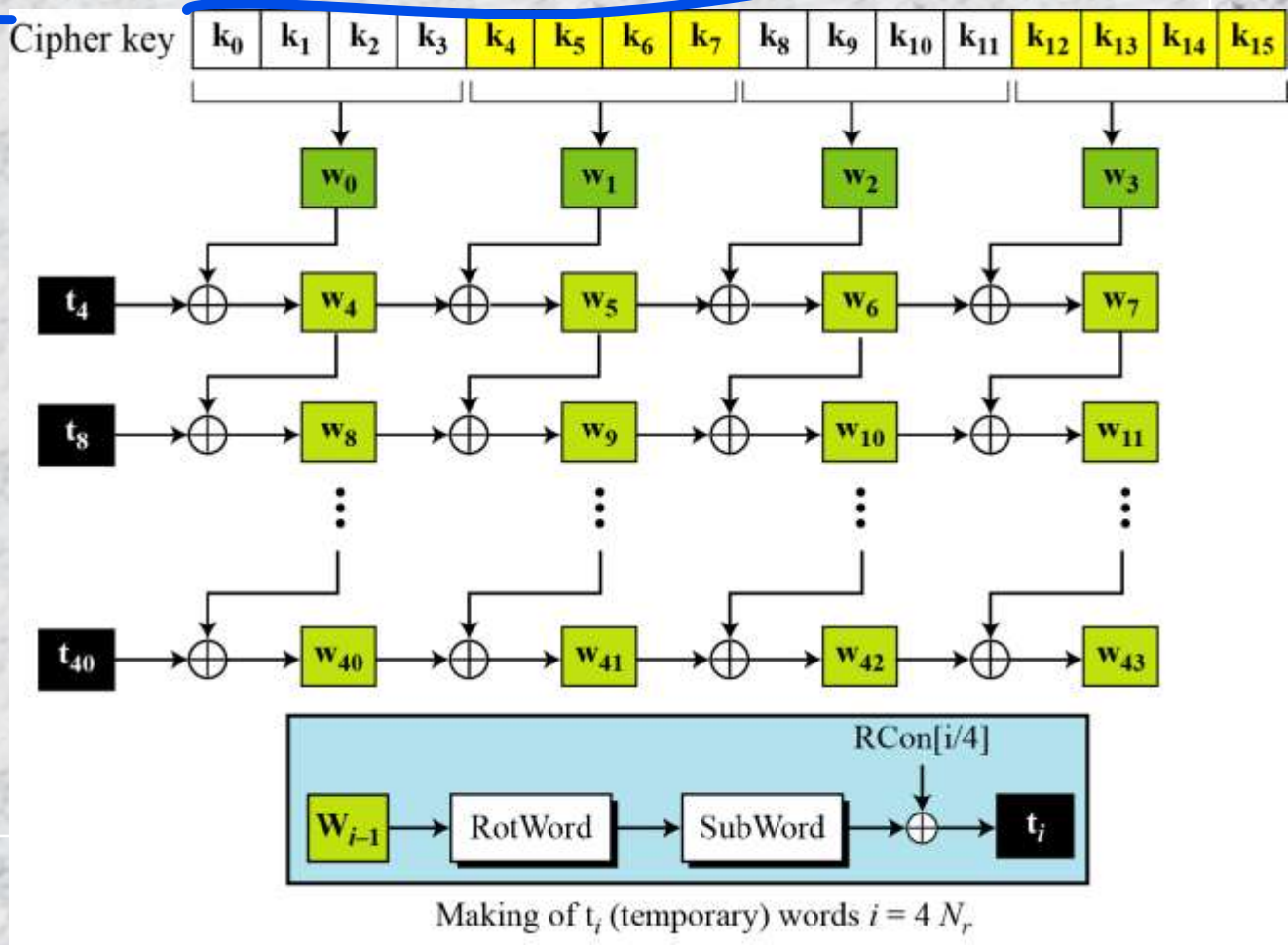
- Key-expansion creates round keys word-by-word in an array of four bytes

Table 7.3 *Words for each round*

Round	*Words*			
Pre-round	$\mathbf{w}_0$	$\mathbf{w}_1$	$\mathbf{w}_2$	$\mathbf{w}_3$
1	$\mathbf{w}_4$	$\mathbf{w}_5$	$\mathbf{w}_6$	$\mathbf{w}_7$
2	$\mathbf{w}_8$	$\mathbf{w}_9$	$\mathbf{w}_{10}$	$\mathbf{w}_{11}$
. . .	. . .			
N_r	$\mathbf{w}_{4N_r}$	$\mathbf{w}_{4N_r+1}$	$\mathbf{w}_{4N_r+2}$	$\mathbf{w}_{4N_r+3}$

KEY EXPANSION in AES-128

- AES-128 has 10 rounds
  - 11 keys ➔ 44 words, made from the 128-bit cipher key



Making of t_i (temporary) words $i = 4\ N_r$

Round key generation process

- Pre-round key (w_0, w_1, w_2, w_3) is generated from the cipher key
  - $w_0 = [k_0, \ldots, k_3]$, w_1, w_2, $w_3 \ldots$
- For i= 4 to 43
  - If $i \neq 4r$, $w_i = w_{i-1} \oplus w_i$
  - Else $w_i = t \oplus w_{i-4}$, where
    $t = \text{SubWord}(\text{RotWord}(w_{i-1})) \oplus \text{RCon}_{\text{int}(i/4)}$

Round key generation process (contd…)

- RotWord: rotated word, same as ShiftRows
  - Here only one row (4 bytes, word) will be rotated
- SubWord: substitute word, similar as SubByte
  - Applied on four bytes, takes each byte of the word and substitutes another byte for it
- Round Constant: a four byte number, the rightmost three bytes are always zero

Round key generation process (contd…)

- The constants are determined as
  - $RC_i = x^{i-1} \bmod x^8 + x^4 + x^3 + x + 1$

Table 7.4 *RCon constants*

Round	*Constant (RCon)*	*Round*	*Constant (RCon)*
1	(**01** 00 00 00)$_{16}$	6	(**20** 00 00 00)$_{16}$
2	(**02** 00 00 00)$_{16}$	7	(**40** 00 00 00)$_{16}$
3	(**04** 00 00 00)$_{16}$	8	(**80** 00 00 00)$_{16}$
4	(**08** 00 00 00)$_{16}$	9	(**1B** 00 00 00)$_{16}$
5	(**10** 00 00 00)$_{16}$	10	(**36** 00 00 00)$_{16}$

$RC_1 \rightarrow 01_{16}$
$RC_2 \rightarrow 02_{16}$
$RC_3 \rightarrow 04_{16}$
$RC_4 \rightarrow 08_{16}$
$RC_5 \rightarrow 10_{16}$
$RC_6 \rightarrow 20_{16}$
$RC_7 \rightarrow 40_{16}$
$RC_8 \rightarrow 80_{16}$
$RC_9 \rightarrow 1B_{16}$
$RC_{10} \rightarrow 36_{16}$

Pseudocode for Key Expansion in AES-128

Algorithm 7.5 *Pseudocode for key expansion in AES-128*

```
KeyExpansion ([key0 to key15], [w0 to w43])
{
    for (i = 0 to 3)
        wi ← key4i + key4i+1 + key4i+2 + key4i+3

    for (i = 4 to 43)
    {
        if (i mod 4 ≠ 0)    wi ← wi–1 + wi–4
        else
        {
            t ← SubWord (RotWord (wi–1)) ⊕ RConi/4        // t is a temporary word
            wi ← t + wi–4
        }
    }
}
```

Key expansion example

- How the keys for each round are calculated assuming that the 128-bit cipher key $(24\ 75\ A2\ B3\ 34\ 75\ 56\ 88\ 31\ E2\ 12\ 00\ 13\ AA\ 54\ 87)_{16}$.

Table 7.5 *Key expansion example*

Round	*Values of t's*	*First word in the round*	*Second word in the round*	*Third word in the round*	*Fourth word in the round*
—		$w_{00} = 2475A2B3$	$w_{01} = 34755688$	$w_{02} = 31E21200$	$w_{03} = 13AA5487$
1	AD20177D	$w_{04} = 8955B5CE$	$w_{05} = BD20E346$	$w_{06} = 8CC2F146$	$w_{07} = 9F68A5C1$
2	470678DB	$w_{08} = CE53CD15$	$w_{09} = 73732E53$	$w_{10} = FFB1DF15$	$w_{11} = 60D97AD4$
3	31DA48D0	$w_{12} = FF8985C5$	$w_{13} = 8CFAAB96$	$w_{14} = 734B7483$	$w_{15} = 2475A2B3$
4	47AB5B7D	$w_{16} = B822deb8$	$w_{17} = 34D8752E$	$w_{18} = 479301AD$	$w_{19} = 54010FFA$
5	6C762D20	$w_{20} = D454F398$	$w_{21} = E08C86B6$	$w_{22} = A71F871B$	$w_{23} = F31E88E1$
6	52C4F80D	$w_{24} = 86900B95$	$w_{25} = 661C8D23$	$w_{26} = C1030A38$	$w_{27} = 321D82D9$
7	E4133523	$w_{28} = 62833EB6$	$w_{29} = 049FB395$	$w_{30} = C59CB9AD$	$w_{31} = F7813B74$
8	8CE29268	$w_{32} = EE61ACDE$	$w_{33} = EAFE1F4B$	$w_{34} = 2F62A6E6$	$w_{35} = D8E39D92$
9	0A5E4F61	$w_{36} = E43FE3BF$	$w_{37} = 0EC1FCF4$	$w_{38} = 21A35A12$	$w_{39} = F940C780$
10	3FC6CD99	$w_{40} = DBF92E26$	$w_{41} = D538D2D2$	$w_{42} = F49B88C0$	$w_{43} = 0DDB4F40$

Round keys

- Each round key in AES depends on the previous round key
- The dependency, however, is nonlinear because of SubWord transformation
- The addition of the round constants also guarantees that each round key will be different from the previous one

Round keys from different cipher keys

- The two sets of round keys can be created from two cipher keys that are different only in one bit

Table 7.6 *Comparing two sets of round keys*

R.	*Round keys for set 1*	*Round keys for set 2*	*B. D.*
—	1245A2A1 2331A4A3 B2CCAA34 C2BB7723	1245A2A1 2331A4A3 B2CCAB34 C2BB7723	01
1	F9B08484 DA812027 684D8A13 AAF6FD30	F9B08484 DA812027 684D8B13 AAF6FC30	02
2	B9E48028 6365A00F 0B282A1C A1DED72C	B9008028 6381A00F 0BCC2B1C A13AD72C	17
3	A0EAF11A C38F5115 C8A77B09 6979AC25	3D0EF11A 5E8F5115 55437A09 F479AD25	30
4	1E7BCEE3 DDF49FF6 1553E4FF 7C2A48DA	839BCEA5 DD149FB0 8857E5B9 7C2E489C	31
5	EB2999F3 36DD0605 238EE2FA 5FA4AA20	A2C910B5 7FDD8F05 F78A6ABC 8BA42220	34
6	82852E3C B4582839 97D6CAC3 C87260E3	CB5AA788 B487288D 430D4231 C8A96011	56
7	82553FD4 360D17ED A1DBDD2E 69A9BDCD	588A2560 EC0D0DED AF004FDC 67A92FCD	50
8	D12F822D E72295C0 46F948EE 2F50F523	0B9F98E5 E7929508 4892DAD4 2F3BF519	44
9	99C9A438 7EEB31F8 38127916 17428C35	F2794CF0 15EBD9F8 5D79032C 7242F635	51
10	83AD32C8 FD460330 C5547A26 D216F613	E83BDAB0 FDD00348 A0A90064 D2EBF651	52

Example

- The concept of weak keys, as we discussed for DES, does not apply to AES
- Assume that all bits in the cipher key are 0s. The following shows the words for some rounds:

Pre-round:	00000000	00000000	00000000	00000000
Round 01:	62636363	62636363	62636363	62636363
Round 02:	9B9898C9	F9FBFBAA	9B9898C9	F9FBFBAA
Round 03:	90973450	696CCFFA	F2F45733	0B0FAC99
. . .	. . .	. . .	. . .	. . .
Round 10:	B4EF5BCB	3E92E211	23E951CF	6F8F188E

  - The words in the pre-round and the first round are all the same
  - In the second round, the first word matches with the third; the second word matches with the fourth
  - However, after the second round the pattern disappears; every word is different.

Key expansion in AES-192 and AES-256

- *Key-expansion algorithms in the AES-192 and AES-256 versions are very similar to the key expansion algorithm in AES-128, with the following differences:*
  - stead of four
  - i-8

Algorithm 7.5 *Pseudocode for key expansion in AES-128*

```
KeyExpansion ([key0 to key15], [w0 to w43])
{
    for (i = 0 to 3)
        wi ← key4i + key4i+1 + key4i+2 + key4i+3

    for (i = 4 to 43)
    {
        if (i mod 4 ≠ 0)    wi ← wi–1 + wi–4
        else
        {
            t ← SubWord (RotWord (wi–1)) ⊕ RConi/4        // t is a temporary word
            wi ← t + wi–4
        }
    }
}
```

Key-Expansion Analysis

- *The key-expansion in AES has been designed to provide several features that thwart the cryptanalyst*
  - *If a part of the cipher key or words in some round keys are known to an adversary, the adversary needs to know rest of the cipher key only then all round keys will be known*
    - *This is due to non-linearity of the SubWord transformation*
  - *Two different cipher key produce two expansions that differ in at least some rounds*
  - *Each bit of the cipher key is diffused into several rounds*

Key-Expansion Analysis (contd…)

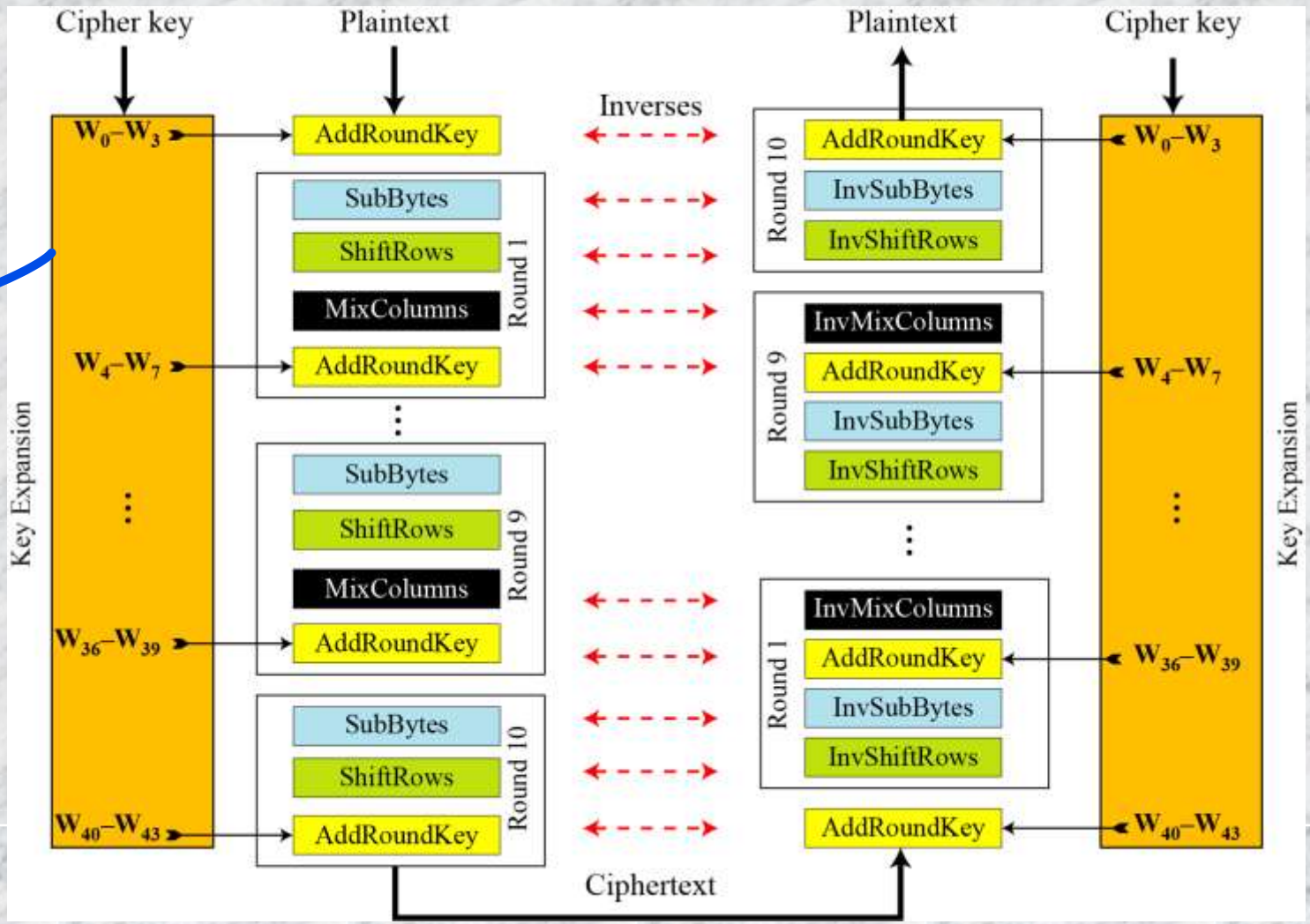
- *Rcons, removes symmetry that may have been created by some transformations*
- *There are no serious weak key in AES*
- *It is easily implementable*
- *Without using table, this can be implemented with the arithmetics in GF(2^8) and GF(2) fields.*

Ciphers

- AES uses four different types of transformations for encryption and decryption
- AES is a non-Feistel cipher
  - (i.e., all the transformation or group of transformations must be invertible)
- Two different designs have considered for implementation of all AES versions
  - Original design
  - Alternative design

Original Design

The order of SubBytes and ShiftRows is changed in the reverse cipher
Similarly, for MixColumns and AddRoundKey



Pseudocode for the AES-128 cipher

Algorithm 7.6 *Pseudocode for cipher in the original design*

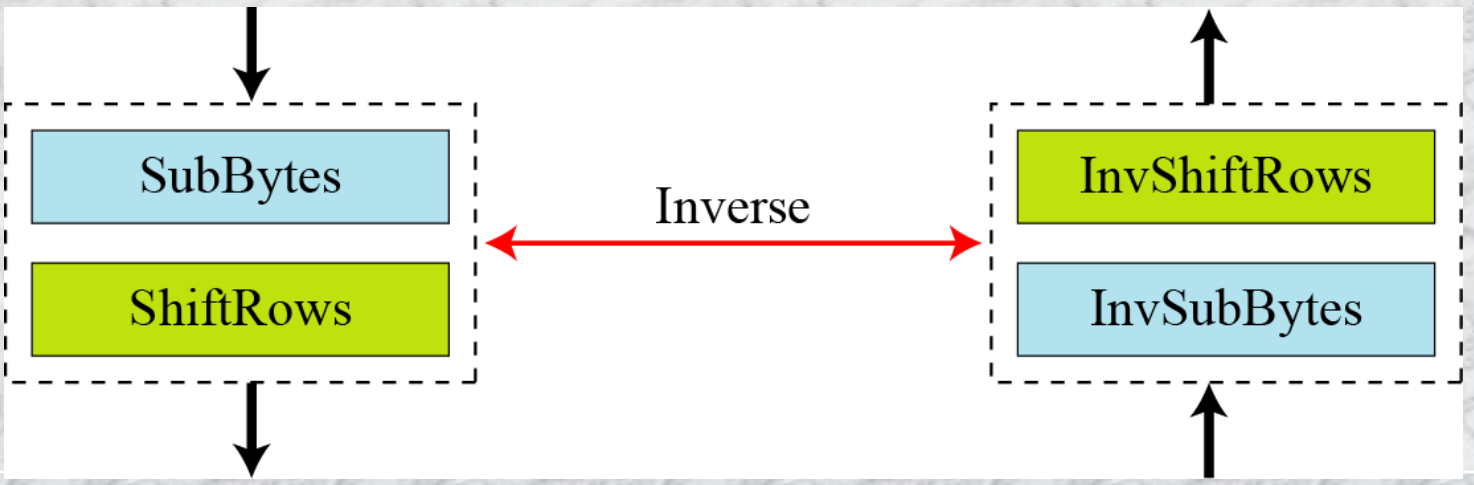
```
Cipher (InBlock [16], OutBlock[16], w[0 … 43])
{
    BlockToState (InBlock, S)

    S ← AddRoundKey (S, w[0…3])
     for (round = 1 to 10)
     {
         S ← SubBytes (S)
         S ← ShiftRows (S)
          if (round ≠ 10)   S ← MixColumns (S)
          S ← AddRoundKey (S, w[4 × round, 4 × round + 3])
     }

    StateToBlock (S, OutBlock);
}
```

Alternative design

- The order of transformations in cipher and reverse cipher is same
- Invertibility is provided for a pair of transformations, not for each single transformation

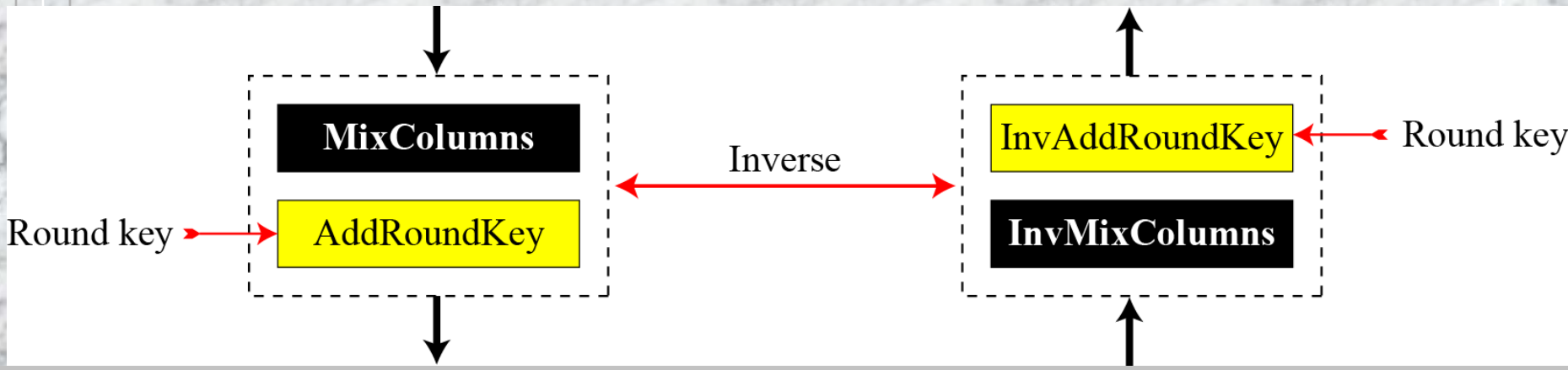
Invertibility of SubBytes and ShiftRows combinations

- SubBytes change the contents of each byte without changing the order of the bytes
- ShiftRows change the order of the bytes without changing the contents of the bytes



Invertibility of MixColumns and AddRoundKey combination

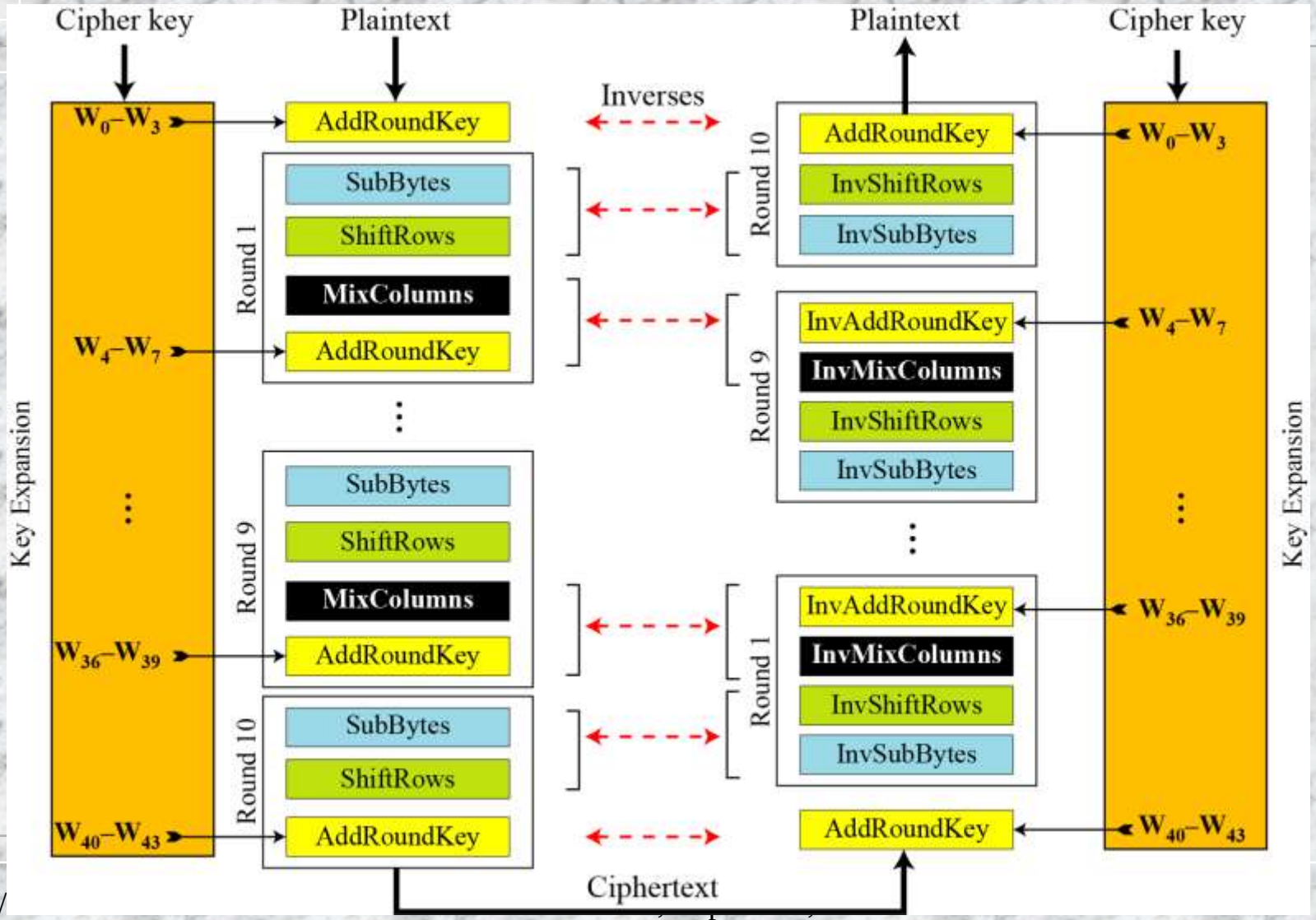
- These two groups are inverse of each other when 'InvAddRoundKey' is obtained by multiplying the key matrix with inverse of the constant matrix used in MixColumn



Cipher: $T = (CS) \oplus K$

Inverse cipher: $C^{-1} T \oplus C^{-1} K = C^{-1} (CS \oplus K) \oplus C^{-1} K = C^{-1}CS \oplus C^{-1}K \oplus C^{-1} K = S$

Cipher and reverse cipher in alternate design



Examples

- The following shows the ciphertext block created from a plaintext block using a randomly selected cipher key

Plaintext:	00	04	12	14	12	04	12	00	0C	00	13	11	08	23	19	19
Cipher Key:	24	75	A2	B3	34	75	56	88	31	E2	12	00	13	AA	54	87
Ciphertext:	BC	02	8B	D3	E0	E3	B1	95	55	0D	6D	FB	E6	F1	82	41

Examples (contd...)

Table 7.7 *Example of encryption*

Round	*Input State*	*Output State*	*Round Key*
Pre-round	00 12 0C 08	24 26 3D 1B	24 34 31 13
	04 04 00 23	71 71 E2 89	75 75 E2 AA
	12 12 13 19	B0 44 01 4D	A2 56 12 54
	14 00 11 19	A7 88 11 9E	B3 88 00 87
1	24 26 3D 1B	6C 44 13 BD	89 BD 8C 9F
	71 71 E2 89	B1 9E 46 35	55 20 C2 68
	B0 44 01 4D	C5 B5 F3 02	B5 E3 F1 A5
	A7 88 11 9E	5D 87 FC 8C	CE 46 46 C1
2	6C 44 13 BD	1A 90 15 B2	CE 73 FF 60
	B1 9E 46 35	66 09 1D FC	53 73 B1 D9
	C5 B5 F3 02	20 55 5A B2	CD 2E DF 7A
	5D 87 FC 8C	2B CB 8C 3C	15 53 15 D4

Examples (contd...)

3	1A 90 15 B2 66 09 1D FC 20 55 5A B2 2B CB 8C 3C	F6 7D A2 B0 1B 61 B4 B8 67 09 C9 45 4A 5C 51 09	FF 8C 73 13 89 FA 4B 92 85 AB 74 0E C5 96 83 57
4	F6 7D A2 B0 1B 61 B4 B8 67 09 C9 45 4A 5C 51 09	CA E5 48 BB D8 42 AF 71 D1 BA 98 2D 4E 60 9E DF	B8 34 47 54 22 D8 93 01 DE 75 01 0F B8 2E AD FA
5	CA E5 48 BB D8 42 AF 71 D1 BA 98 2D 4E 60 9E DF	90 35 13 60 2C FB 82 3A 9E FC 61 ED 49 39 CB 47	D4 E0 A7 F3 54 8C 1F 1E F3 86 87 88 98 B6 1B E1
6	90 35 13 60 2C FB 82 3A 9E FC 61 ED 49 39 CB 47	18 0A B9 B5 64 68 6A FB 5A EF D7 79 8E B2 10 4D	86 66 C1 32 90 1C 03 1D 0B 8D 0A 82 95 23 38 D9

Examples (contd...)

7	18 0A B9 B5 64 68 6A FB 5A EF D7 79 8E B2 10 4D	01 63 F1 96 55 24 3A 62 F4 8A DE 4D CC BA 88 03	62 04 C5 F7 83 9F 9C 81 3E B3 B9 3B B6 95 AD 74
8	01 63 F1 96 55 24 3A 62 F4 8A DE 4D CC BA 88 03	2A 34 D8 46 2D 6B A2 D6 51 64 CF 5A 87 A8 F8 28	EE EA 2F D8 61 FE 62 E3 AC 1F A6 9D DE 4B E6 92
9	2A 34 D8 46 2D 6B A2 D6 51 64 CF 5A 87 A8 F8 28	0A D9 F1 3C 95 63 9F 35 2A 80 29 00 16 76 09 77	E4 0E 21 F9 3F C1 A3 40 E3 FC 5A C7 BF F4 12 80
10	0A D9 F1 3C 95 63 9F 35 2A 80 29 00 16 76 09 77	BC E0 55 E6 02 E3 0D F1 8B B1 6D 82 D3 95 F8 41	DB D5 F4 0D F9 38 9B DB 2E D2 88 4F 26 D2 C0 40

Example

- One may be curious to see the result of encryption when the plaintext is made of all 0s

Plaintext:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
Cipher Key:	24	75	A2	B3	34	75	56	88	31	E2	12	00	13	AA	54	87
Ciphertext:	63	2C	D4	5E	5D	56	ED	B5	62	04	01	A0	AA	9C	2D	8D

Example

- Let us check the avalanche effect
- Let us change only one bit in the plaintext and compare the results.

Plaintext 1:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
Plaintext 2:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	01
Ciphertext 1:	63	2C	D4	5E	5D	56	ED	B5	62	04	01	A0	AA	9C	2D	8D
Ciphertext 2:	26	F3	9B	BC	A1	9C	0F	B7	C7	2E	7E	30	63	92	73	13

- We changed only one bit in the last byte. The result clearly shows the effect of diffusion and confusion. Changing a single bit in the plaintext has affected many bits in the ciphertext

Example

- The following shows the effect of using a cipher key in which all bits are 0s.

Plaintext:	00	04	12	14	12	04	12	00	0c	00	13	11	08	23	19	19
Cipher Key:	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
Ciphertext:	5A	6F	4B	67	57	B7	A5	D2	C4	30	91	ED	64	9A	42	72

ANALYSIS OF AES

- Topics to be covered
  - Security
  - Implementation
  - Simplicity and Cost

Security issue of AES

- *AES was designed after DES. Most of the known attacks on DES were already tested on AES.*
- *Brute-Force Attack*
  - *AES is definitely more secure than DES due to the larger-size key.*
- *Statistical Attacks*
  - *Numerous tests have failed to do statistical analysis of the ciphertext.*
- *Differential and Linear Attacks*
  - *There are no differential and linear attacks on AES as yet*

Implementation of AES

- *AES can be implemented in software, hardware, and firmware*
- *The implementation can use table lookup process or routines that use a well-defined algebraic structure*

Simplicity and Cost

- *The algorithms used in AES are so simple that they can be easily implemented using cheap processors and a minimum amount of memory*