# *Chapter 7*

# *Advanced Encryption Standard (AES)*

## *Objectives*

❏ *To review a short history of AES*

❏ *To define the basic structure of AES*

❏ *To define the transformations used by AES*

❏ *To define the key expansion process*

❏ *To discuss different implementations*

## 7-1   INTRODUCTION

*The Advanced Encryption Standard (AES) is a symmetric-key block cipher published by the National Institute of Standards and Technology (NIST) in December 2001.*

### Topics discussed in this section:

7.1.1   **History**
7.1.2   **Criteria**
7.1.3   **Rounds**
7.1.4   **Data Units**
7.1.5   **Structure of Each Round**

*In February 2001, NIST announced that a draft of the Federal Information Processing Standard (FIPS) was available for public review and comment. Finally, AES was published as FIPS 197 in the Federal Register in December 2001.*

*The criteria defined by NIST for selecting AES fall into three areas:*

*1. Security*

*2. Cost*

*3. Implementation.*

*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.*
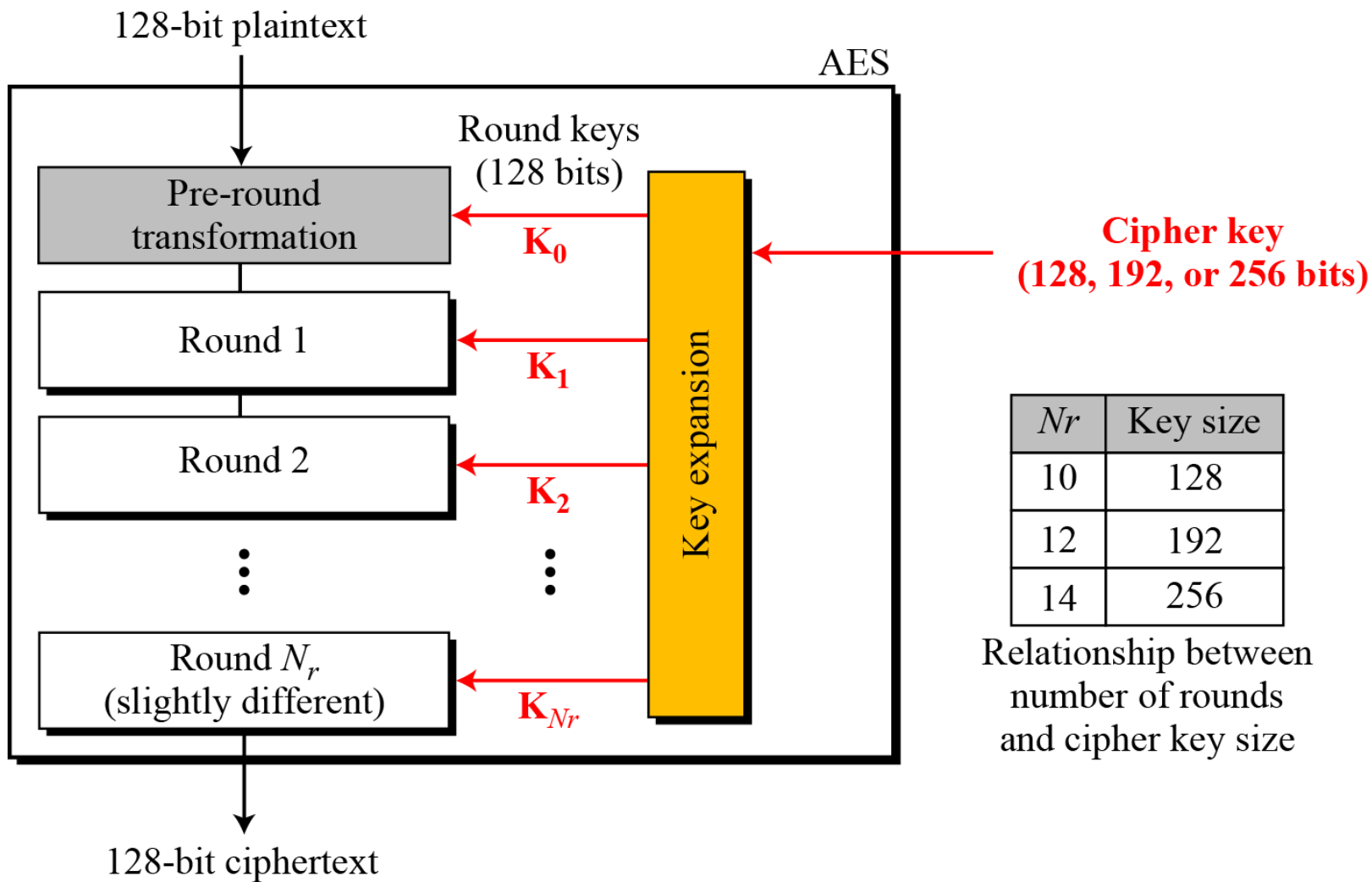
*Note*

**AES has defined three versions, with 10, 12, and 14 rounds.**

**Each version uses a different cipher key size (128, 192, or 256), but the round keys are always 128 bits.**

## Figure 7.1  *General design of AES encryption cipher*



| Nr | Key size |
|----|----------|
| 10 | 128 |
| 12 | 192 |
| 14 | 256 |

Relationship between number of rounds and cipher key size

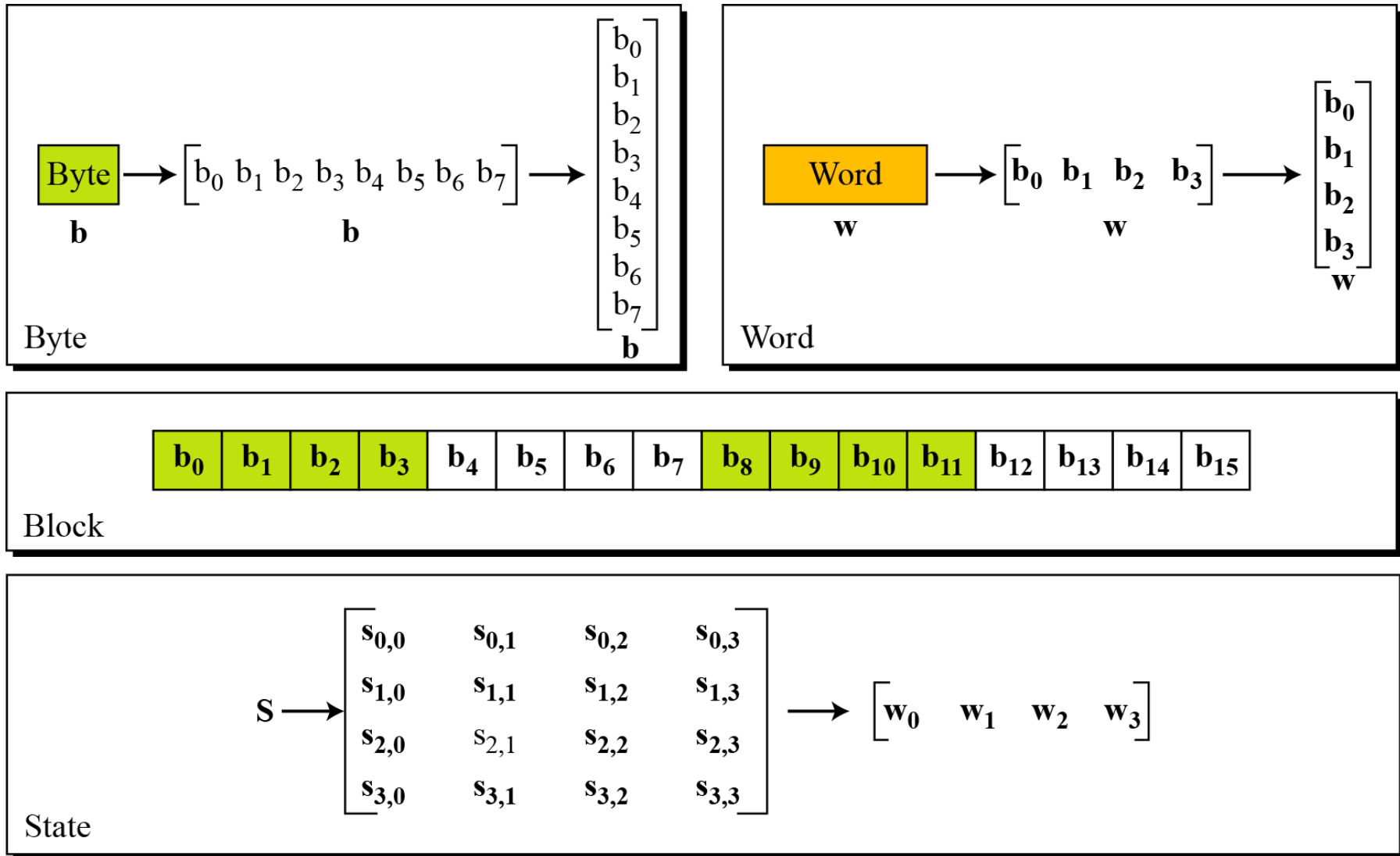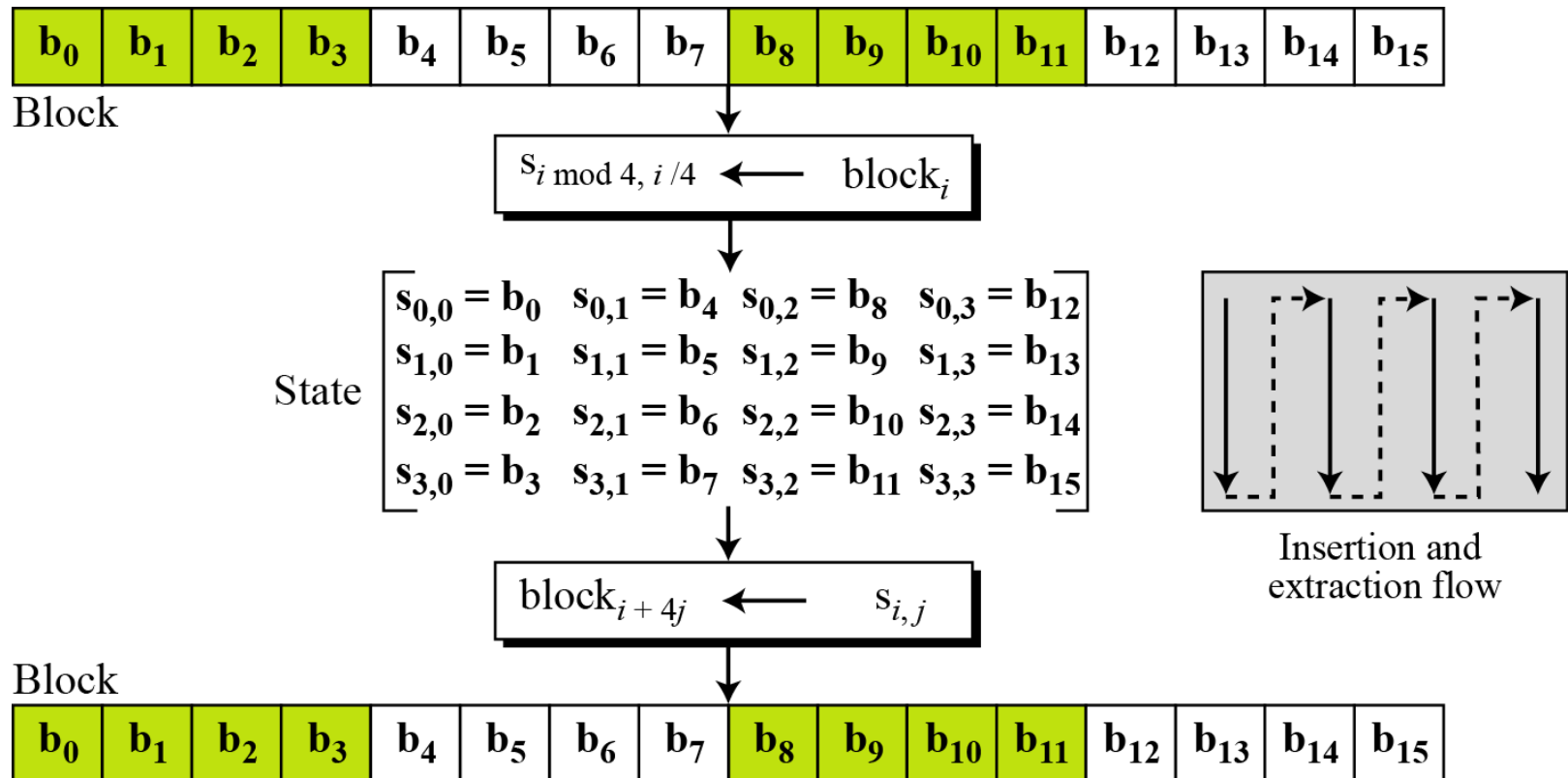## Figure 7.2  Data units used in AES

**Figure 7.3**  *Block-to-state and state-to-block transformation*

*Example 7.1*  *Continue*

*Figure 7.4*  *Changing plaintext to state*

| Text | A | E | S | U | S | E | S | A | M | A | T | R | I | X | Z | Z |
|------|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Hexadecimal | 00 | 04 | 12 | 14 | 12 | 04 | 12 | 00 | 0C | 00 | 13 | 11 | 08 | 23 | 19 | 19 |

$$\begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 23 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix} \text{State}$$

### Figure 7.5  Structure of each round at the encryption site



**Notes:**

1. One AddRoundKey is applied before the first round.

2. The third transformation is missing in the last round.
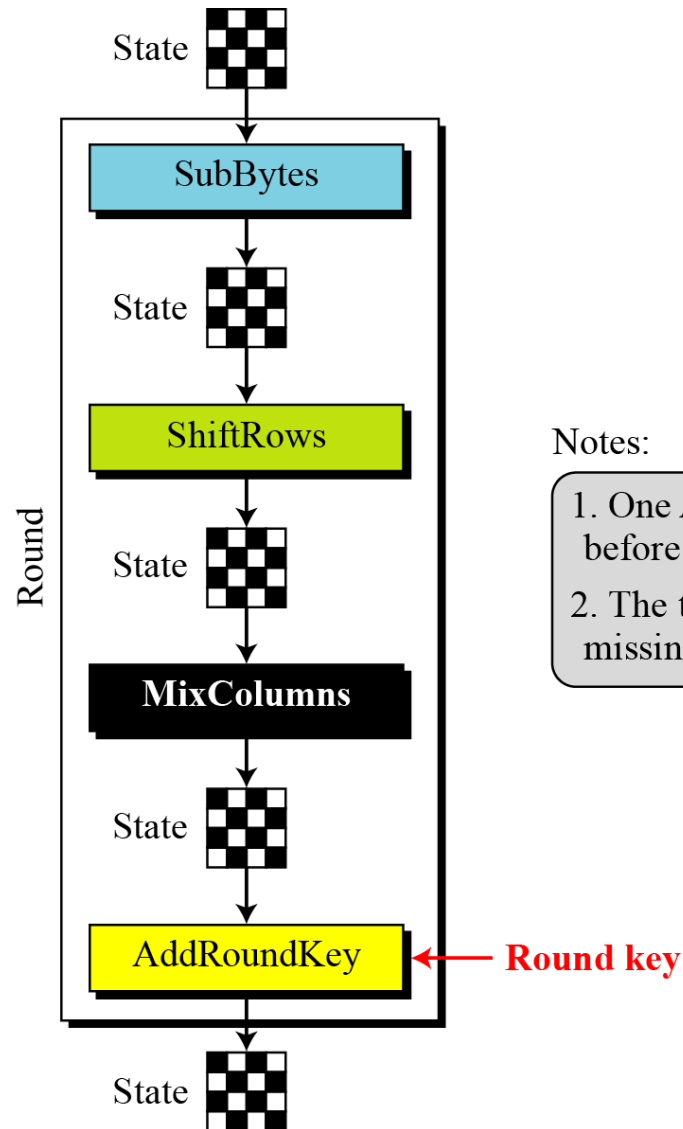
*To provide security, AES uses four types of transformations: substitution, permutation, mixing, and key-adding.*

**Topics discussed in this section:**

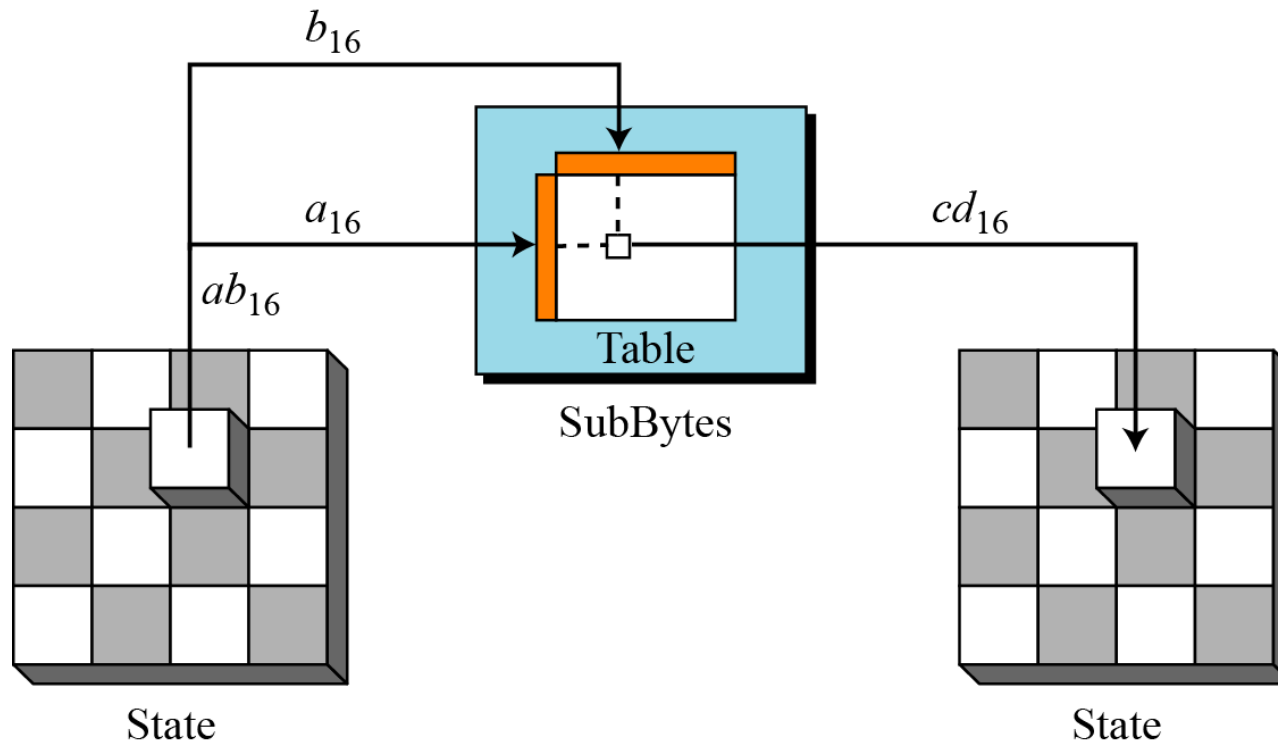**AES, like DES, uses substitution. AES uses two invertible transformations.**

**SubBytes**

**The first transformation, SubBytes, is used at the encryption site. To substitute a byte, we interpret the byte as two hexadecimal digits.**

**Note**

**The SubBytes operation involves 16 independent byte-to-byte transformations.**

## *Figure 7.6*  *SubBytes transformation*

**Table 7.1** *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 |

**Table 7.1** *SubBytes transformation table (continued)*

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

# *InvSubBytes*

**Table 7.2**   *InvSubBytes transformation table*

|   | 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 |

# InvSubBytes *(Continued)*

|   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

## Example 7.2

*Figure 7.7 shows how a state is transformed using the SubBytes transformation. The figure also shows that the InvSubBytes transformation creates the original one. Note that if the two bytes have the same values, their transformation is also the same.*

**Figure 7.7**  *SubBytes transformation for Example 7.2*
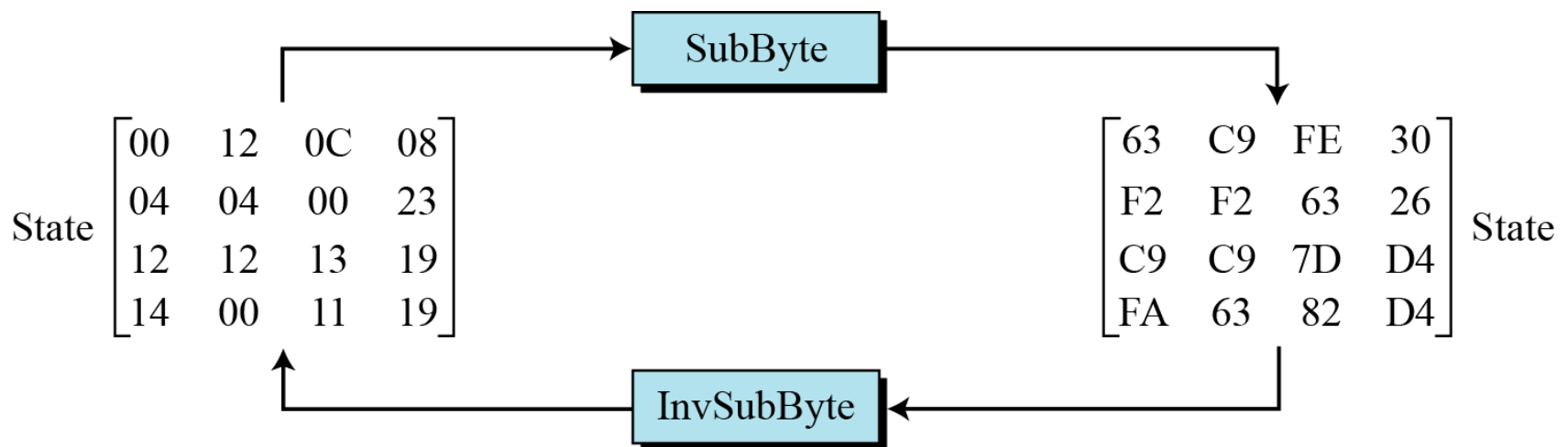
## *Transformation Using the GF(2⁸) Field*

*AES also defines the transformation algebraically using the GF(2⁸) field with the irreducible polynomials ($x^8 + x^4 + x^3 + x + 1$), as shown in Figure 7.8.*

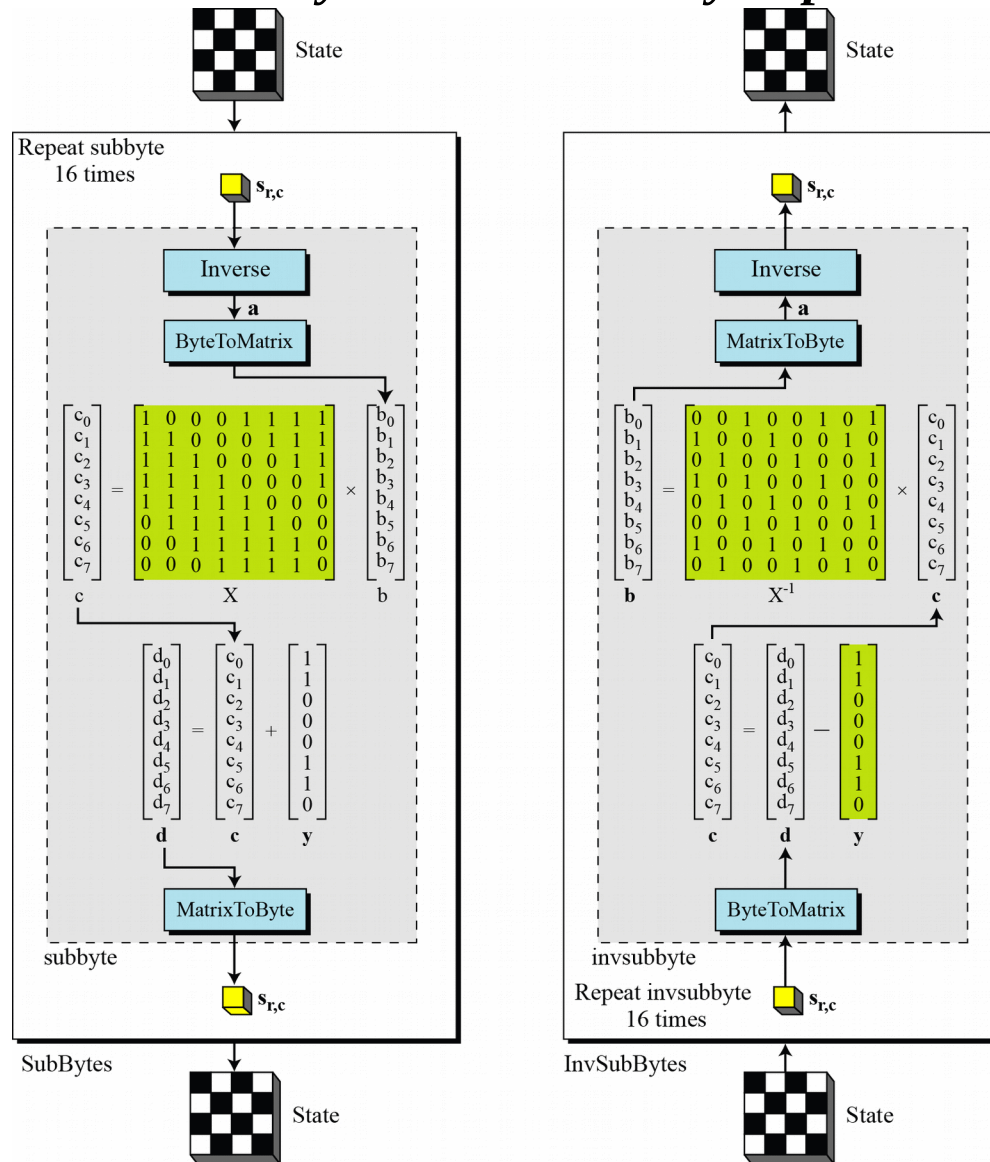subbyte: $\rightarrow$ $\mathbf{d} = \mathbf{X}\,(s_{r,c})^{-1} \oplus \mathbf{y}$

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

**Note**

**The SubBytes and InvSubBytes transformations are inverses of each other.**

## Figure 7.8  SubBytes and InvSubBytes processes

**Example 7.3**

**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.

**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})

}
```
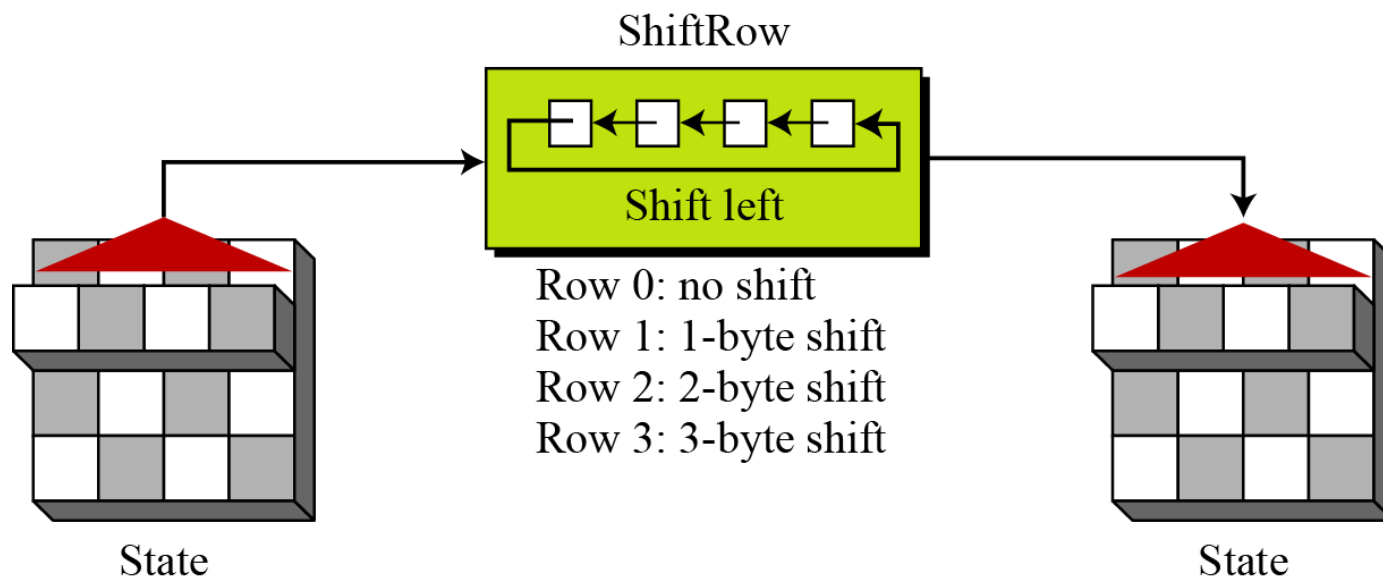
subbyte (**byte**)

$$a \leftarrow byte^{-1} \qquad \textit{// Multiplicative inverse in GF($2^8$) with inverse of 00 to be 00}$$

ByteToMatrix (a, **b**)

for (i = 0 to 7)

{

$$c_i \leftarrow b_i \oplus b_{(i+4)\bmod 8} \oplus b_{(i+5)\bmod 8} \oplus b_{(i+6)\bmod 8} \oplus b_{(i+7)\bmod 8}$$

$$d_i \leftarrow c_i \oplus ByteToMatrix\ (0x63)$$

}

MatrixToByte (**d**, d)

byte $\leftarrow$ d

}

*Another transformation found in a round is shifting, which permutes the bytes.*

*ShiftRows*

*In the encryption, the transformation is called ShiftRows.*

*Figure 7.9* *ShiftRows transformation*



ShiftRow

Shift left

Row 0: no shift
Row 1: 1-byte shift
Row 2: 2-byte shift
Row 3: 3-byte shift

State

State

## *InvShiftRows*

**In the decryption, the transformation is called InvShiftRows and the shifting is to the right.**

**Algorithm 7.2**   *Pseudocode for ShiftRows transformation*

**ShiftRows (S)**

{

    for ($r$ = 1 to 3)

        shiftrow ($s_r$, $r$)                    // $s_r$ *is the rth row*
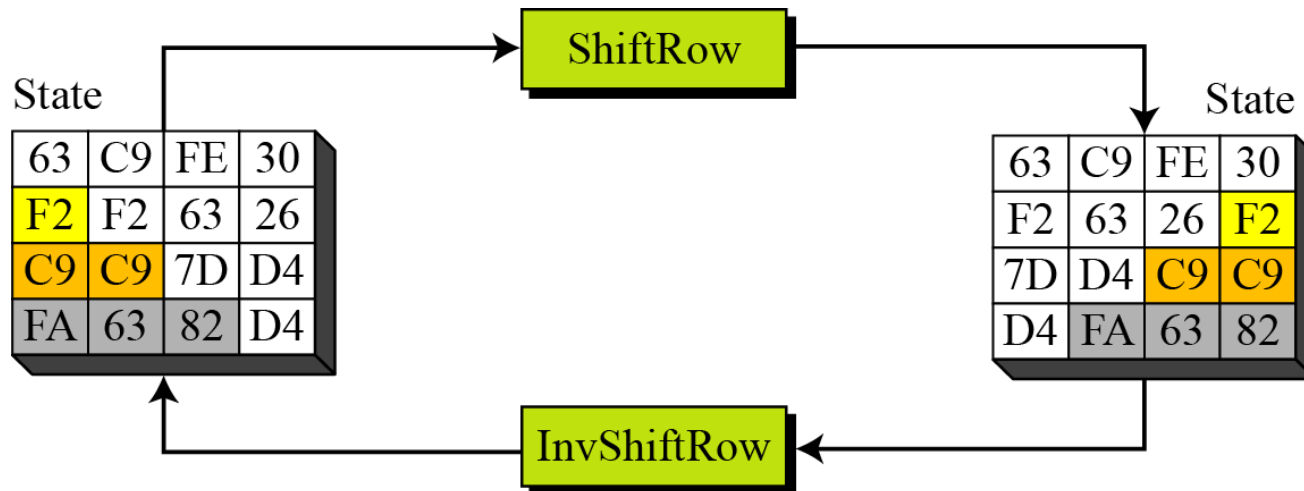
}

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**$_{(c-n) \bmod 4}$ ← **t**$_c$

}

**Example 7.4**

*Figure 7.10 shows how a state is transformed using ShiftRows transformation. The figure also shows that InvShiftRows transformation creates the original state.*

*Figure 7.10* *ShiftRows transformation in Example 7.4*

*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.*

**Figure 7.11**  *Mixing bytes using matrix multiplication*

$$a\mathbf{x} + b\mathbf{y} + c\mathbf{z} + d\mathbf{t}$$
$$e\mathbf{x} + f\mathbf{y} + g\mathbf{z} + h\mathbf{t}$$
$$i\mathbf{x} + j\mathbf{y} + k\mathbf{z} + l\mathbf{t}$$
$$m\mathbf{x} + n\mathbf{y} + o\mathbf{z} + p\mathbf{t}$$

$$= \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \times \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \\ \mathbf{z} \\ \mathbf{t} \end{bmatrix}$$

New matrix      **Constant matrix**      Old matrix

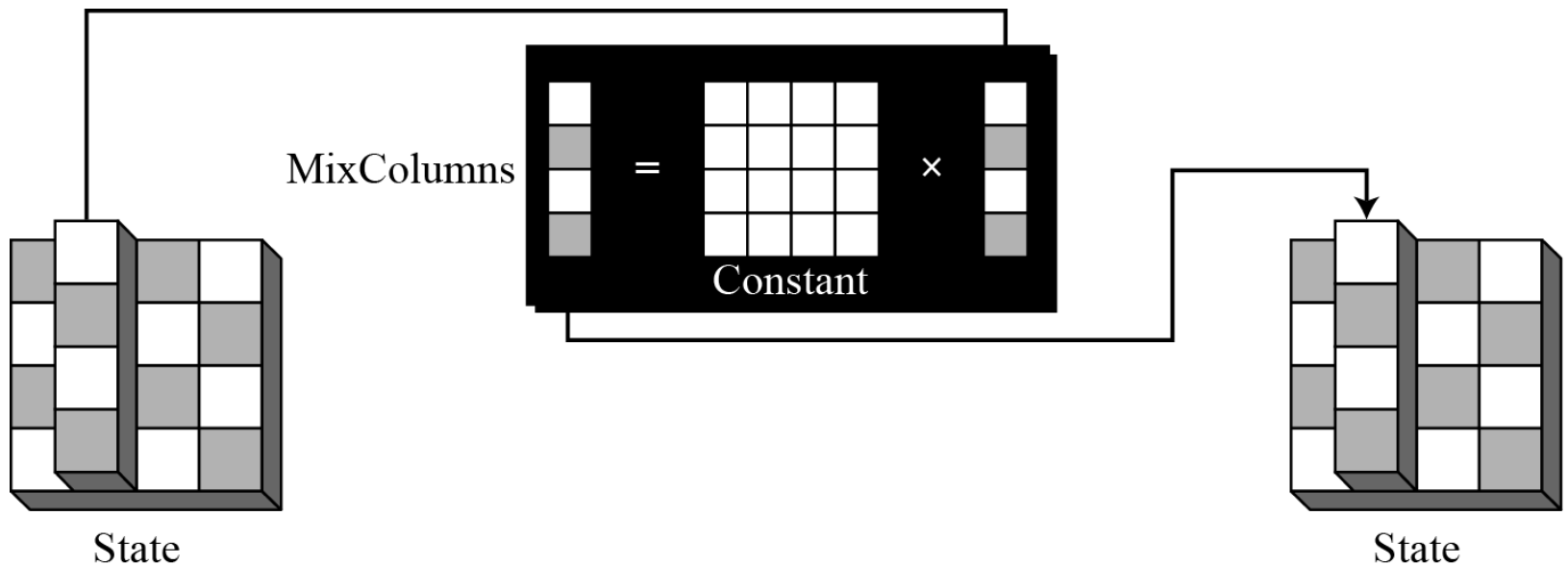**Figure 7.12** *Constant matrices used by MixColumns and InvMixColumns*

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

## *MixColumns*

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

*Figure 7.13*  *MixColumns transformation*

## *InvMixColumns*

*The InvMixColumns transformation is basically the same as the MixColumns transformation.*

> **Note**

**The MixColumns and InvMixColumns transformations are inverses of each other.**

**Algorithm 7.3**   *Pseudocode for MixColumns transformation*

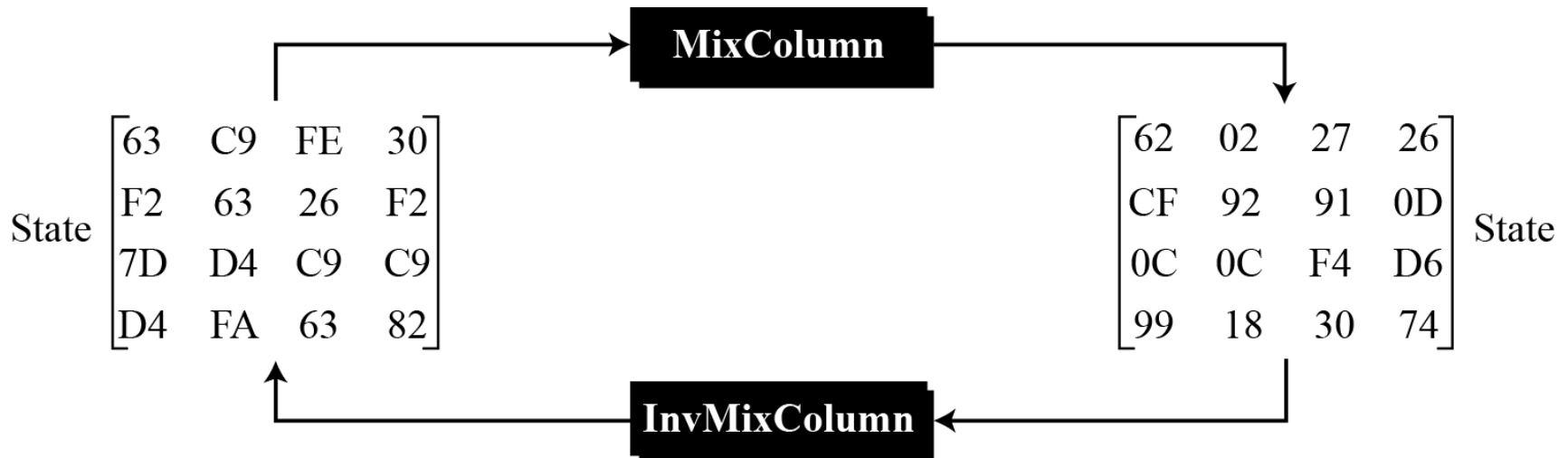**MixColumns (S)**

{

    for ($c$ = 0 to 3)

        mixcolumn ($s_c$)

}

**mixcolumn (col)**

{

  CopyColumn (**col**, **t**)                 // *t is a temporary column*

  $\textbf{col}_0 \leftarrow (0x02) \bullet \textbf{t}_0 \oplus (0x03 \bullet \textbf{t}_1) \oplus \textbf{t}_2 \oplus \textbf{t}_3$

  $\textbf{col}_1 \leftarrow \textbf{t}_0 \oplus (0x02) \bullet \textbf{t}_1 \oplus (0x03) \bullet \textbf{t}_2 \oplus \textbf{t}_3$

  $\textbf{col}_2 \leftarrow \textbf{t}_0 \oplus \textbf{t}_1 \oplus (0x02) \bullet \textbf{t}_2 \oplus (0x03) \bullet \textbf{t}_3$

  $\textbf{col}_3 \leftarrow (0x03 \bullet \textbf{t}_0) \oplus \textbf{t}_1 \oplus \textbf{t}_2 \oplus (0x02) \bullet \textbf{t}_3$

}

## Example 7.5

***Figure 7.14 shows how a state is transformed using the MixColumns transformation. The figure also shows that the InvMixColumns transformation creates the original one.***

**Figure 7.14** *The MixColumns transformation in Example 7.5*

MixColumn

$$
\text{State} \begin{bmatrix} 63 & C9 & FE & 30 \\ F2 & 63 & 26 & F2 \\ 7D & D4 & C9 & C9 \\ D4 & FA & 63 & 82 \end{bmatrix} \qquad \begin{bmatrix} 62 & 02 & 27 & 26 \\ CF & 92 & 91 & 0D \\ 0C & 0C & F4 & D6 \\ 99 & 18 & 30 & 74 \end{bmatrix} \text{State}
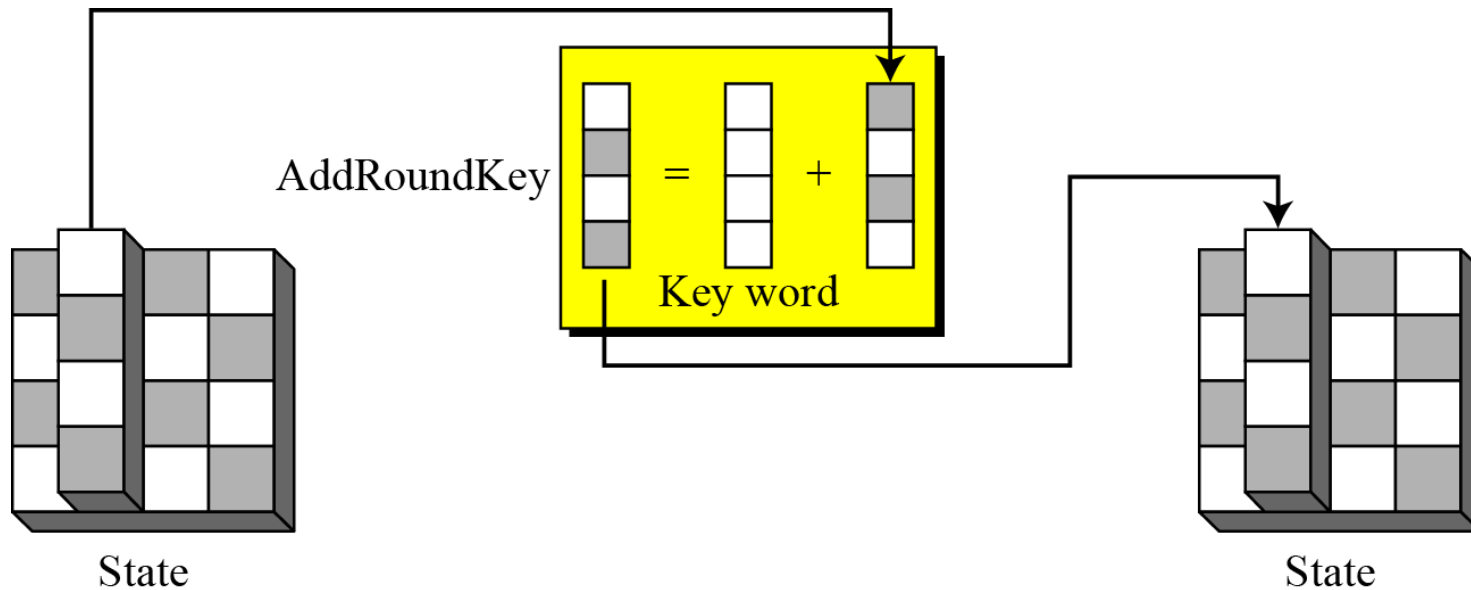$$

InvMixColumn

*AddRoundKey*

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

*Note*

**The AddRoundKey transformation is the inverse of itself.**

## Figure 7.15  AddRoundKey transformation



**Algorithm 7.4**   *Pseudocode for AddRoundKey transformation*

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

*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 128-bit cipher key.*

**Topics discussed in this section:**

**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 | $\mathbf{w}_8$ | $\mathbf{w}_9$ | $\mathbf{w}_{10}$ | $\mathbf{w}_{11}$ |
| . . . | . . . | | | |
| $N_r$ | $\mathbf{w}_{4N_r}$ | $\mathbf{w}_{4N_r+1}$ | $\mathbf{w}_{4N_r+2}$ | $\mathbf{w}_{4N_r+3}$ |

## Figure 7.16 Key expansion in AES



Making of t$_i$ (temporary) words $i = 4 N_r$
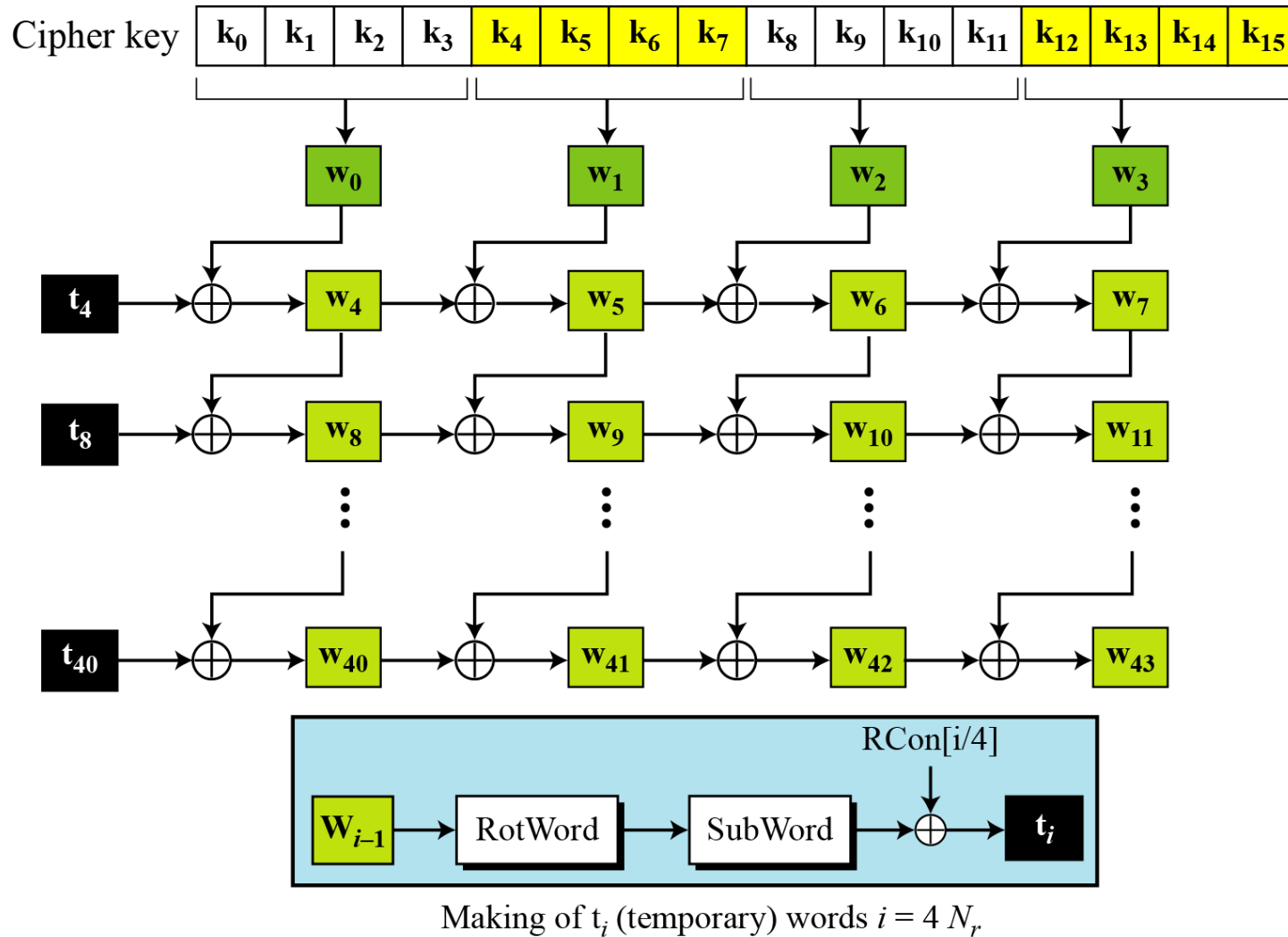
**Table 7.4**  *RCon constants*

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

***The key-expansion routine can either use the above table when calculating the words or use the GF(2⁸) field to calculate the leftmost byte dynamically, as shown below (prime is the irreducible polynomial):***

$RC_1 \rightarrow x^{1-1} = x^0 \quad \text{mod } prime \quad = 1 \quad\quad \rightarrow 00000001 \quad \rightarrow 01_{16}$

$RC_2 \rightarrow x^{2-1} = x^1 \quad \text{mod } prime \quad = x \quad\quad \rightarrow 00000010 \quad \rightarrow 02_{16}$

$RC_3 \rightarrow x^{3-1} = x^2 \quad \text{mod } prime \quad = x^2 \quad\quad \rightarrow 00000100 \quad \rightarrow 04_{16}$

$RC_4 \rightarrow x^{4-1} = x^3 \quad \text{mod } prime \quad = x^3 \quad\quad \rightarrow 00001000 \quad \rightarrow 08_{16}$

$RC_5 \rightarrow x^{5-1} = x^4 \quad \text{mod } prime \quad = x^4 \quad\quad \rightarrow 00010000 \quad \rightarrow 10_{16}$

$RC_6 \rightarrow x^{6-1} = x^5 \quad \text{mod } prime \quad = x^5 \quad\quad \rightarrow 00100000 \quad \rightarrow 20_{16}$

$RC_7 \rightarrow x^{7-1} = x^6 \quad \text{mod } prime \quad = x^6 \quad\quad \rightarrow 01000000 \quad \rightarrow 40_{16}$

$RC_8 \rightarrow x^{8-1} = x^7 \quad \text{mod } prime \quad = x^7 \quad\quad \rightarrow 10000000 \quad \rightarrow 80_{16}$

$RC_9 \rightarrow x^{9-1} = x^8 \quad \text{mod } prime \quad = x^4 + x^3 + x + 1 \quad \rightarrow 00011011 \quad \rightarrow 1B_{16}$

$RC_{10} \rightarrow x^{10-1} = x^9 \quad \text{mod } prime \quad = x^5 + x^4 + x^2 + x \quad \rightarrow 00110110 \quad \rightarrow 36_{16}$

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

**KeyExpansion** ([key$_0$ to key$_{15}$], [**w$_0$ to w$_{43}$**])
{

    for ($i$ = 0 to 3)
        **w**$_i$ ← key$_{4i}$ + key$_{4i+1}$ + key$_{4i+2}$ + key$_{4i+3}$

    for ($i$ = 4 to 43)
    {

      if ($i$ mod 4 ≠ 0)    **w**$_i$ ← **w**$_{i-1}$ + **w**$_{i-4}$

      else
      {

        **t** ← SubWord (RotWord (**w**$_{i-1}$)) ⊕ RCon$_{i/4}$      // **t** *is a temporary word*
        **w**$_i$ ← **t** + **w**$_{i-4}$

      }

    }

}

**Example 7.6**

**Table 7.5 shows how the keys for each round are calculated assuming that the 128-bit cipher key agreed upon by Alice and Bob is (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 | $w_{40} = DBF92E26$ | $w_{41} = D538D2D2$ | $w_{42} = F49B88C0$ | $w_{43} = 0DDB4F40$ |

*Example 7.7*

**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.**

*Example 7.8*

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

Cipher Key 1: 12 45 A2 A1 23 31 A4 A3   B2 CC A**A** 34   C2 BB 77 23
Cipher Key 2: 12 45 A2 A1 23 31 A4 A3   B2 CC A**B** 34   C2 BB 77 23

## Example 7.8  Continue

**Table 7.6**  *Comparing two sets of round keys*

| R. | Round keys for set 1 | Round keys for set 2 | B. D. |
|----|----------------------|----------------------|-------|
| — | 1245A2A1 2331A4A3 B2CCA<u>A</u>34 C2BB7723 | 1245A2A1 2331A4A3 B2CCA<u>B</u>34 C2BB7723 | 01 |
| 1 | F9B08484 DA812027 684D8<u>A</u>13 AAF6F<u>D</u>30 | F9B08484 DA812027 684D8<u>B</u>13 AAF6F<u>C</u>30 | 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 7.9

*The concept of weak keys, as we discussed for DES in INS class (+ Chapter 6 of the book), 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 algorithms in AES-192 and AES-256 versions are very similar to the key expansion algorithm in AES-128.*

**1. In AES-192, the words are generated in groups of six instead of four.**

 **a. The cipher key creates the first six words ($w_0$ to $w_5$).**

 **b. If i mod 6 ≠ 0,  $w_i = w_{i-1} + w_{i-6}$; otherwise $w_i = t + w_{i-6}$**

*2. In AES-256, the words are generated in groups of eight instead of four.*

   *a. the cipher key creates the first eight words ($w_0$ to $w_7$).*

   *b. If i mod 8 ≠ 0, $w_i = w_{i-1} + w_{i-8}$; otherwise $w_i = t + w_{i-8}$.*

   *c. If i mod 4 = 0, but i mod 8 ≠ 0, then $w_i = $ SubWord($w_{i-1}$) + $w_{i-8}$.*

*AES uses four types of transformations for encryption and decryption. In the standard, the encryption algorithm is referred to as the cipher and the decryption algorithm as the inverse cipher.*
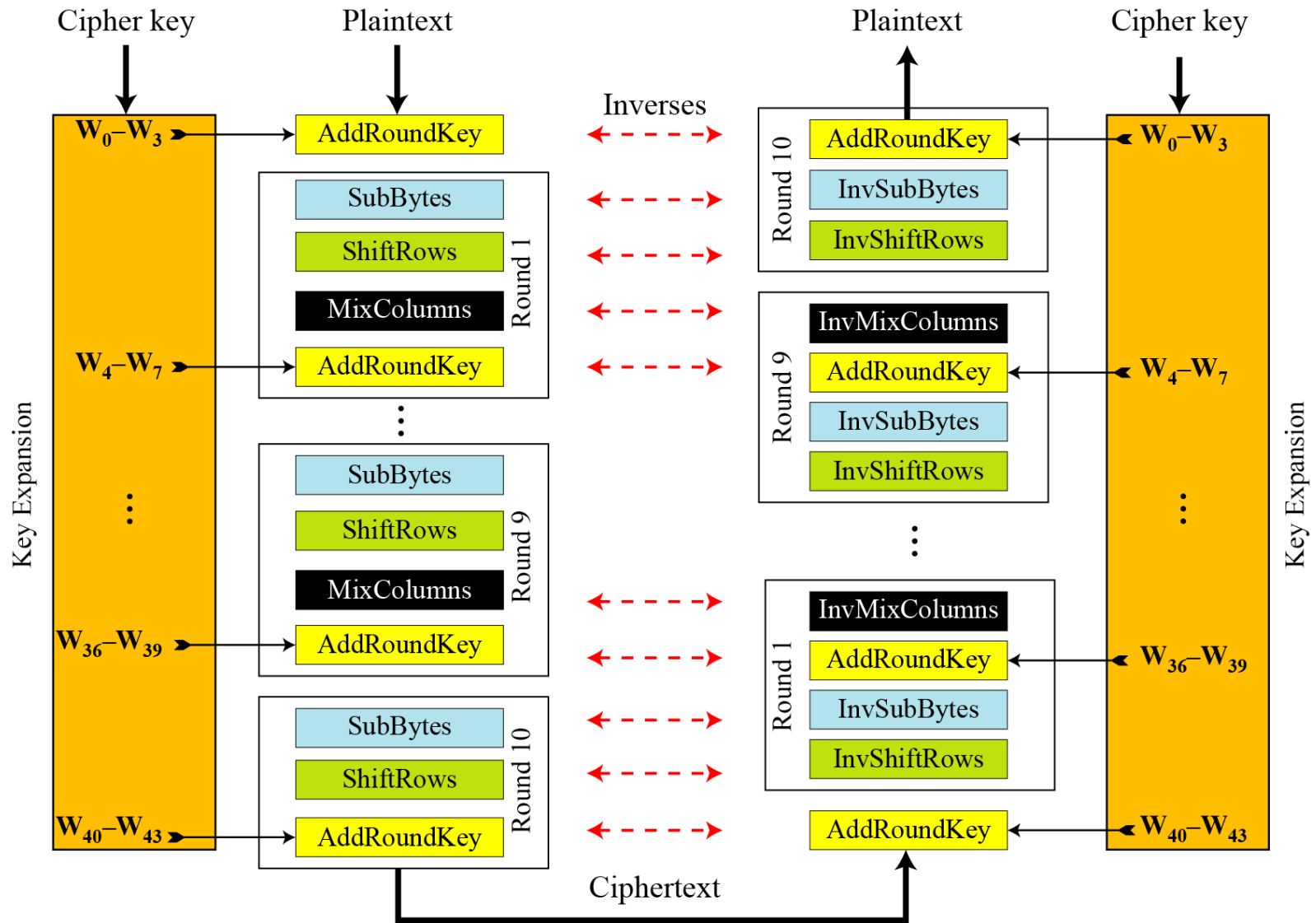
**Topics discussed in this section:**

## Figure 7.17  Ciphers and inverse ciphers of the original design

## *Algorithm*

## *The code for the AES-128 version of this design is shown in Algorithm 7.6.*

**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);
}
```

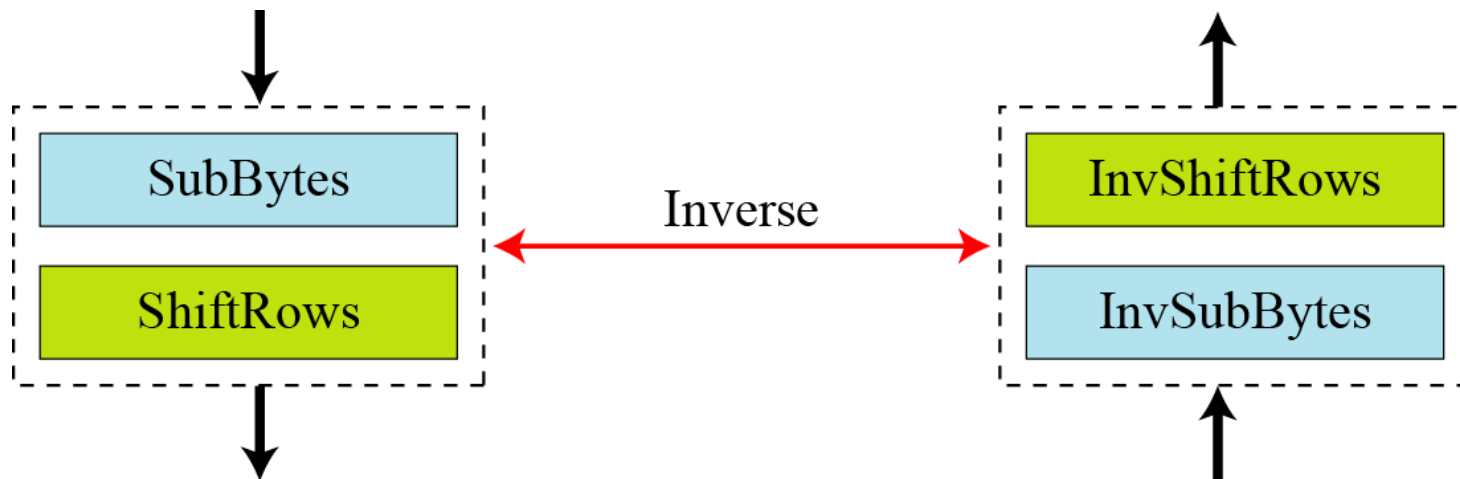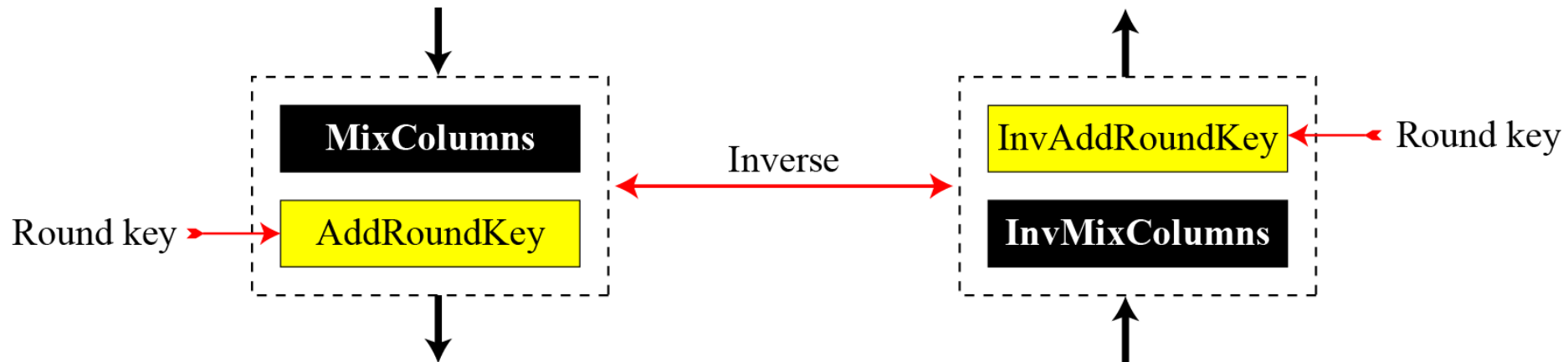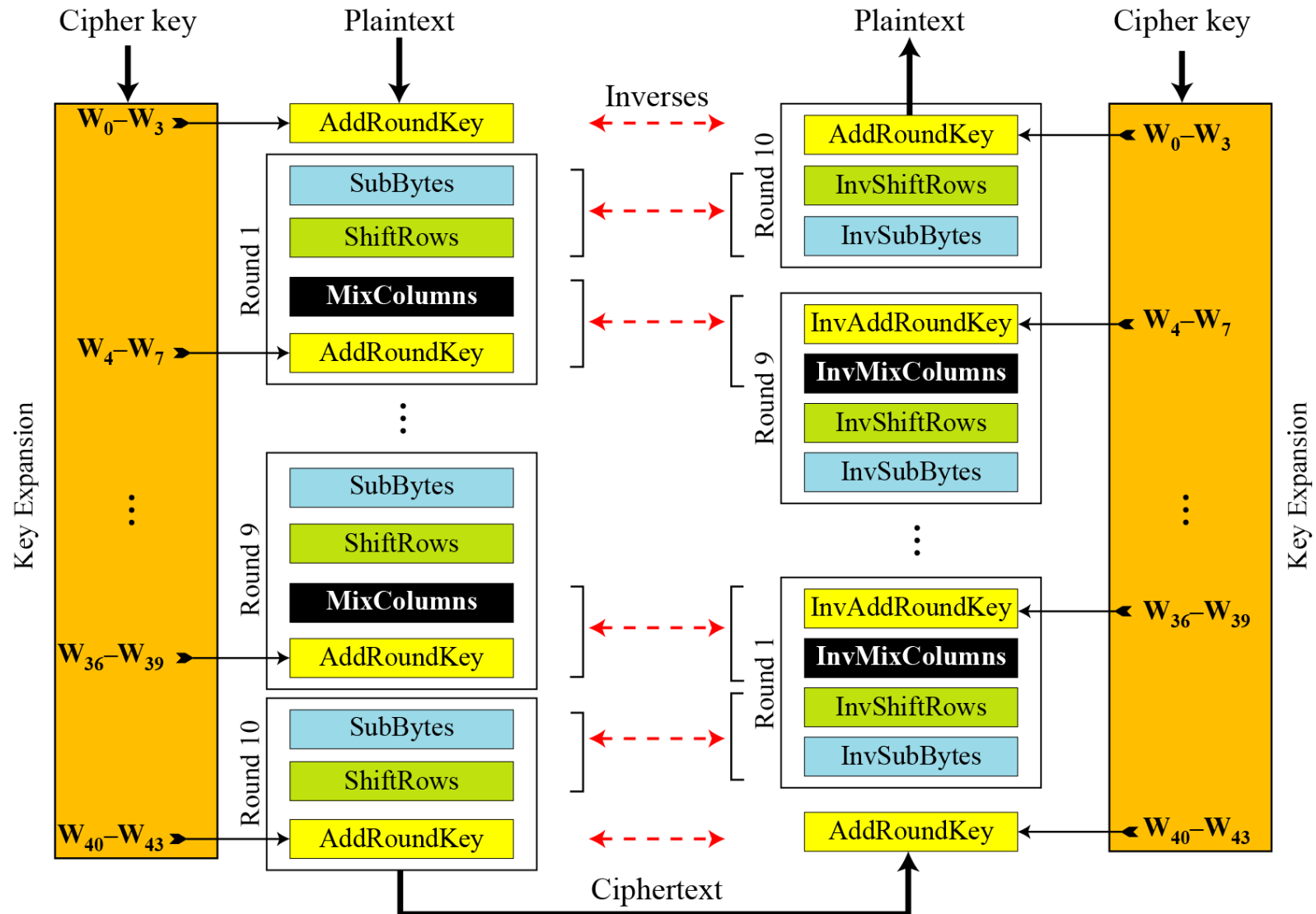## Figure 7.18  Invertibility of SubBytes and ShiftRows combinations

*Figure 7.19*  *Invertibility of MixColumns and AddRoundKey combination*

## Figure 7.20  Cipher and reverse cipher in alternate design

## *Changing Key-Expansion Algorithm*

*Instead of using InvRoundKey transformation in the reverse cipher, the key-expansion algorithm can be changed to create a different set of round keys for the inverse cipher.*

*In this section, some examples of encryption/ decryption and key generation are given to emphasize some points discussed in the two previous sections.*

**Example 7.10**

*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 |

### Example 7.10    Continued

**Table 7.7**    *Example of encryption*

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

## Example 7.10   Continued

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

## Example 7.10    Continued

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

**Example 7.11**

**Figure 7.21 shows the state entries in one round, round 7, in Example 7.10.**

*Figure 7.21*  **States in a single round**

| Input State | After SubBytes | After ShiftRows | After MixColumns | Output State (After RoundKey) |
|:---:|:---:|:---:|:---:|:---:|
| 18 0A B9 B5<br>64 68 6A FB<br>5A EF D7 79<br>8E B2 10 D4 | 7C FB A1 90<br>36 80 AA FC<br>1D E3 BF 7E<br>7B 4B F4 C4 | C4 DE F7 9E<br>4C 95 C0 35<br>FD 7B 69 C7<br>59 E3 1E BA | 2A 34 D8 46<br>2D 6B A2 D6<br>51 64 CF 5A<br>87 A8 F8 28 | 01 63 F1 96<br>55 24 3A 62<br>F4 8A DE 4D<br>CC BA 88 03 |
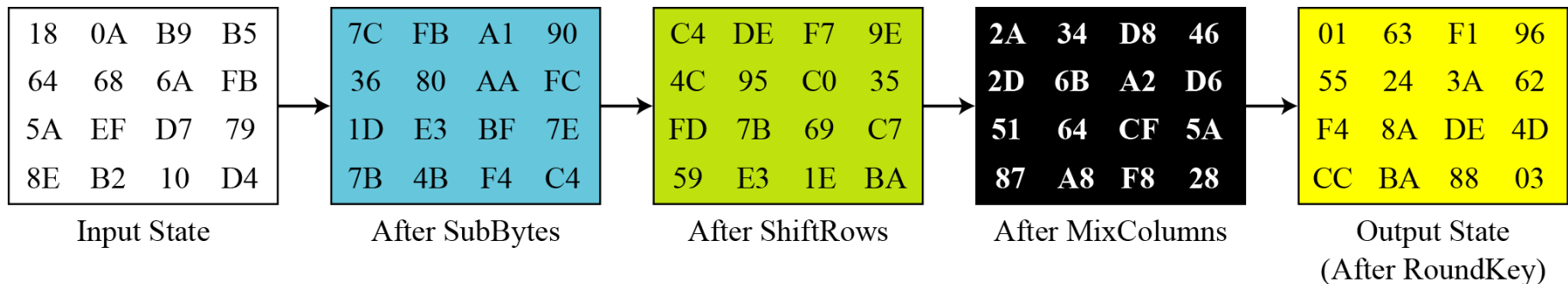
### Example 7.12

*One may be curious to see the result of encryption when the plaintext is made of all 0s. Using the cipher key in Example 7.10 yields the ciphertext.*

| 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 7.13

**Let us check the avalanche effect that we discussed in Chapter 6. Let us change only one bit in the plaintext and compare the results. 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.**

```
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
```

**Example 7.14**

*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
```

*This section is a brief review of the three characteristics of AES.*

**Topics discussed in this section:**

*7.6.1    Security*
*7.6.2    Implementation*
*7.6.3    Simplicity and Cost*

*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.***

*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.*

*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.*

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