CENG 471 Cryptography

Symmetrical Cryptosystems

Advanced Encryption Standard
(AES)

Asst. Prof. Dr. Serap ŞAHİN

Izmir Institute of Technology

***Reference**: Slides of B.Forouzan, “Cryptography and Network Security”*

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
- Security and Implementation

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.

History

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.

Criteria

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

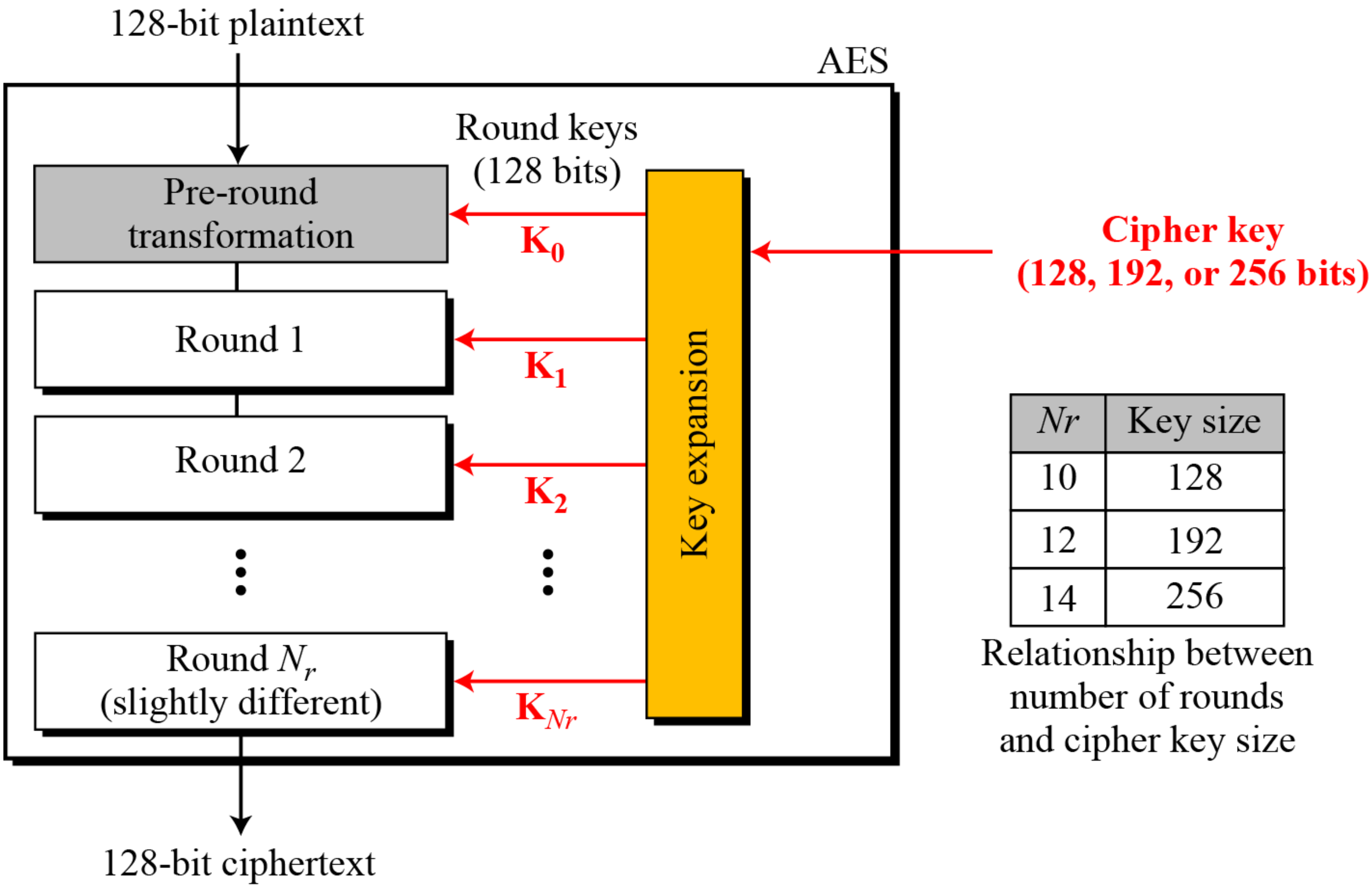
1. Security
2. Cost
3. Implementation.

Rounds

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.

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**.

General design of AES encryption cipher

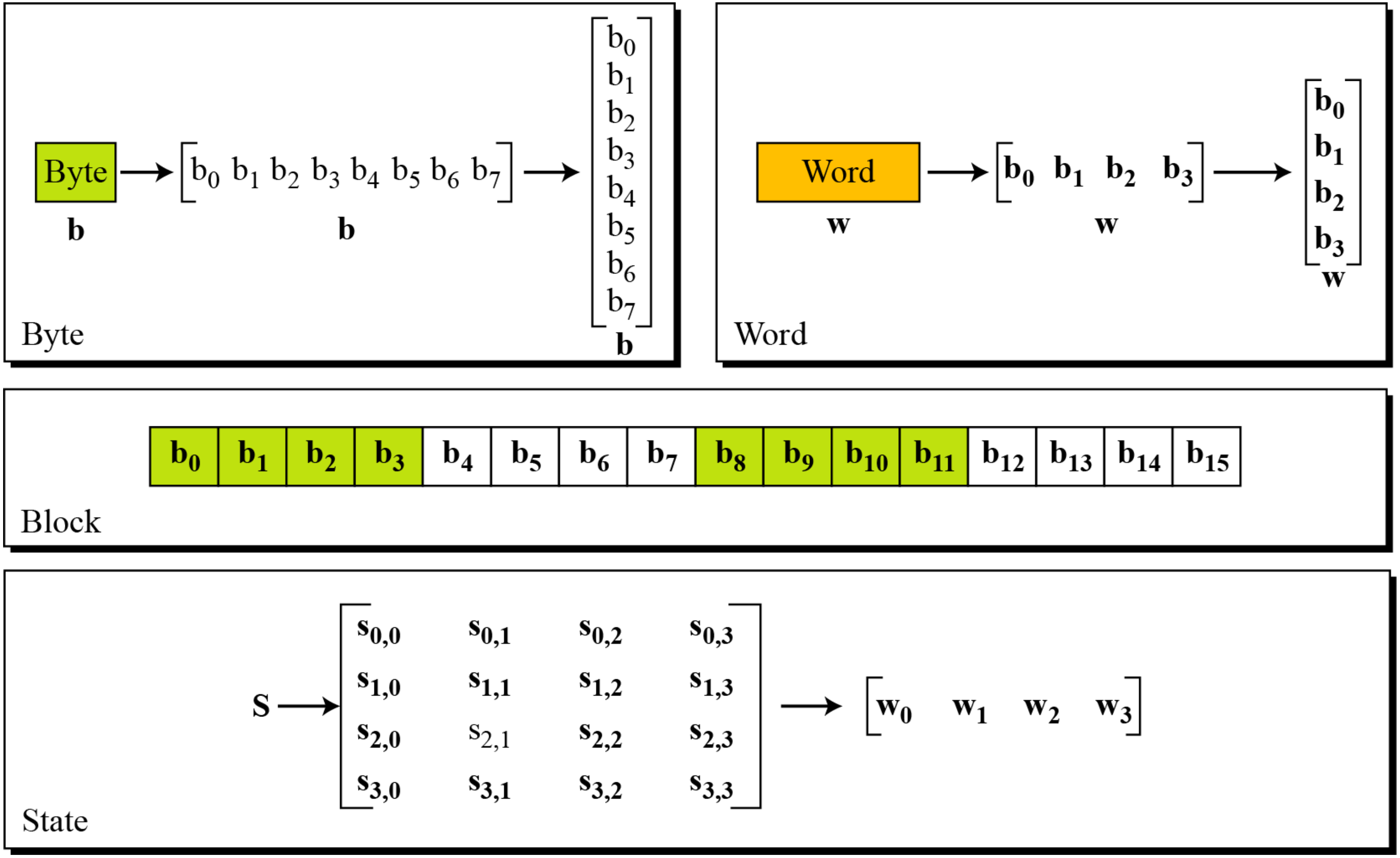


Nr	Key size
10	128
12	192
14	256

Relationship between
number of rounds
and cipher key size

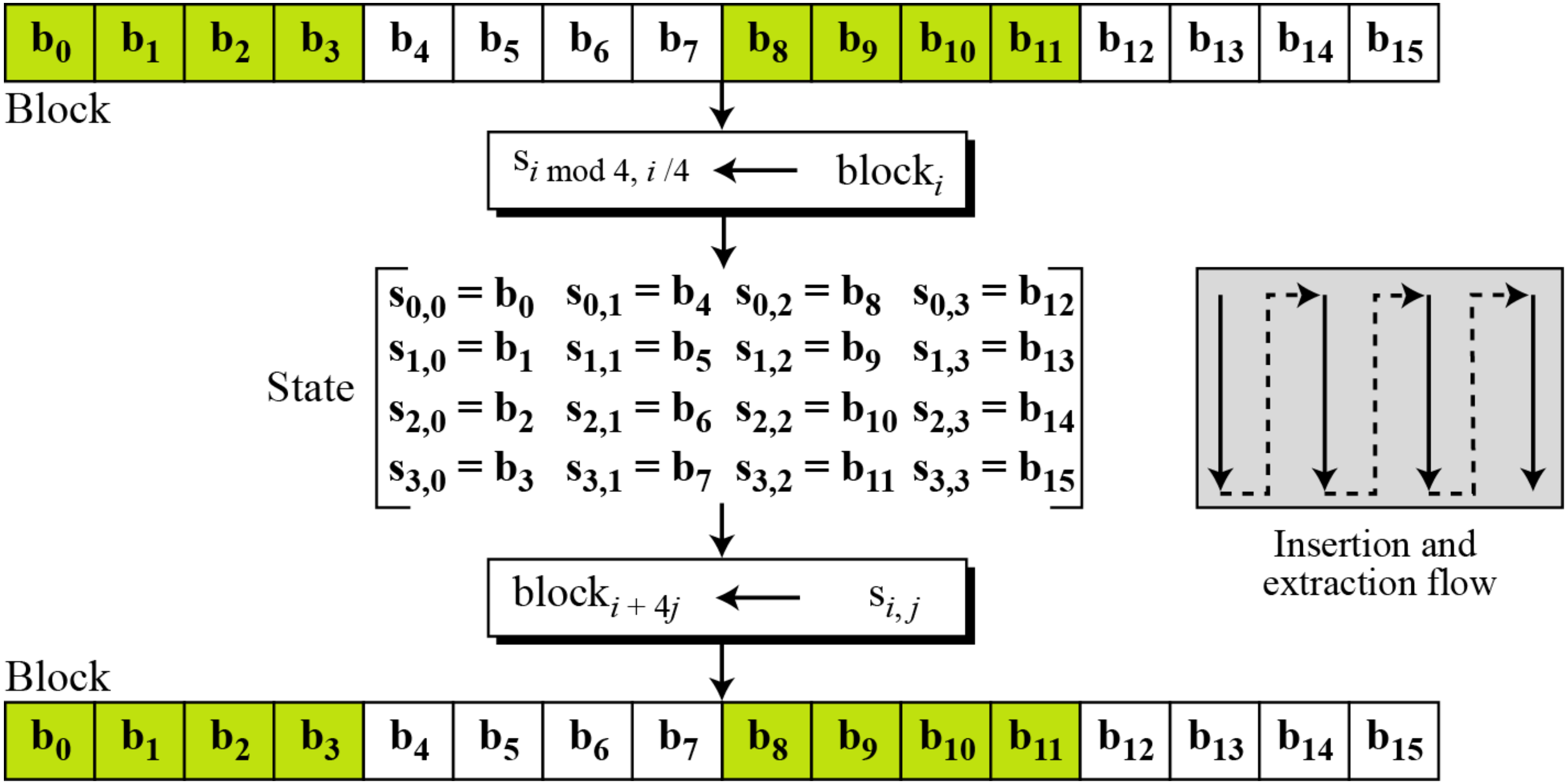
Data Units

Data units used in AES


Byte
b
$[b_0\ b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7]$
b
$[b_0\ b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7]^T$
b
Byte
Word
w
$[b_0\ b_1\ b_2\ b_3]$
w
$[b_0\ b_1\ b_2\ b_3]^T$
w
Word
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
$s_{0,0}$ $s_{0,1}$ $s_{0,2}$ $s_{0,3}$
$s_{1,0}$ $s_{1,1}$ $s_{1,2}$ $s_{1,3}$
$s_{2,0}$ $s_{2,1}$ $s_{2,2}$ $s_{2,3}$
$s_{3,0}$ $s_{3,1}$ $s_{3,2}$ $s_{3,3}$
$[w_0\ w_1\ w_2\ w_3]$
State

Data Units

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



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

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

State

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

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

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}

Insertion and extraction flow

Example 1

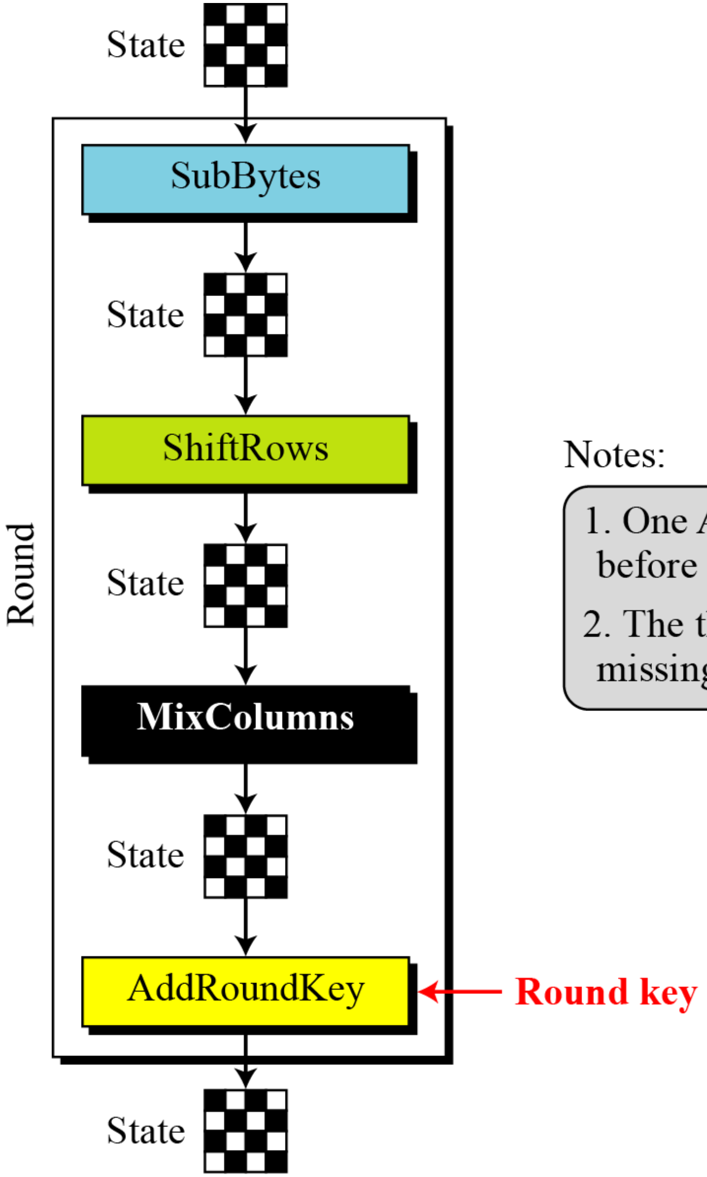
Changing plaintext to state

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

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

Structure of Each Round

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.

TRANSFORMATIONS

To provide security, AES uses four types of transformations:

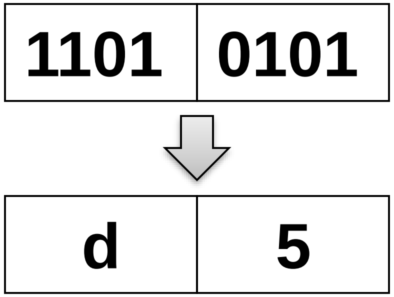
- substitution,
- permutation,
- mixing, and
- key-adding.

Substitution

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**.

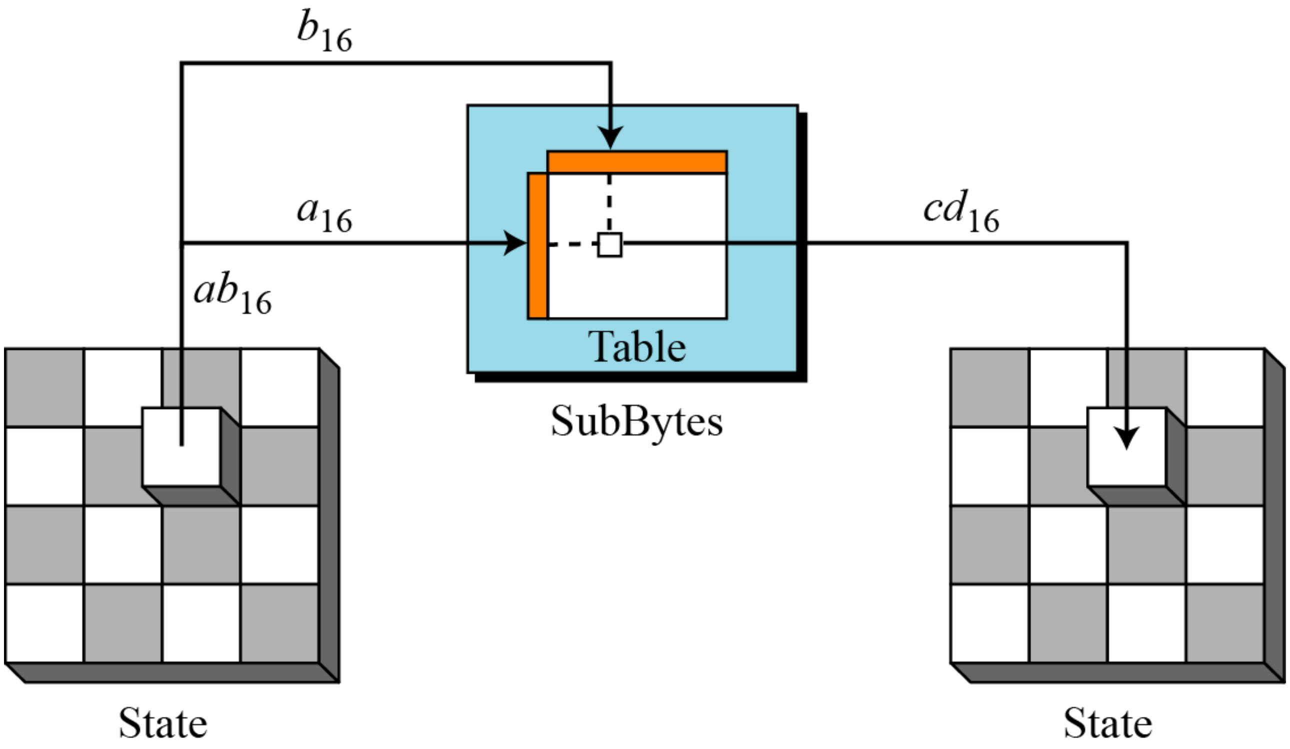


1101 0101

d 5

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

SubBytes transformation



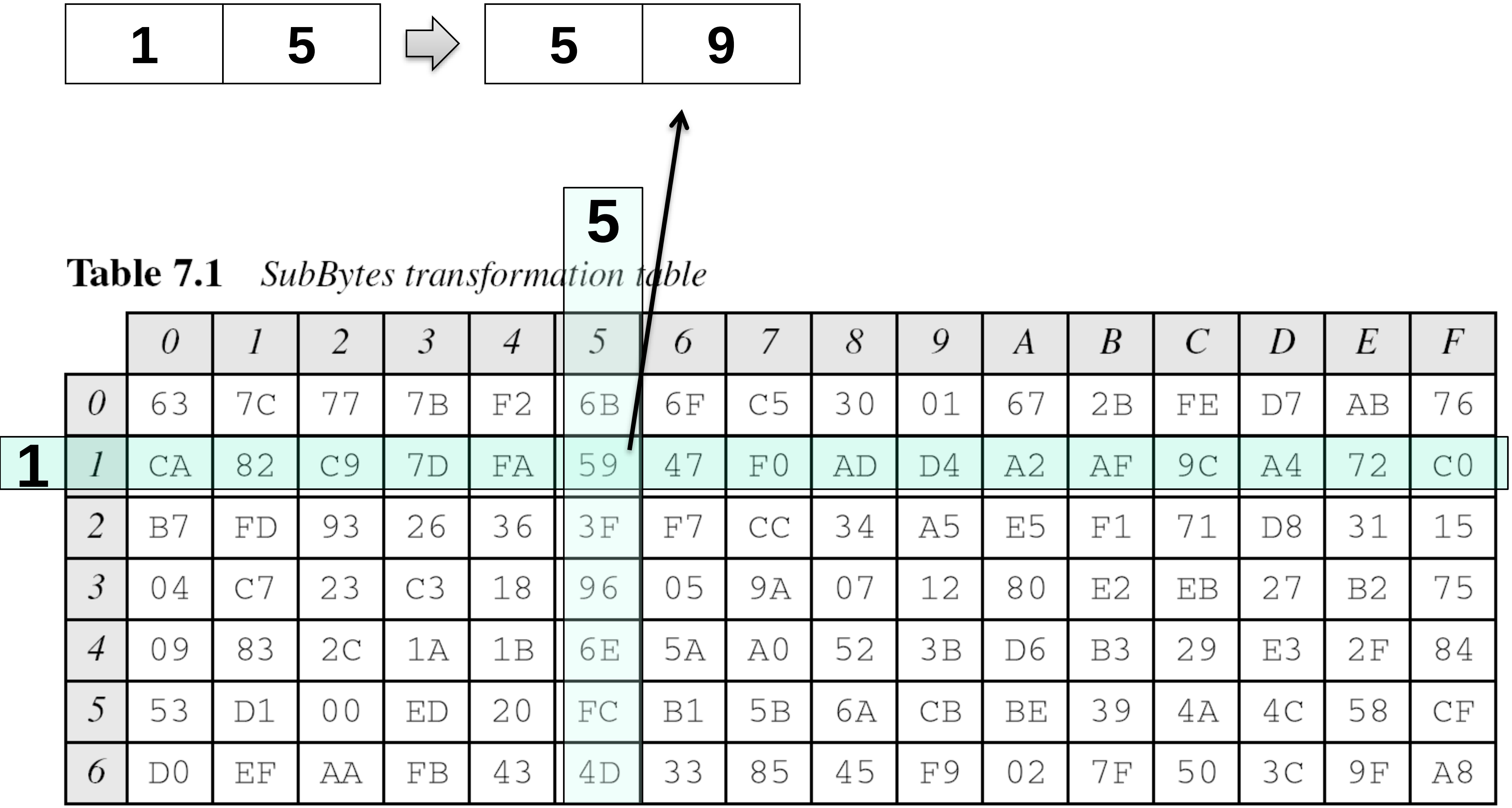


Table 7.1 *SubBytes transformation table*

	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

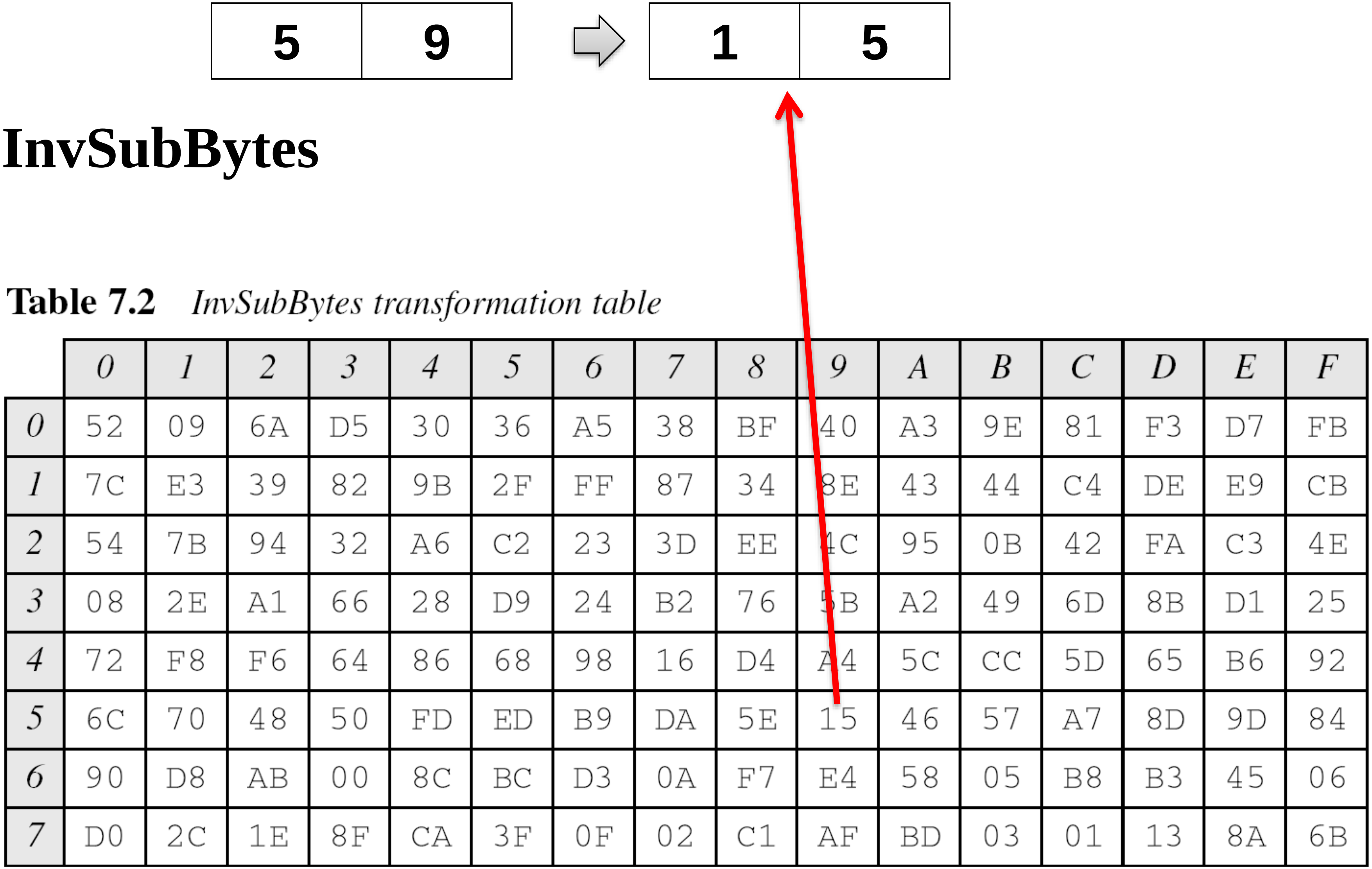
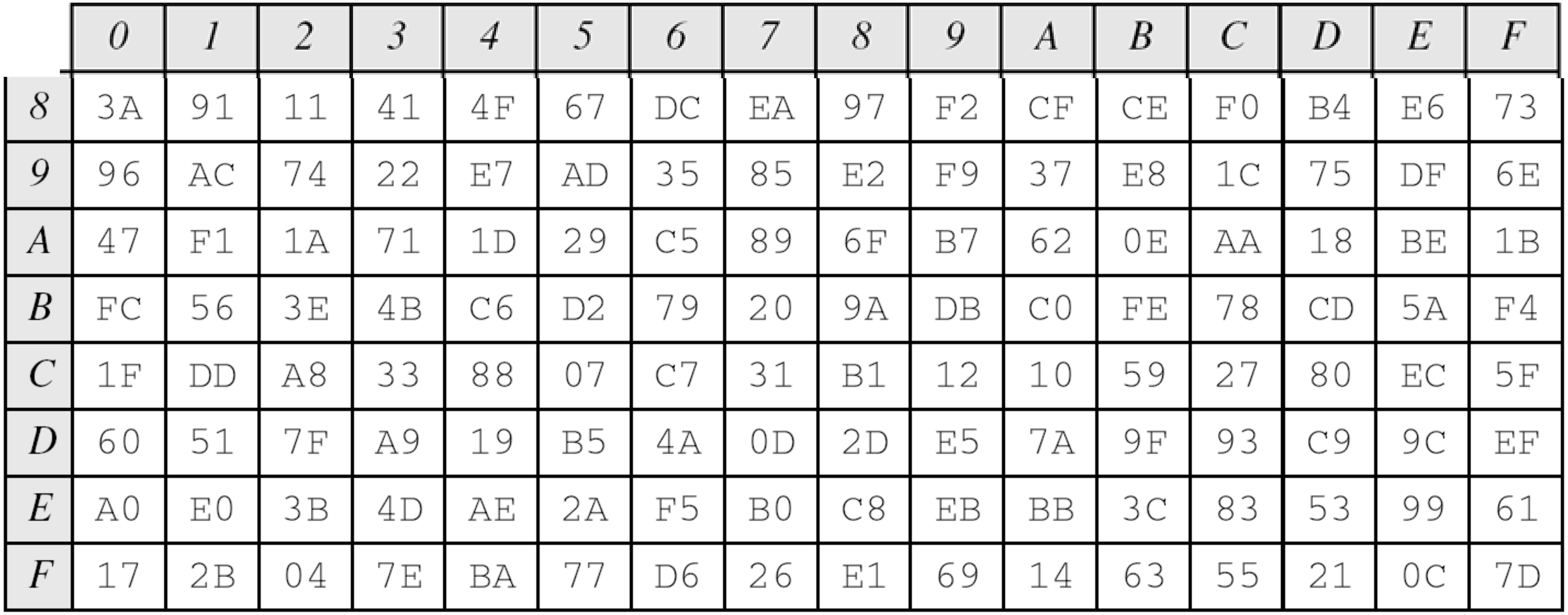


Table 7.2 *InvSubBytes transformation table*

	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

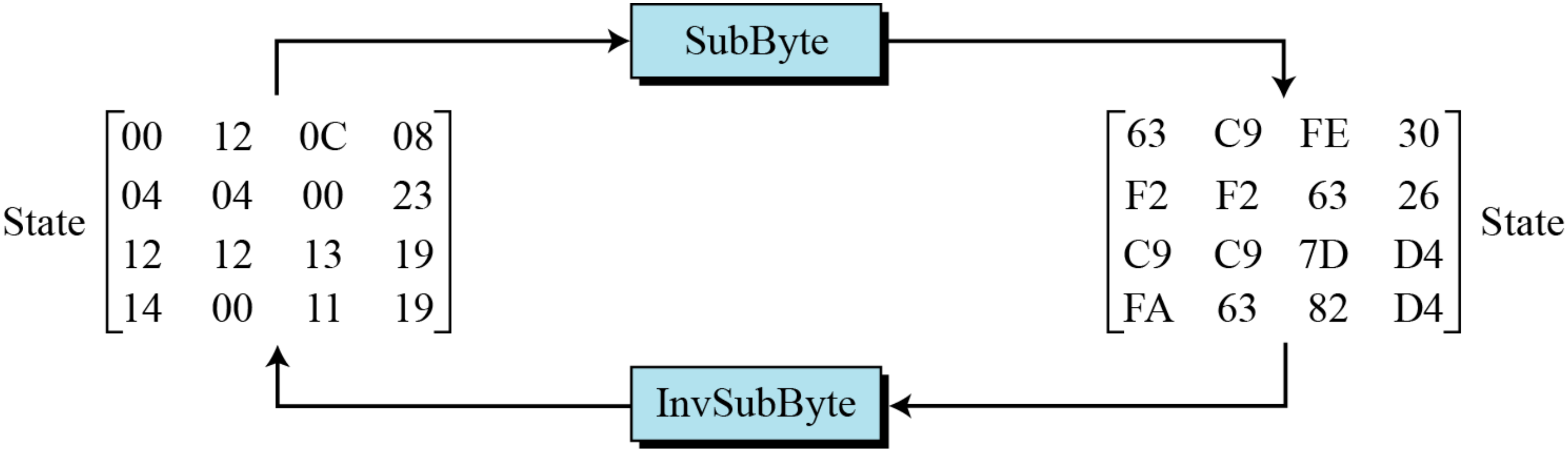


	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	F0	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	1C	75	DF	6E
A	47	F1	1A	71	1D	29	C5	89	6F	B7	62	0E	AA	18	BE	1B
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	1F	DD	A8	33	88	07	C7	31	B1	12	10	59	27	80	EC	5F
D	60	51	7F	A9	19	B5	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 2

Figure 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.

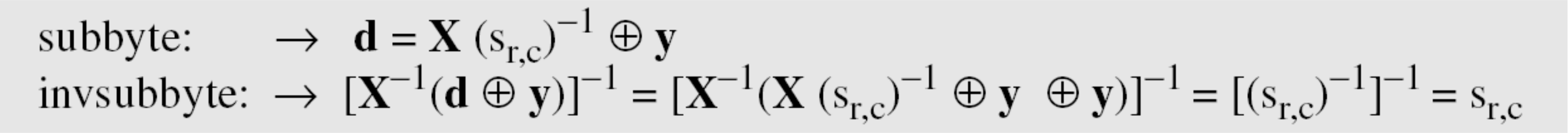
SubBytes transformation for Example 2



$$\text{State}\ \begin{bmatrix} 00 & 12 & 0C & 08 \\ 04 & 04 & 00 & 23 \\ 12 & 12 & 13 & 19 \\ 14 & 00 & 11 & 19 \end{bmatrix} \qquad \begin{bmatrix} 63 & C9 & FE & 30 \\ F2 & F2 & 63 & 26 \\ C9 & C9 & 7D & D4 \\ FA & 63 & 82 & D4 \end{bmatrix}\ \text{State}$$

Transformation Using the GF(2^8) Field

AES also defines the transformation algebraically using the GF(2^8) field with the irreducible polynomials ($x^8 + x^4 + x^3 + x + 1$).

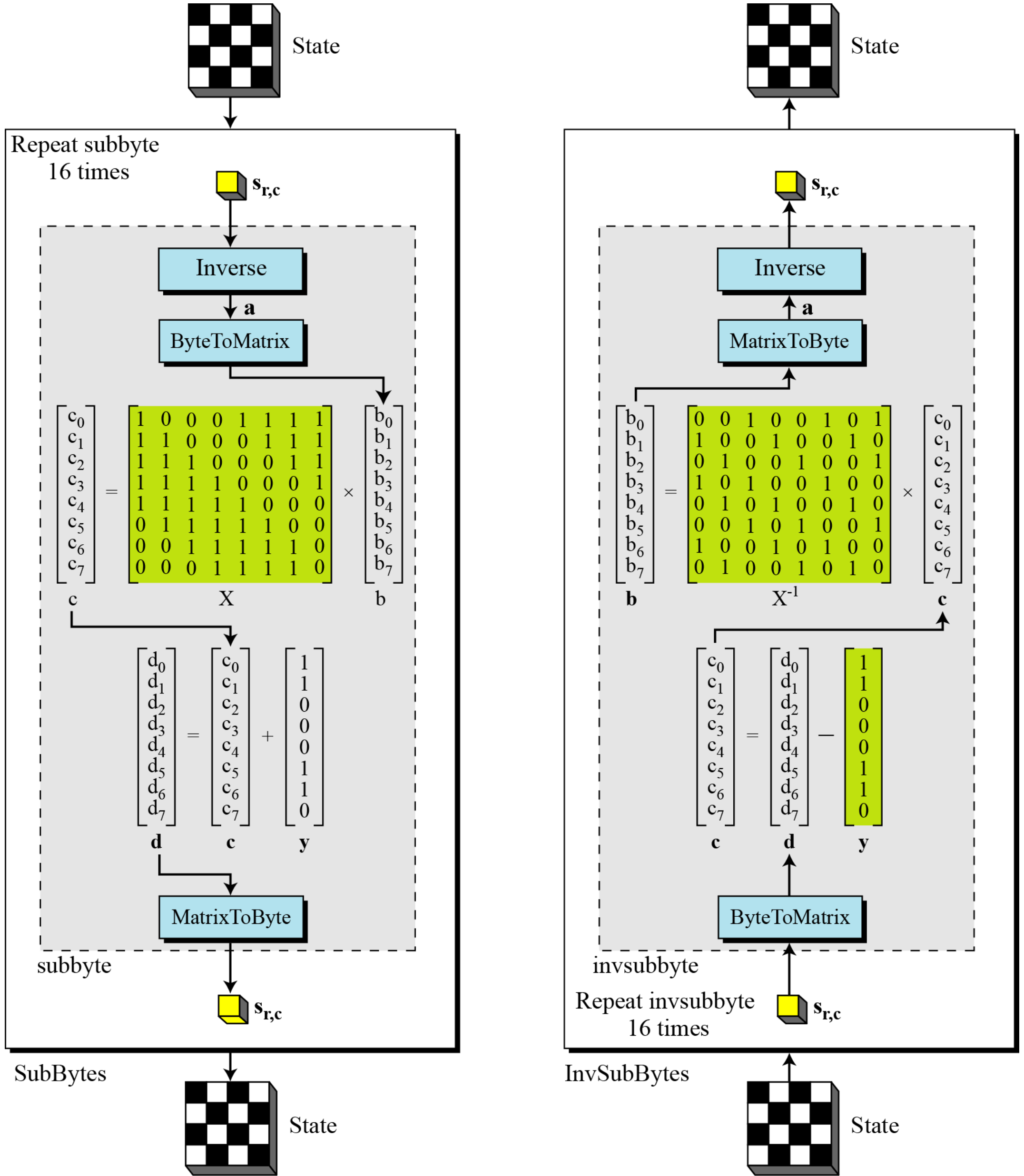


subbyte: → $\mathbf{d} = \mathbf{X}\,(s_{r,c})^{-1} \oplus \mathbf{y}$

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

The SubBytes and InvSubBytes transformations are inverses of each other.

SubBytes and InvSubBytes processes



$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix}$$

c X b

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \end{bmatrix} = \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} + \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

d **c** **y**

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix}$$

b X^{-1} **c**

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \end{bmatrix} = \begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \\ d_4 \\ d_5 \\ d_6 \\ d_7 \end{bmatrix} - \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

c **d** **y**

Example 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.

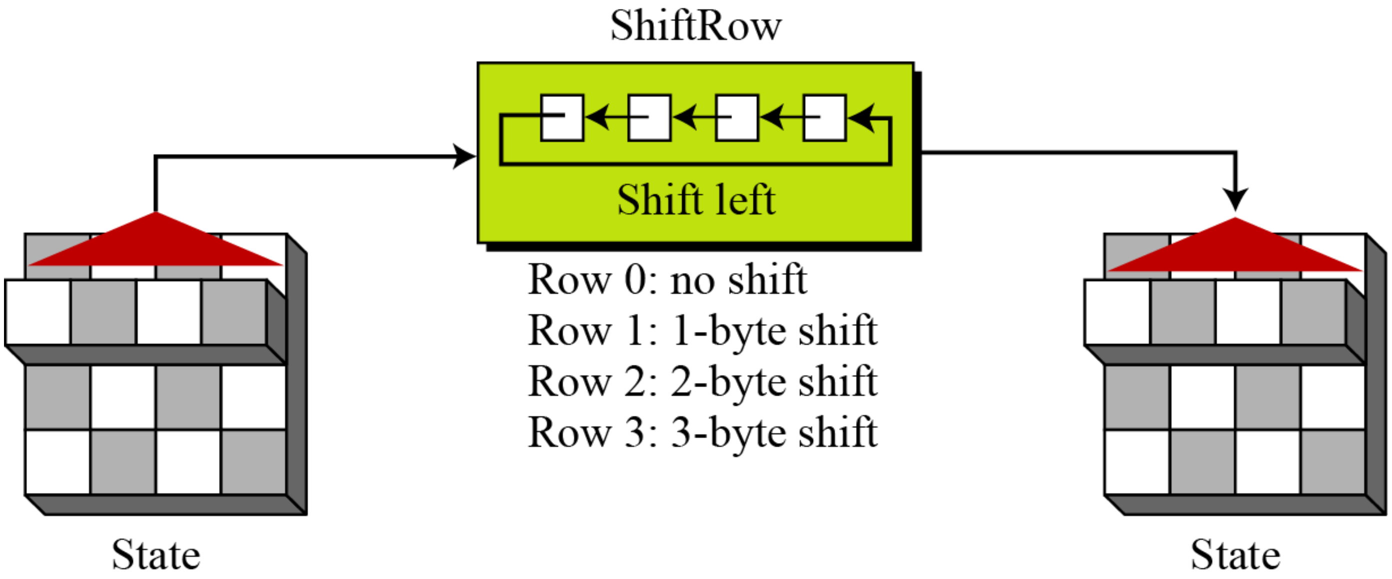
Permutation

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

ShiftRows

In the encryption, the transformation is called ShiftRows.

ShiftRows transformation

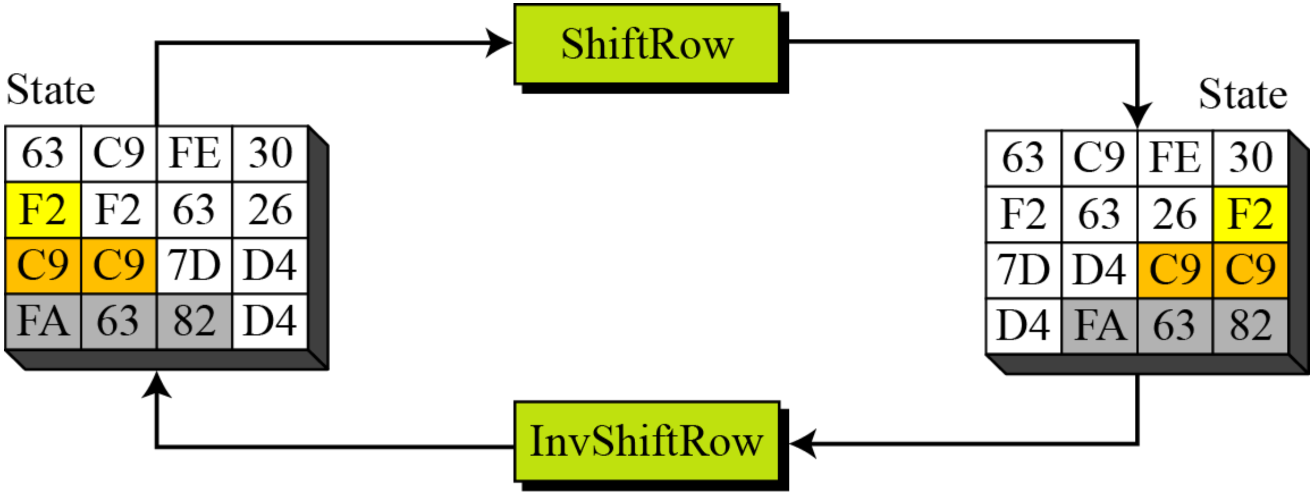


Row 0: no shift
Row 1: 1-byte shift
Row 2: 2-byte shift
Row 3: 3-byte shift

Example 4

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

ShiftRows transformation in Example 4



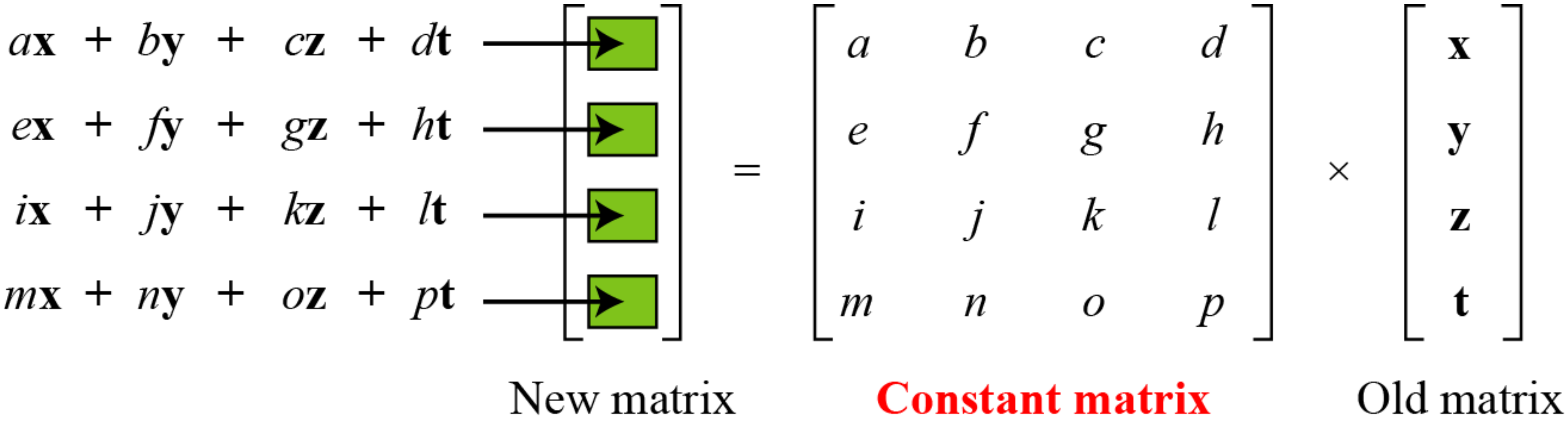
63	C9	FE	30
F2	F2	63	26
C9	C9	7D	D4
FA	63	82	D4

63	C9	FE	30
F2	63	26	F2
7D	D4	C9	C9
D4	FA	63	82

Mixing

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**.

Mixing bytes using matrix multiplication


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

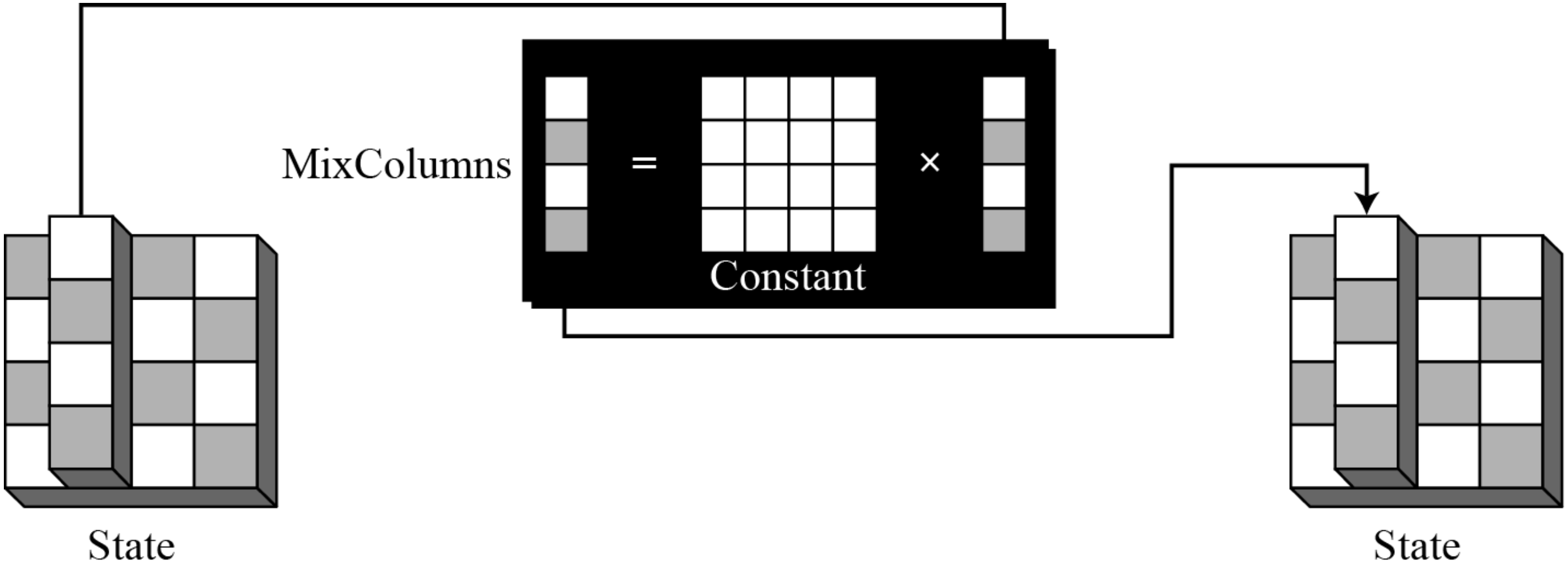
Constant matrices used by MixColumns and InvMixColumns

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

MixColumns

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

MixColumns transformation



InvMixColumns

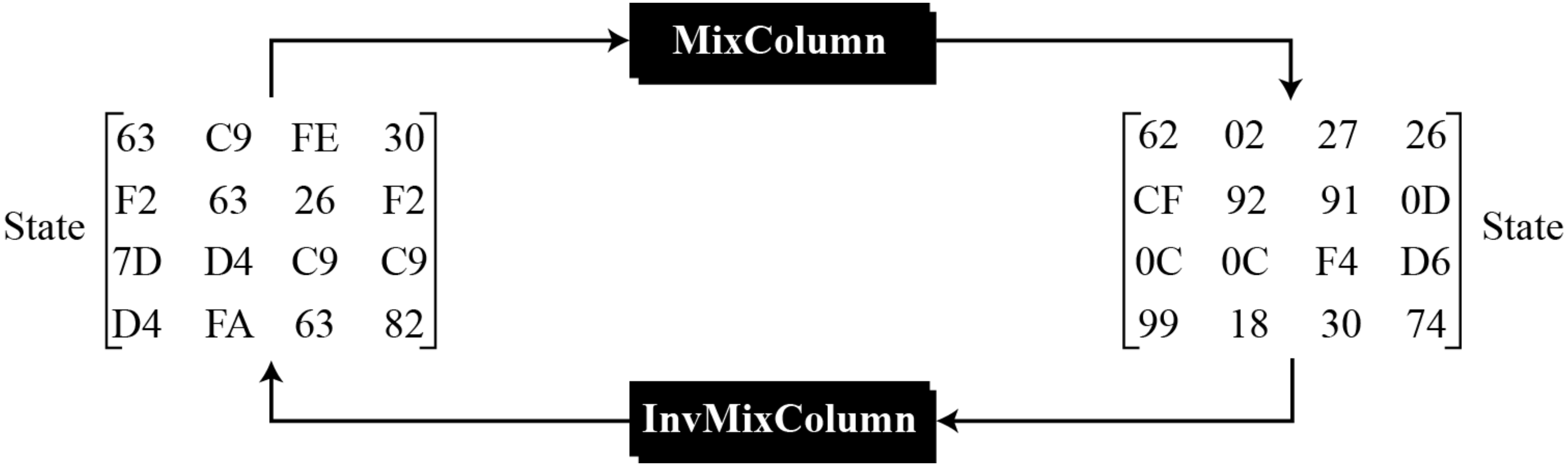
The InvMixColumns transformation is basically the same as the MixColumns transformation.

The MixColumns and InvMixColumns transformations are inverses of each other.

Example 5

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

The MixColumns transformation in Example 5



MixColumn

InvMixColumn

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

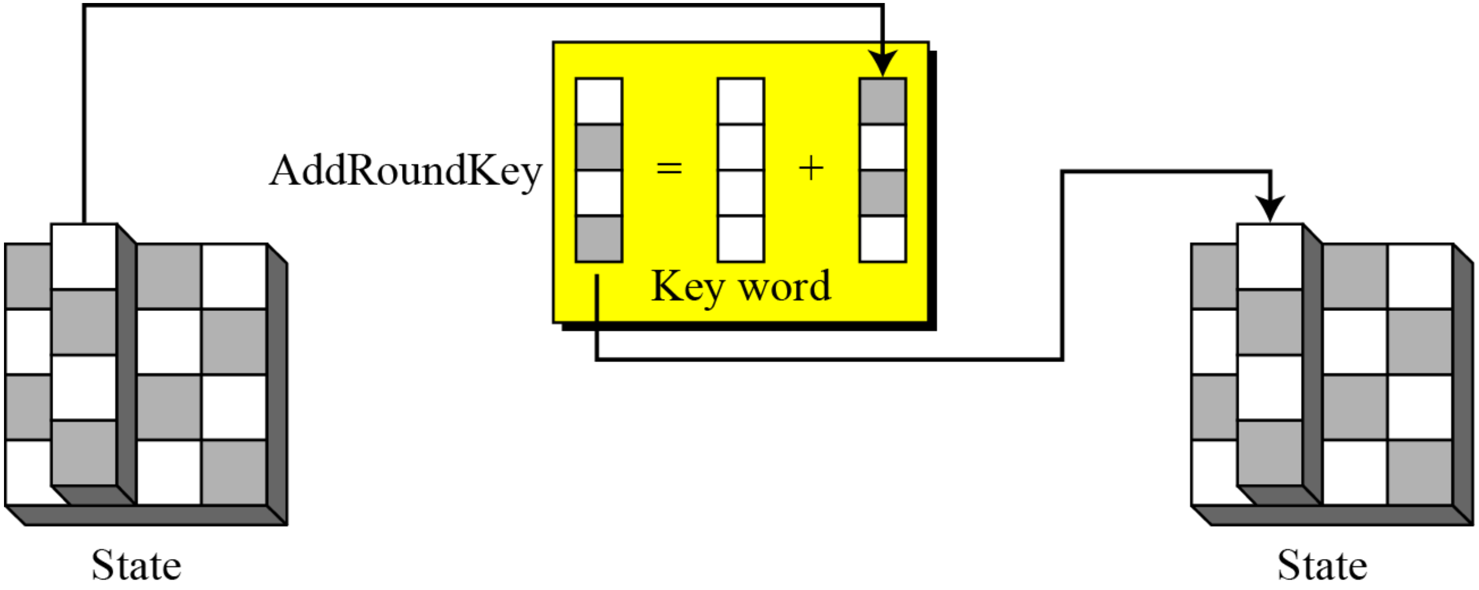
Key Adding

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.

The AddRoundKey transformation is the inverse of itself.

AddRoundKey transformation



KEY EXPANSION

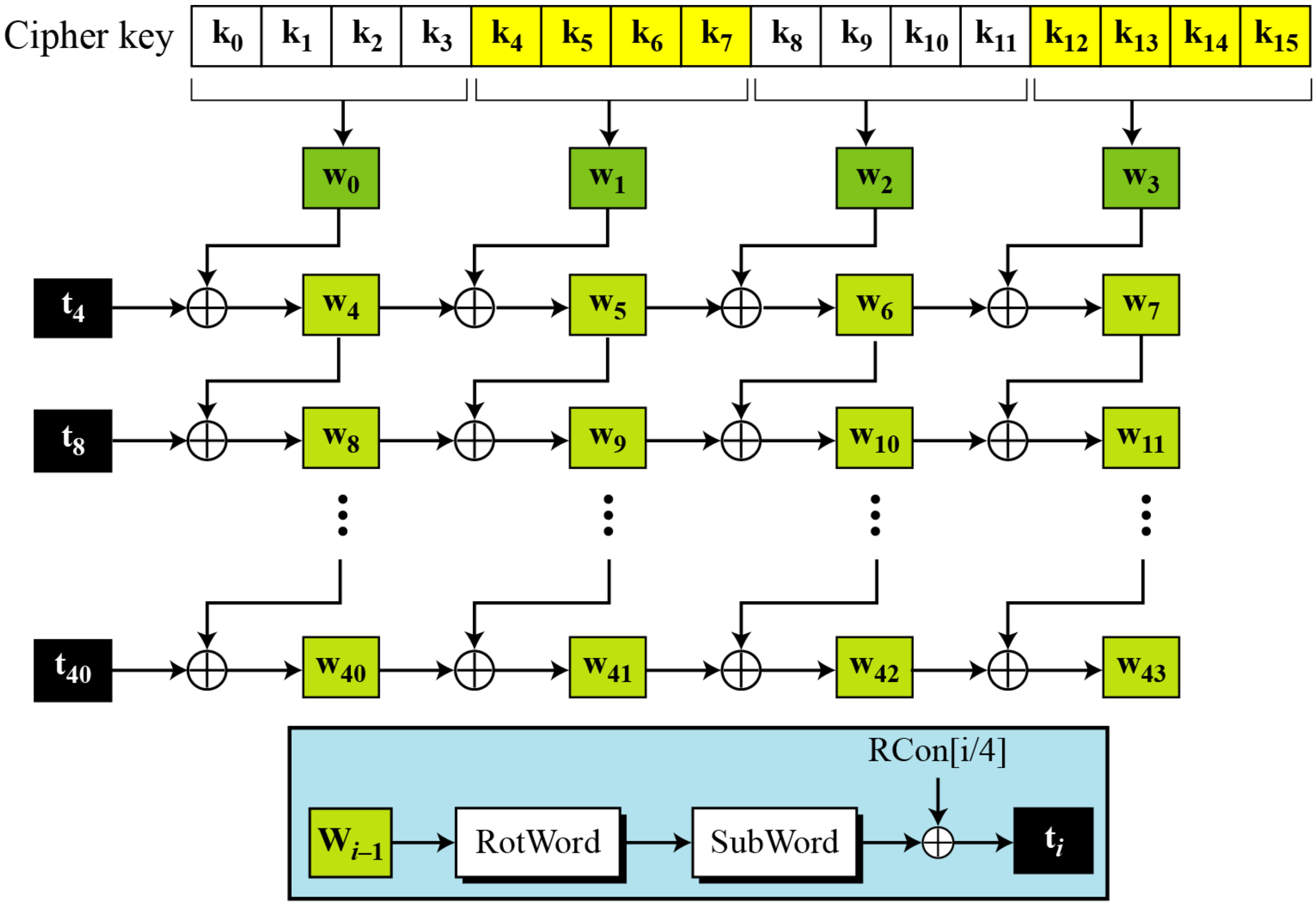
To create round keys for each round, AES uses a key-expansion process. If the number of rounds is N_r , the key-expansion routine creates $N_r + 1$ 128-bit round keys from one single 128-bit cipher key.

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

Key expansion in AES


Cipher key
k_0 k_1 k_2 k_3 k_4 k_5 k_6 k_7 k_8 k_9 k_{10} k_{11} k_{12} k_{13} k_{14} k_{15}
w_0 w_1 w_2 w_3
t_4 w_4 w_5 w_6 w_7
t_8 w_8 w_9 w_{10} w_{11}
t_{40} w_{40} w_{41} w_{42} w_{43}
RCon[i/4]
W_{i-1}
RotWord
SubWord
t_i

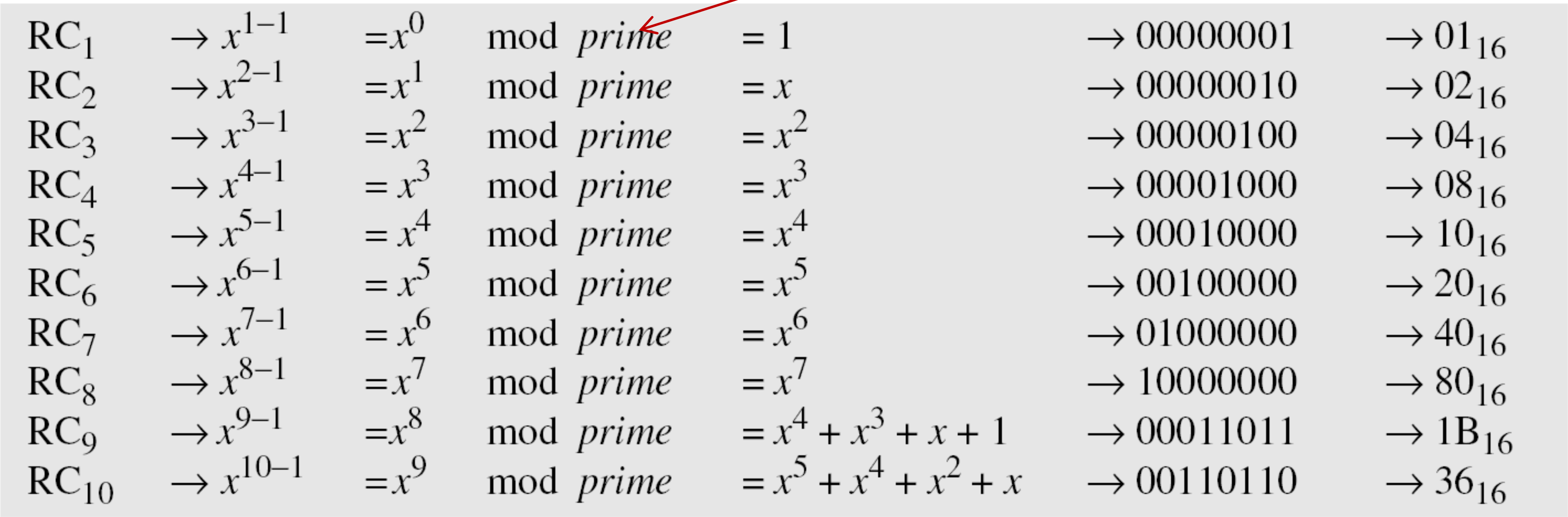
Making of t_i (temporary) words $i = 4\ N_r$

Table 7.4 *RCon constants*

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

The key-expansion routine can either use the following table when calculating the words or use the GF(2^8) field to calculate the leftmost byte dynamically, as shown below (prime is the irreducible polynomial):

$m(x) = x^8 + x^4 + x^3 + x + 1$

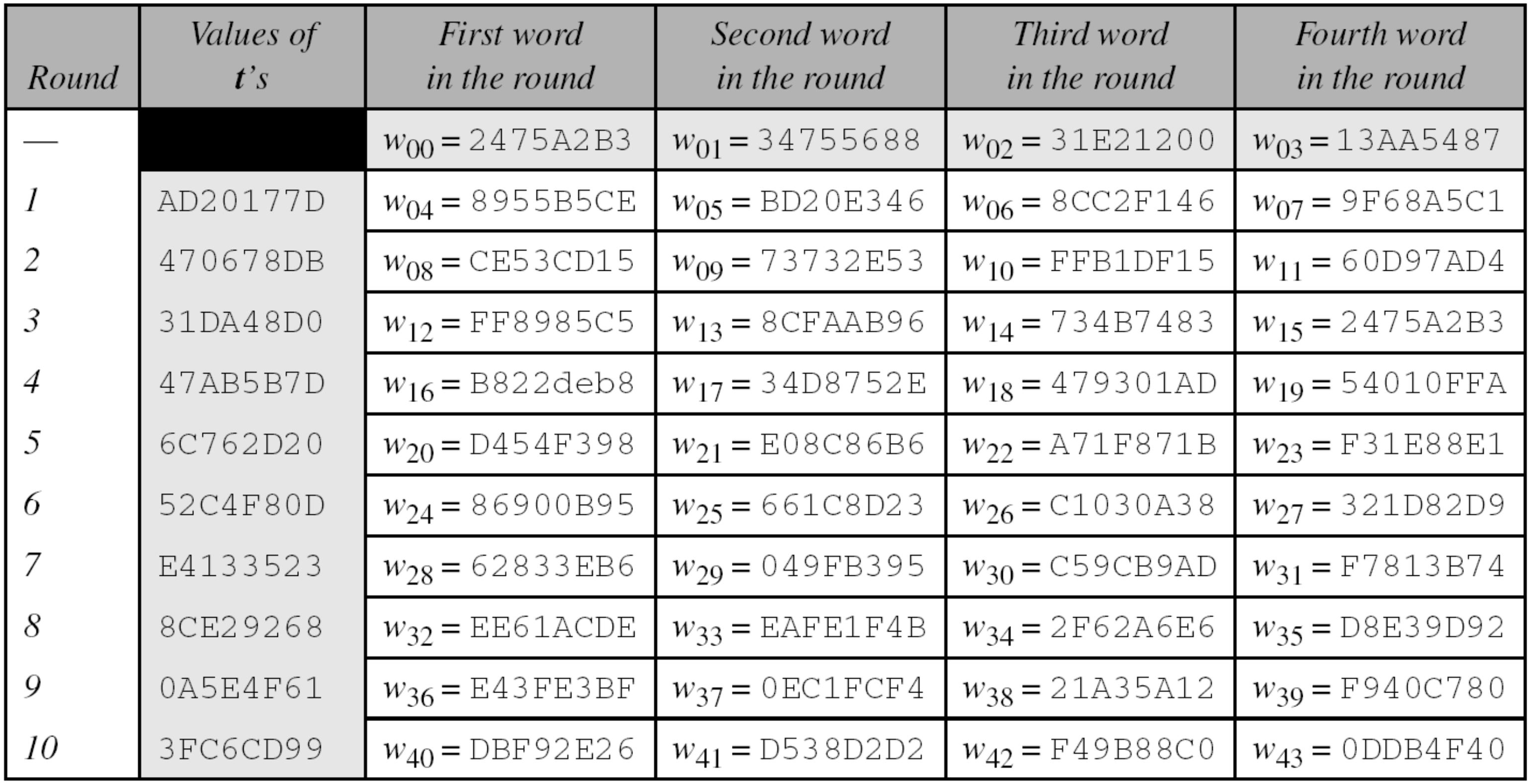


RC_1	$\rightarrow x^{1-1}$	$=x^0$	mod *prime*	$= 1$	$\rightarrow 00000001$	$\rightarrow 01_{16}$
RC_2	$\rightarrow x^{2-1}$	$=x^1$	mod *prime*	$= x$	$\rightarrow 00000010$	$\rightarrow 02_{16}$
RC_3	$\rightarrow x^{3-1}$	$=x^2$	mod *prime*	$= x^2$	$\rightarrow 00000100$	$\rightarrow 04_{16}$
RC_4	$\rightarrow x^{4-1}$	$=x^3$	mod *prime*	$= x^3$	$\rightarrow 00001000$	$\rightarrow 08_{16}$
RC_5	$\rightarrow x^{5-1}$	$=x^4$	mod *prime*	$= x^4$	$\rightarrow 00010000$	$\rightarrow 10_{16}$
RC_6	$\rightarrow x^{6-1}$	$=x^5$	mod *prime*	$= x^5$	$\rightarrow 00100000$	$\rightarrow 20_{16}$
RC_7	$\rightarrow x^{7-1}$	$=x^6$	mod *prime*	$= x^6$	$\rightarrow 01000000$	$\rightarrow 40_{16}$
RC_8	$\rightarrow x^{8-1}$	$=x^7$	mod *prime*	$= x^7$	$\rightarrow 10000000$	$\rightarrow 80_{16}$
RC_9	$\rightarrow x^{9-1}$	$=x^8$	mod *prime*	$= x^4 + x^3 + x + 1$	$\rightarrow 00011011$	$\rightarrow 1B_{16}$
RC_{10}	$\rightarrow x^{10-1}$	$=x^9$	mod *prime*	$= x^5 + x^4 + x^2 + x$	$\rightarrow 00110110$	$\rightarrow 36_{16}$

Example 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 $(24\ 75\ \mathrm{A2}\ \mathrm{B3}\ 34\ 75\ 56\ 88\ 31\ \mathrm{E2}\ 12\ 00\ 13\ \mathrm{AA}\ 54\ 87)_{16}$.

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

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 8

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

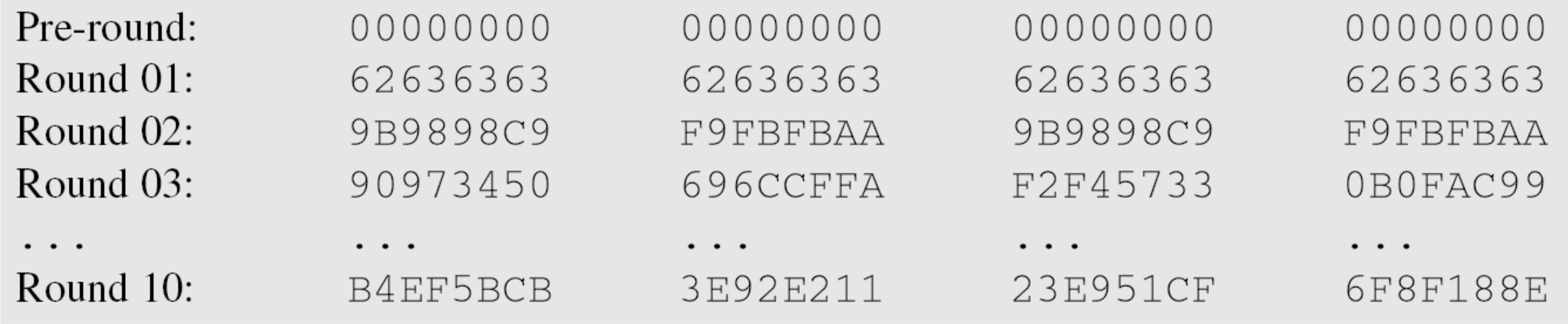
Cipher Key 1: 12 45 A2 A1 23 31 A4 A3 B2 CC A**A** 34 C2 BB 77 23
Cipher Key 2: 12 45 A2 A1 23 31 A4 A3 B2 CC A**B** 34 C2 BB 77 23

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 9

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.

Key-Expansion Analysis

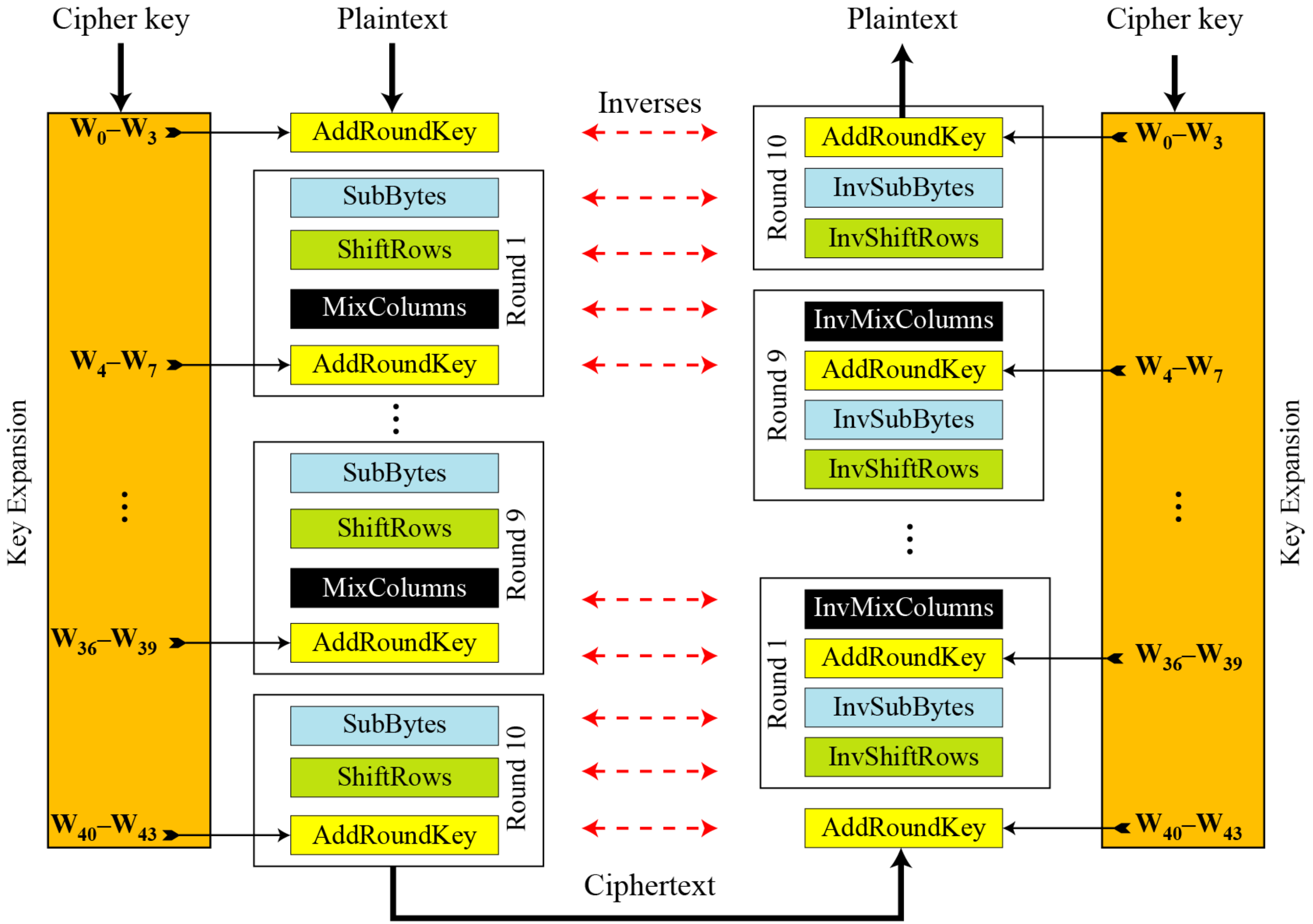
The key-expansion mechanism in AES has been designed to provide several features that thwart the cryptanalyst.

CIPHERS

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.

Original Design

Ciphers and inverse ciphers of the original design



Security

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.

But side-channel (timing attacks), meet-in-the-middle and Square matrix attacks are under research.

Security

AES structure has advantages and disadvantages.

- Each step consists of a number of operations that can be performed in parallel.
  - This makes high-speed implementations easy.
  - But, the decryption operation is significantly different from encryption; we need the inverse lookup table of the S-Box, and the inverse mixing operation is different from the original mixing operation.
- S-Boxes provide non-linearity and byte shuffle and mixing function provide diffusion.
- AES designers showed an attack on 6 rounds. Therefore round counts chosen 10-14.
- In 2009; “Key Recovery Attacks on AES up to 10 rounds” http://eprint.iacr.org/2009/374
- In 2009; “Related key attacks, Full AES-192 and AES-256” http://eprint.iacr.org/2009/317

Security

All known attacks are theoretical, not practical. AES is already a standard and is using in many real applications.

Implementation

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.