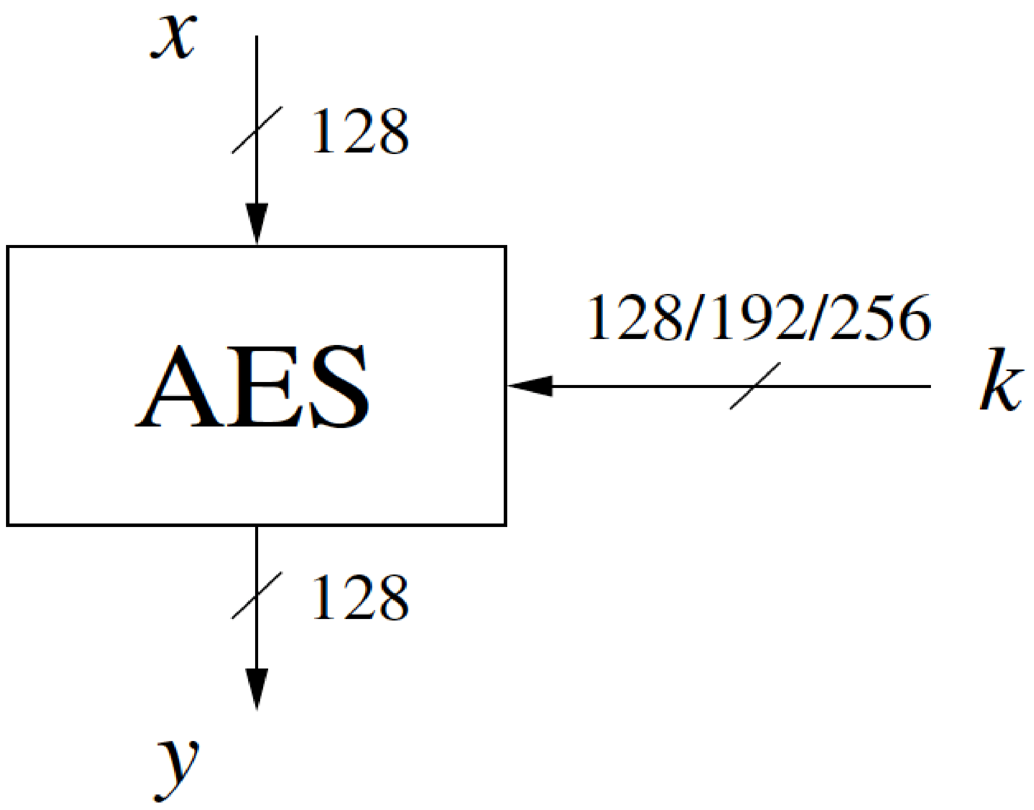
Chapter 4: The Advanced Encryption Standard (AES)

- The encryption and decryption function of AES
- Introduction to Galois Fields
- The internal structure of AES:
  - byte substitution layer
  - diffusion layer
  - key addition layer
  - key schedule

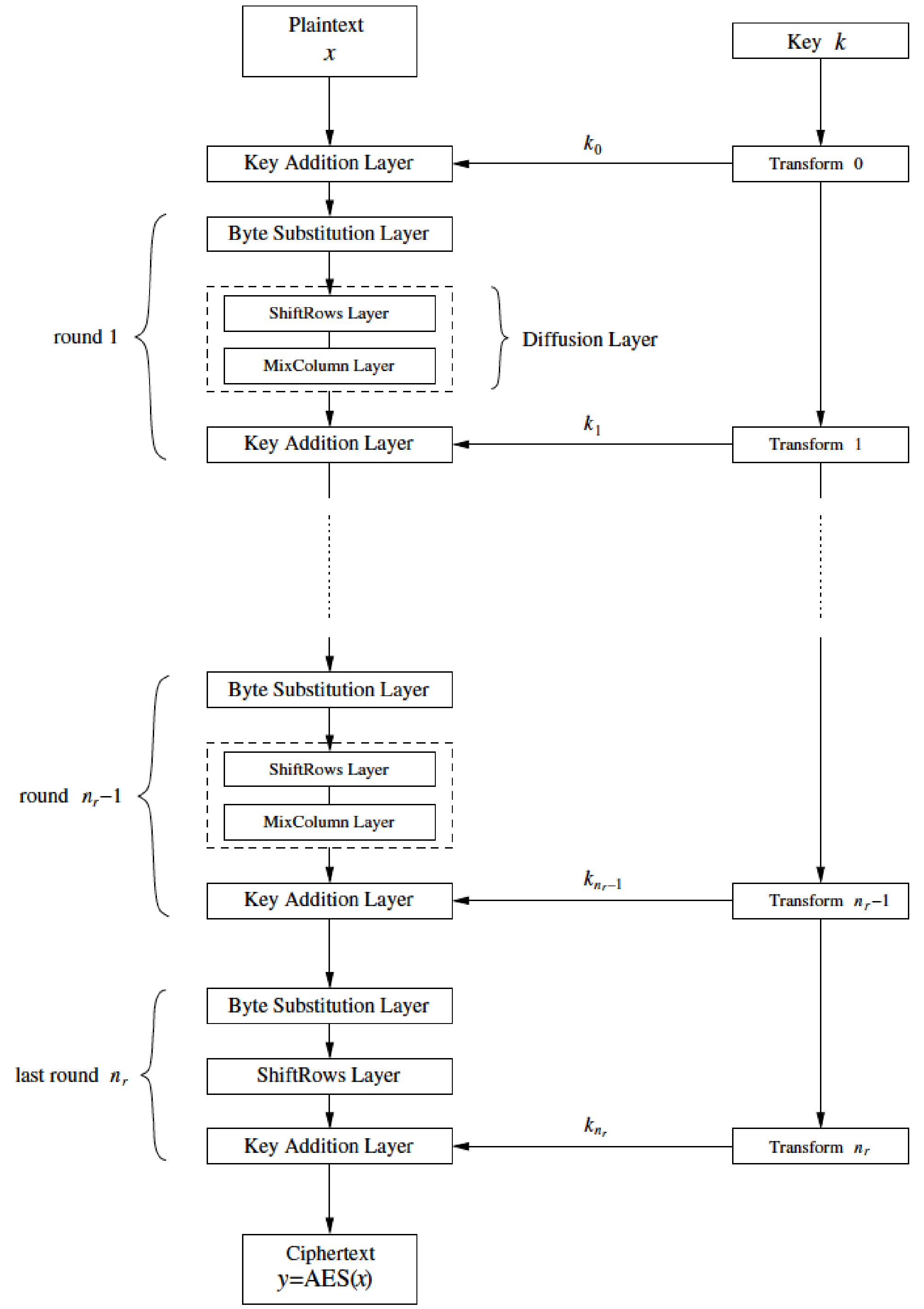
Advanced Encryption Standard (AES)

- The most-used symmetric cipher
- In 1997 NIST (National Institute of Standards and Technology) called for proposals for a new Advanced Encryption Standard
- The requirements for all AES candidate submissions were:
  - Block cipher with 128-bit block size
  - Three supported key lengths: 128, 192 and 256 bit
  - Efficiency in software and hardware
- In 1999, five finalist algorithms were announced:
  - **Mars** by IBM Corporation
  - **RC6** by RSA Laboratories
  - **Rijndael**, by Vincent Rijmen and Joan Daemen
  - **Serpent**, by Ross Anderson, Eli Biham and Lars Knudsen
  - **Twofish**, by Bruce Schneier, John Kelsey, Doug Whiting, DavidWagner, Chris Hall and Niels Ferguson
- In 2001, NIST declared **Rijndael** as the new AES and approved as a US federal standard.

AES input/output parameters

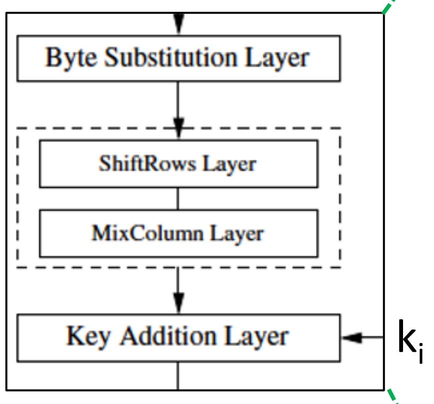


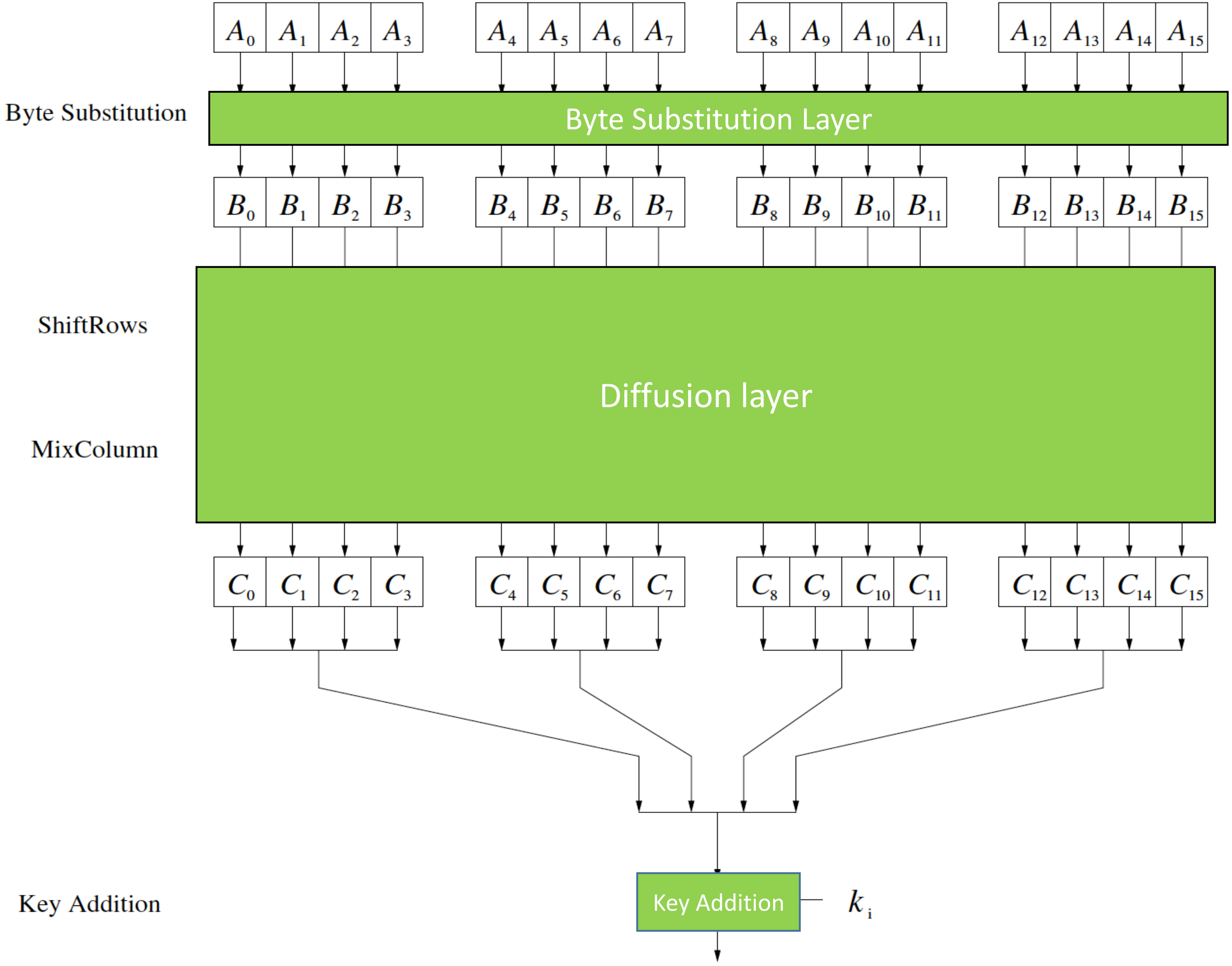
key lengths	# rounds = n_r
128 bit	10
192 bit	12
256 bit	14



- Each round consists of ***layers*** .
- Each layer manipulates all 128 bits of the ***data path*** (also referred to as the ***state*** of the algorithm).
- Encrypts all 128 bits in one round.
  - AES does not have a Feistel structure.

AES Encryption
i^{th} Round





Each square such as A_0, B_0, and C_0 represents a **byte** (= 8 bits).

AES and Galois Field

- AES is a **byte**-oriented cipher (for software efficiency)
- AES encryption and decryption perform arithmetic operations (**+, -, *, /**) on bytes.
- In AES, every byte of the internal data path is treated as an element of the the **Galois field GF(2^8)** and manipulates the data by performing arithmetic in this finite field.
- In Abstract Algebra, there are three mathematical objects: Group, Ring, and Field.

- What is a **field**?
  1. The field is a set of elements with which you can perform **+, -, *, /.**
  2. Every element (except for zero) must have a multiplicative inverse.
     - Therefore, $a/b = a*b^{-1}$ is always defined if b != 0.
- Field Examples:
  - The set of rational numbers
  - The set of real numbers
  - Z_5 = {0, 1, 2, 3, 4}. Modular arithmetic
  - What about Z_{26} = {0, 1, 2, …, 25}?

Galois Field $GF(2^8)$ – Data Representation

- Bit representation
  - $GF(2^8) = \{(a_7,a_6,a_5,a_4,a_3,a_2,a_1,a_0) \mid a_i = 0, 1\}$
  - There are $2^8 = 256$ elements in $GF(2^8)$.

- Polynomial Representation
  - Each element $A \in GF(2^8)$ can be represented as a polynomial with coefficients a_i:

    $a_7x^7 + a_6x^6 + \ldots + a_1x + a_0$, each $a_i = 0$ or 1.
  - $GF(2^8) = \{a_7x^7 + a_6x^6 + \ldots + a_1x + a_0 \mid a_i = 0, 1\}$

- A byte of **8 bits** can be represented as a **polynomial** (with the 8 bits as the coefficients) and vice versa:

$$(a_7,a_6,a_5,a_4,a_3,a_2,a_1,a_0) = a_7x^7 + a_6x^6 + \ldots + a_1x + a_0$$

- Examples:

$(0,1,1,0,1,0,0,0) = x^6 + x^5 + x^3$

$(1,0,0,0,1,0,1,1) = x^7 + x^3 + x + 1$

Galois Field GF(2^8) - Addition

- Let A = (a_7,a_6,a_5,a_4,a_3,a_2,a_1,a_0) and B = (b_7,b_6,b_5,b_4,b_3,b_2,b_1,b_0)
- **A + B** is defined as **A XOR B:**

A + B = (a_7^b_7, a_6 ^ b_7, …, a_0 ^ b_0)

```
   A = (1, 0, 1, 1 ,0, 1, 0, 0)
+  B = (0, 0, 1, 0, 1, 1, 0, 1)
-----------------------------------------------------------------
A+B = (1,  0, 0, 1, 1, 0, 0, 1)
```

Galois Field $GF(2^8)$ - Addition

- Let A = $(a_7,a_6,a_5,a_4,a_3,a_2,a_1,a_0)$ and B = $(b_7,b_6,b_5,b_4,b_3,b_2,b_1,b_0)$
- **A + B** is defined as **A XOR B:**

A + B = $(a_7 \wedge b_7,\ a_6 \wedge b_7,\ \ldots,\ a_0 \wedge b_0)$,

```
    A = (1, 0, 1, 1 ,0, 1, 0, 0)    =  x^7 +   x^5 +  x^4 +       +  x^2
+   B = (0, 0, 1, 0, 1, 1, 0, 1)    =          x^5          +  x^3 +  x^2  + 1
------------------------------------------------------------------------------
A+B = (1, 0, 0, 1, 1, 0, 0, 1)    = x^7+          + x^4 +x^3+  + 1
```

The coefficients of the polynomials in $GF(2^8)$ are field elements in $\mathbf{Z_2}$ = {0, 1}

Galois Field $GF(2^8)$ - Multiplication

$$A = (0, 0, 1, 0, 0, 0, 1, 0) = x^5 + x$$
$$* \quad B = (0, 1, 0, 0, 0, 0, 0, 0) = x^6$$

$$A * B = (x^5 + x)\, x^6$$

$$= x^{11} + x^7 \qquad \bmod (x^8 + x^4 + x^3 + x + 1)$$

- AES uses $P(x) = x^8 + x^4 + x^3 + x + 1$ as the **reduction** (or **irreducible**) polynomial.

- x*(x^7+x^3+x^2+1) = 1
- What is the inverse of x = (0,0,0,0,0,0,1,0)
- Multiplication in $GF(2^8)$ is the polynomial multiplication with modulo P(x).
  - To obtain a remainder, you can divide the product by P(x) using the long division.
  - A better way to obtain a remainder is "reducing" the product using the relation: $x^8 = x^4 + x^3 + x + 1 \bmod P(x)$

Galois Field GF(2^8) – Multiplication

$GF(2^8) = \{a_7x^7 + a_6x^6 + \ldots + a_1x + a_0 \mid a_i = 0, 1\}$

Modulo polynomial:

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

$$
\begin{array}{r|l}
 & x^3 \\
\hline
x^8 + x^4 + x^3 + x + 1 & x^{11} + x^7 \\
 & -\; x^{11} + x^7 + x^6 + x^4 + x^3 \\
\hline
 & -x^6 - x^4 - x^3
\end{array}
$$

In summary,

$$
\begin{aligned}
(x^5 + x)\,x^6 &= x^{11} + x^7 \\
&= -x^6 - x^4 - x^3 \\
&= x^6 + x^4 + x^3 \mod P(x)
\end{aligned}
$$

Galois Field GF(2^8) – Multiplication

- The other way to obtain a remainder is "reducing" the product using the relation:
  $x^8 = x^4 + x^3 + x + 1 \mod P(x)$

Why?

Since $x^8 + x^4 + x^3 + x + 1 = 0$ in mod P(x),

x^8 = -x^4 - x^3 - x - 1

= x^4 + x^3 + x + 1

$x^{11} + x^7$

= x^3 * ($x^4 + x^3 + x + 1$) + x^7

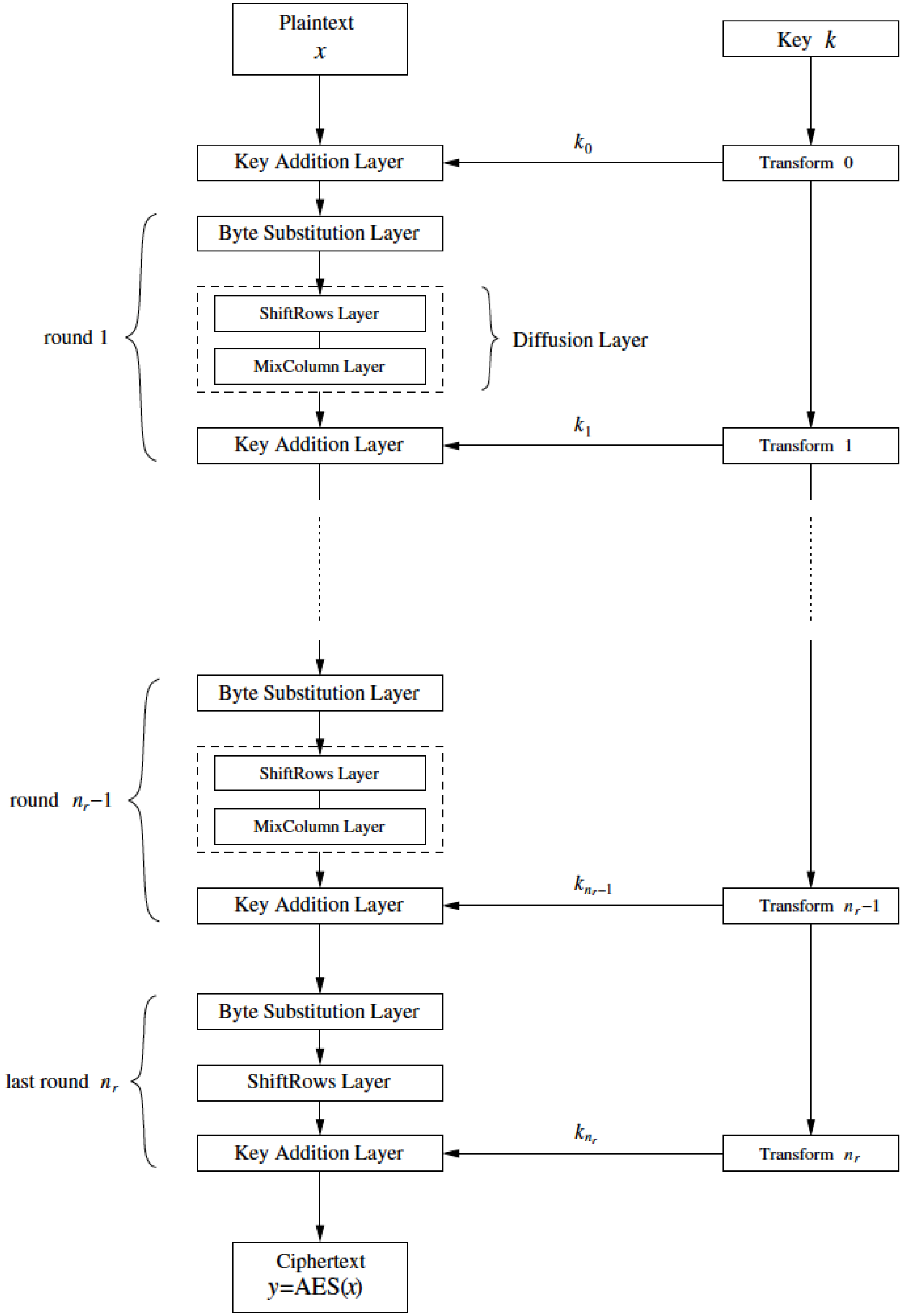
= x^7 + x^6 + x^4 + x^3 + x^7

= x^6 + x^4 + x^3

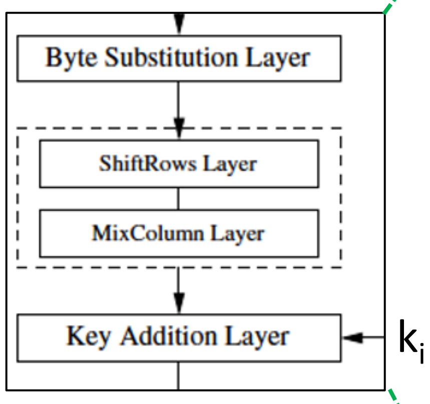
= (0,1,0,1,1,0,0,0)

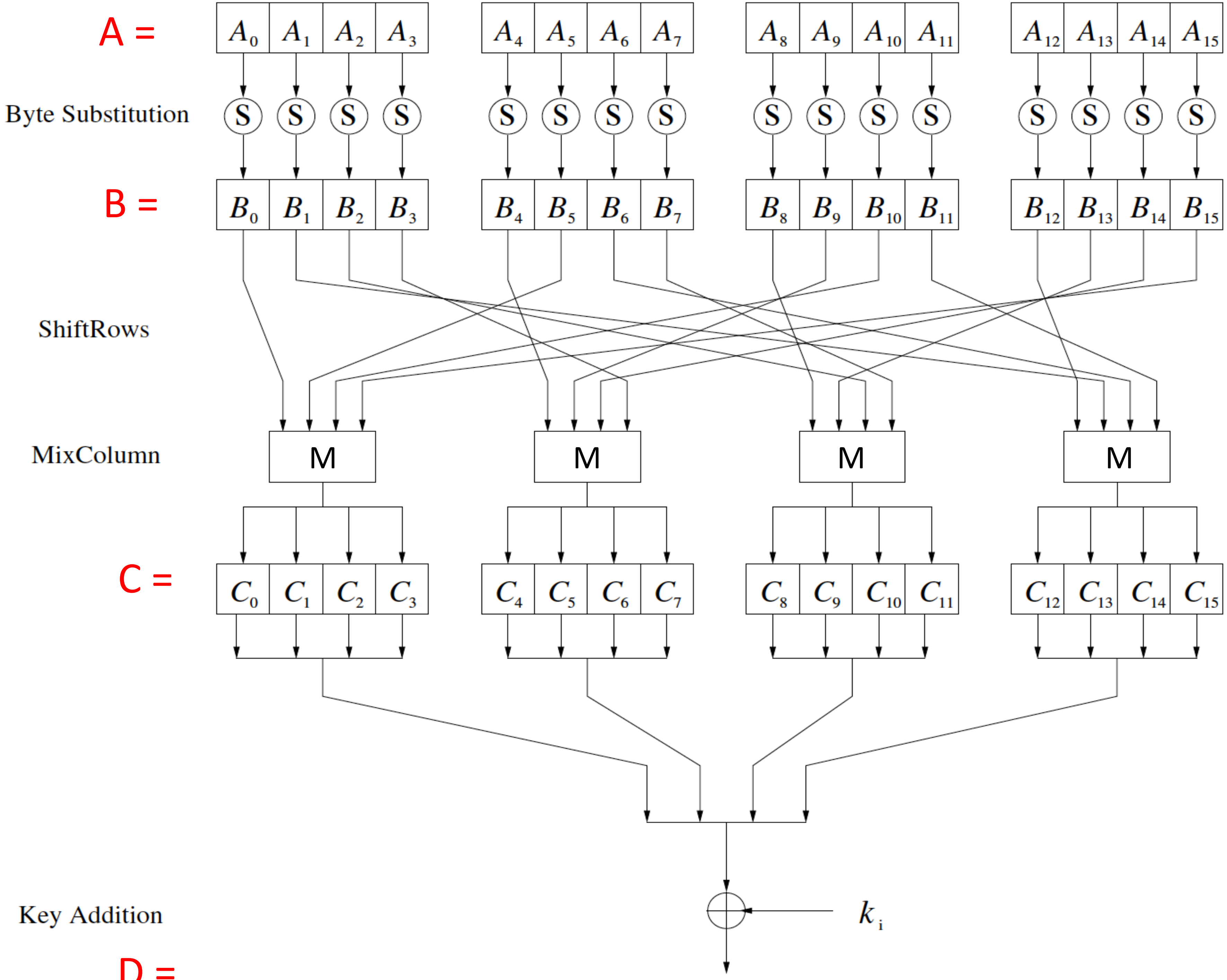
Layers

- Byte Substitution layer (S-Box)
  - Each element of the state is nonlinearly transformed using lookup tables with special mathematical properties.
- Diffusion layer
  - The ***ShiftRows*** layer permutes the data on a byte level.
  - The ***MixColumn*** layer is a matrix operation which mixes blocks of four bytes.
- Key Addition layer
  - A 128-bit round key is XORed to the state.



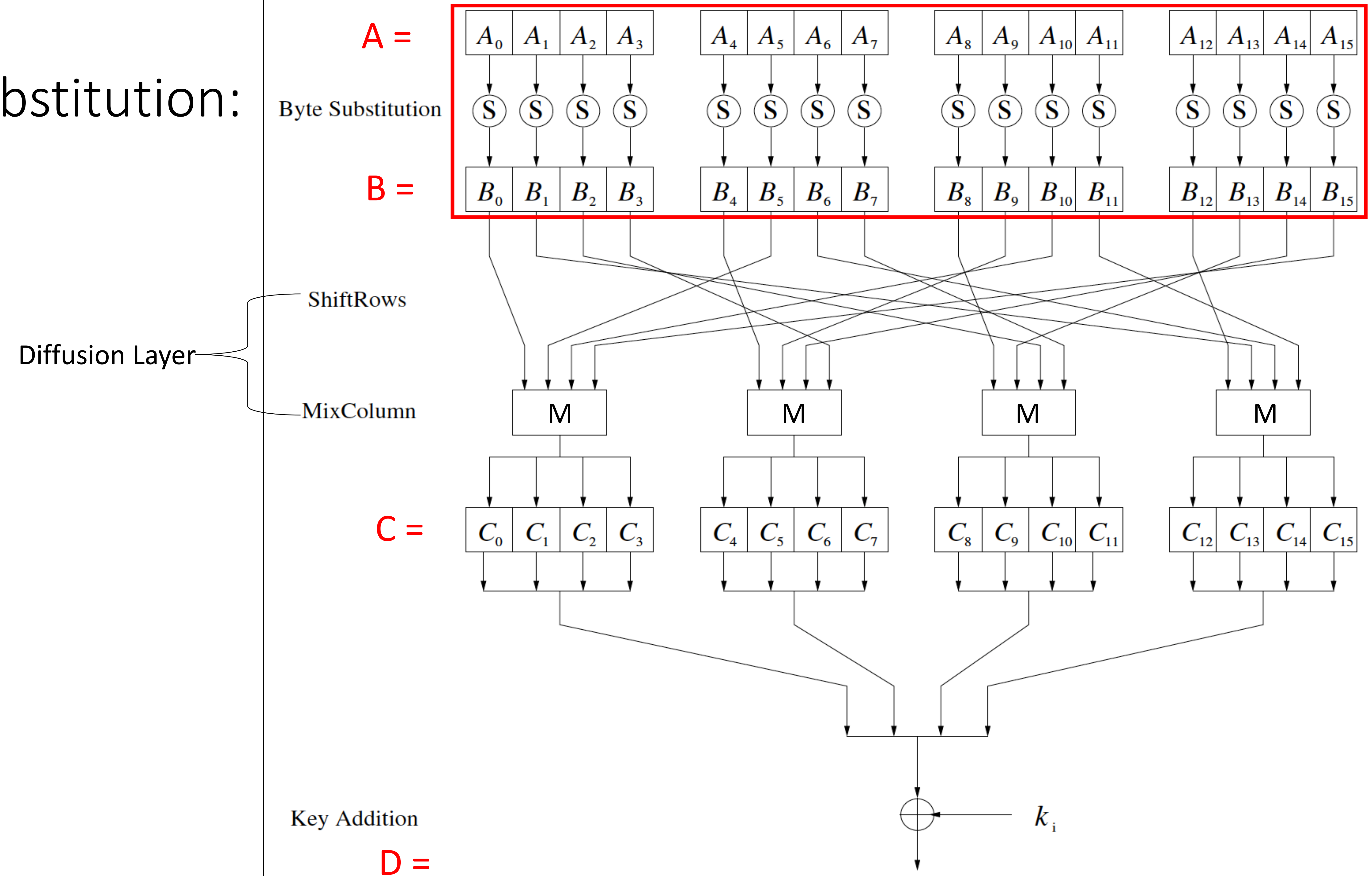
A Round


Byte Substitution Layer
ShiftRows Layer
MixColumn Layer
Key Addition Layer
k_i


A =
A_0 A_1 A_2 A_3 A_4 A_5 A_6 A_7 A_8 A_9 A_{10} A_{11} A_{12} A_{13} A_{14} A_{15}
Byte Substitution
S
B =
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}
ShiftRows
MixColumn
M
C =
C_0 C_1 C_2 C_3 C_4 C_5 C_6 C_7 C_8 C_9 C_{10} C_{11} C_{12} C_{13} C_{14} C_{15}
Key Addition
k_i
D =

Byte Substitution:

A --> B



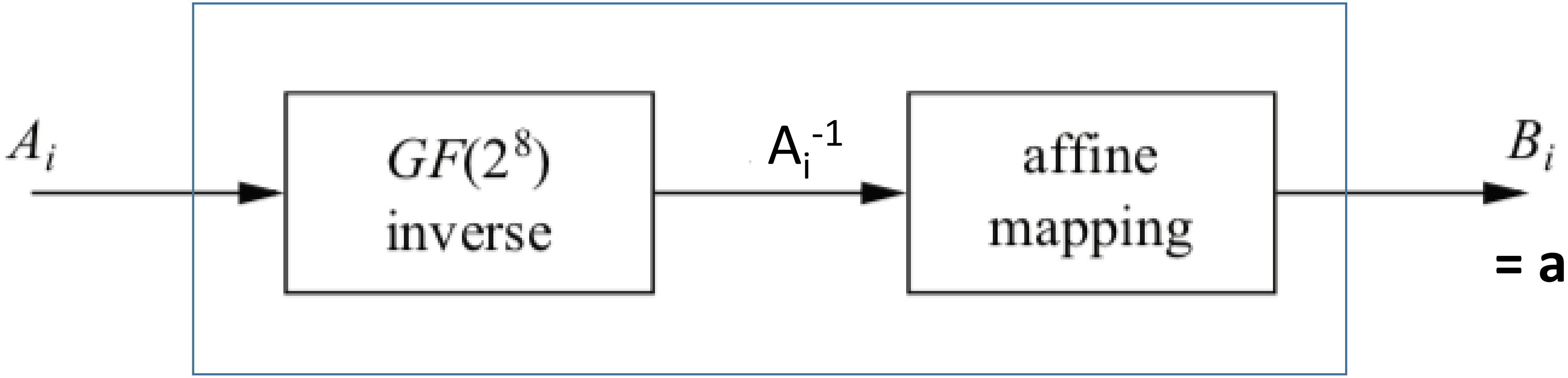
Mathematical description of the S-Box

- Unlike the DES S- Boxes, which are essentially random tables that fulfill certain properties, the AES S-Box has a strong algebraic structure. An AES S-Box can be viewed as a two- step mathematical transformation:

S-box

Mathematical description of the S-Box

- Unlike the DES S- Boxes, which are essentially random tables that fulfill certain properties, the AES S-Box has a strong algebraic structure. An AES S-Box can be viewed as a two- step mathematical transformation:



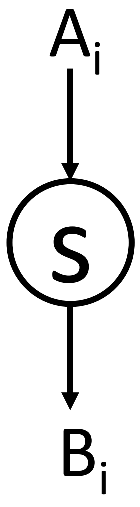
$= \mathbf{a} * A_i^{-1} + \mathbf{b}$

S-box

AES S-Box

The numbers are in hexadecimal notation for input byte **xy**

x \ 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
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



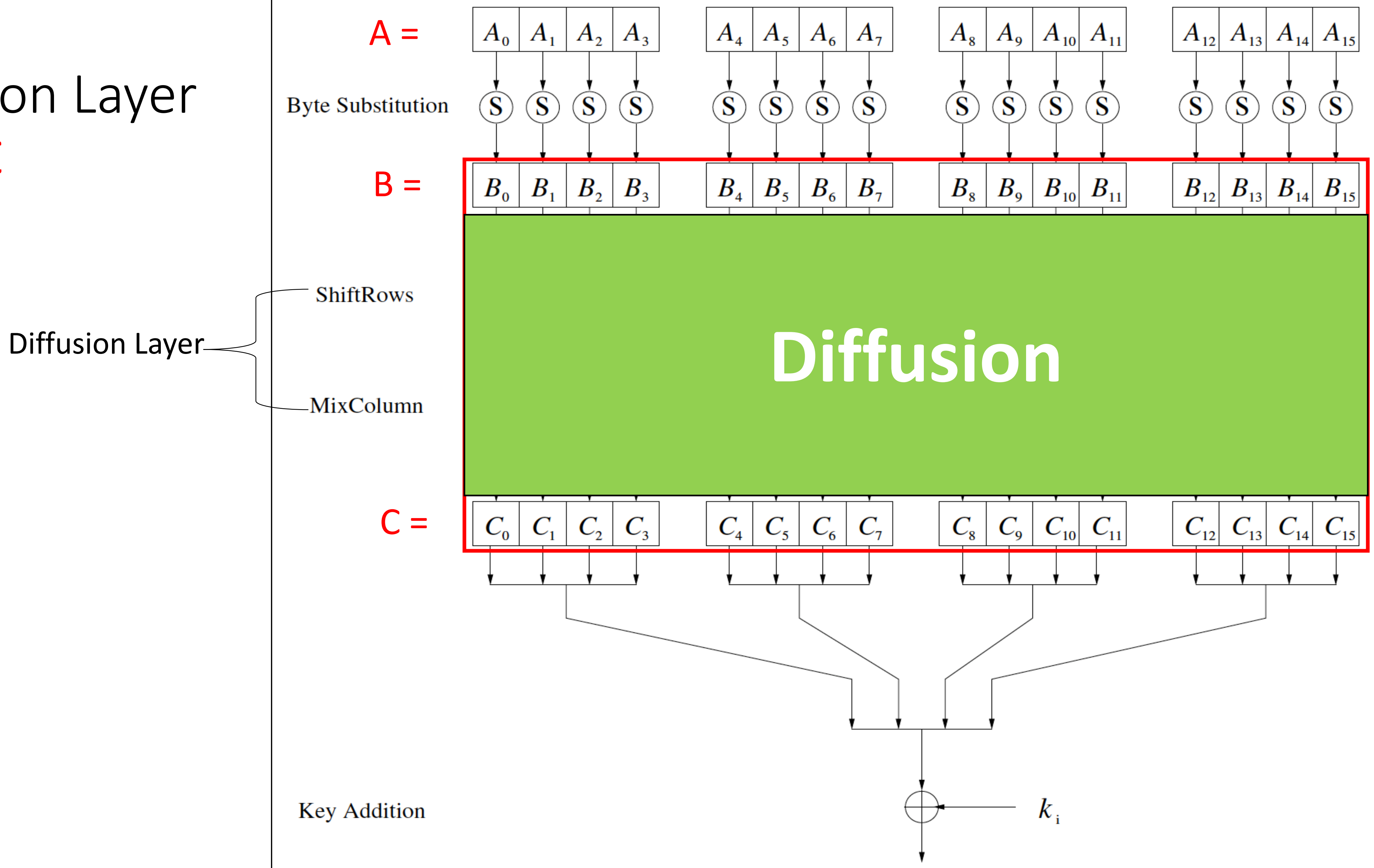
If A_0 = 00110111, what is B_0?

$$B_0 = S(\overbrace{0011}^{x}\overbrace{0111}^{y})$$

$$9A = 10011010$$

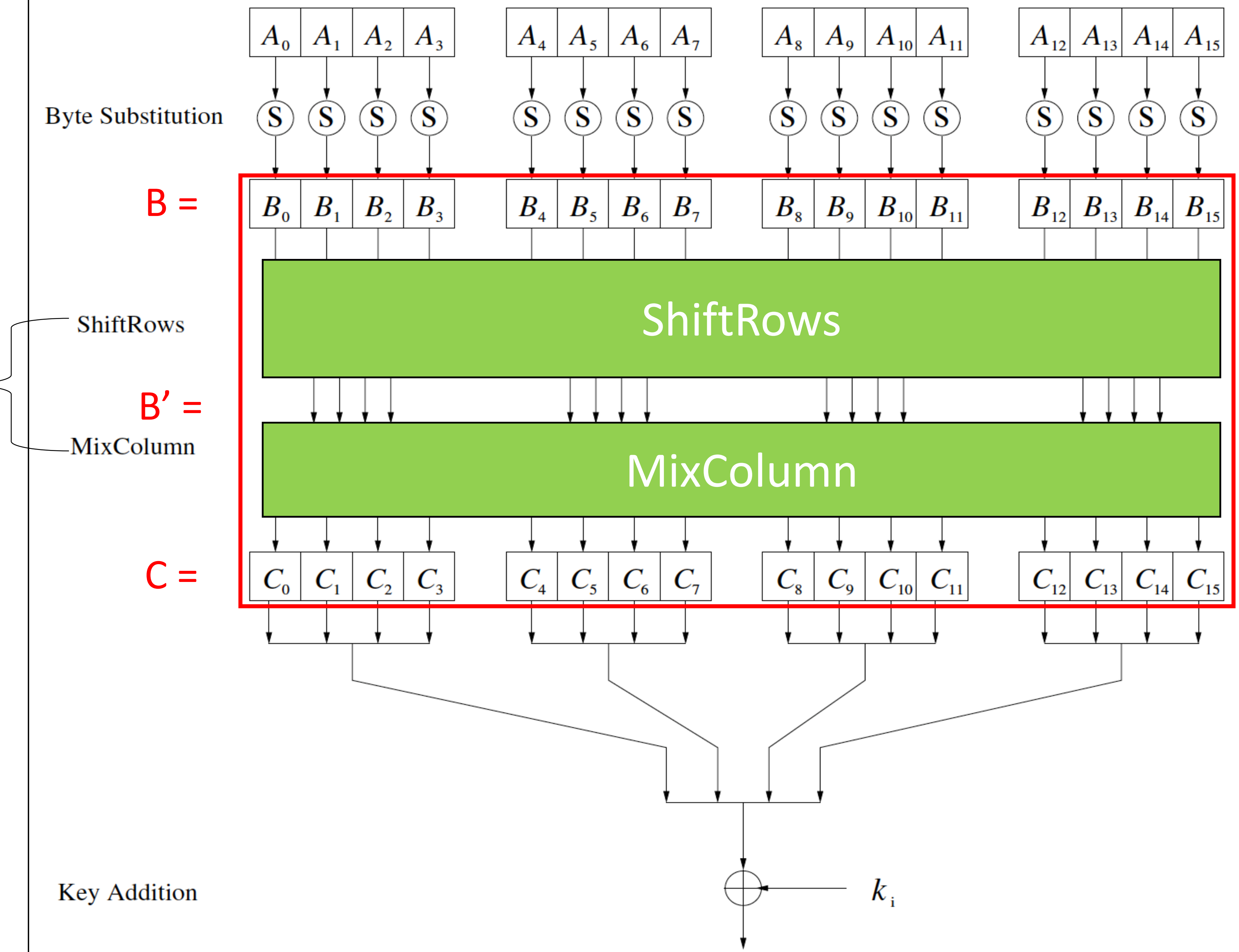
Diffusion Layer

B --> C



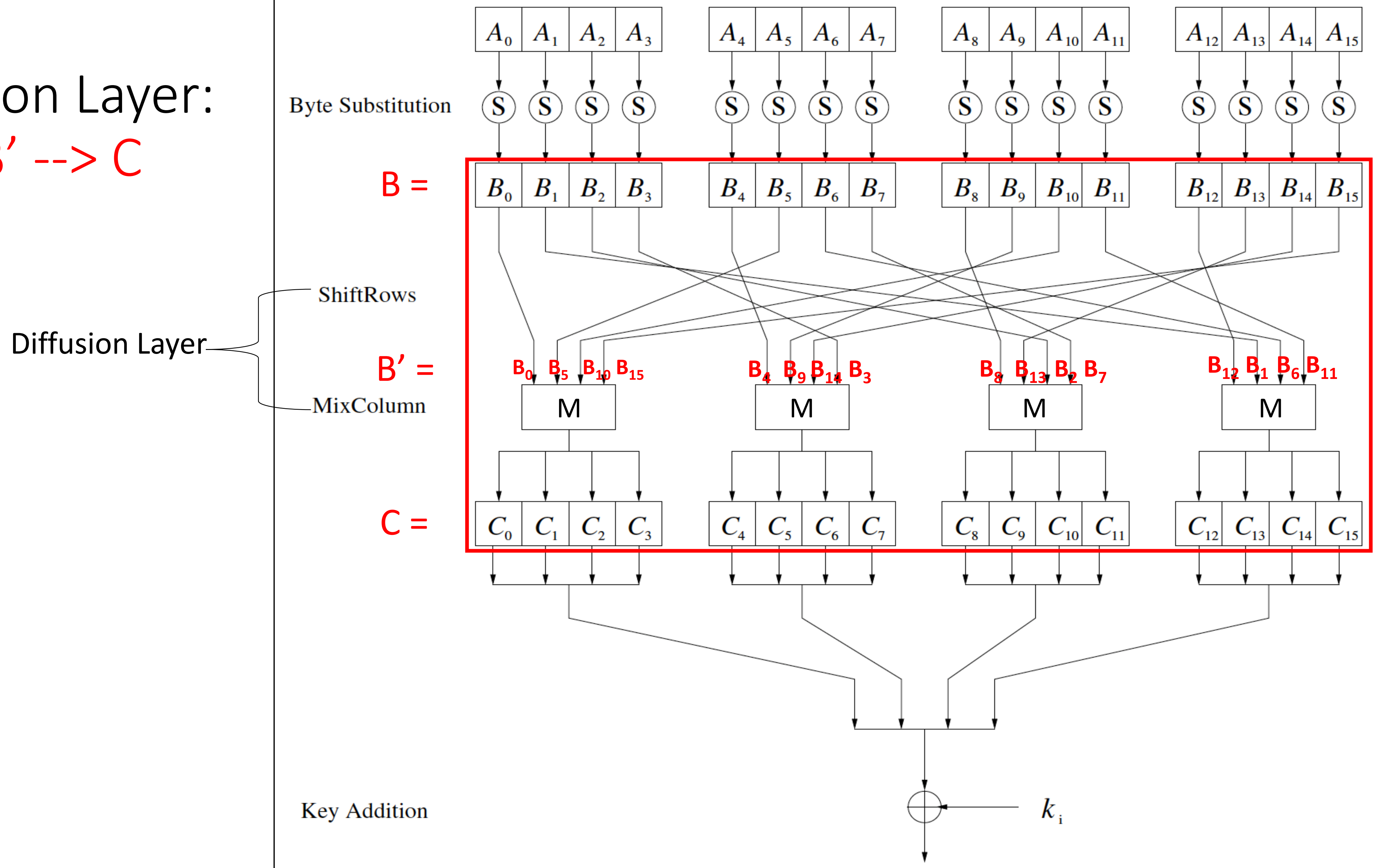
Diffusion Layer:

B --> B’ --> C



Diffusion Layer:

B --> B’ --> C



Diffusion Layer: B --> B’ --> C

In Diffusion Layer, 16 bytes in the path are arranged in a 4x4 matrix.

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}

B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

B

B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

ShiftRow

B’

B_0	B_4	B_8	B_{12}
B_5	B_9	B_{13}	B_1
B_{10}	B_{14}	B_2	B_6
B_{15}	B_3	B_7	B_{11}

MixColumn

C

C_0	C_4	C_8	C_{12}
C_1	C_5	C_9	C_{13}
C_2	C_6	C_{10}	C_{14}
C_3	C_7	C_{11}	C_{15}

ShiftRows: B → B’

B

B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

ShiftRow

B’

B_0	B_4	B_8	B_{12}
B_5	B_9	B_{13}	B_1
B_{10}	B_{14}	B_2	B_6
B_{15}	B_3	B_7	B_{11}

<-- No shift

<-- Left shift by 1 position

<-- Left shift by 2 positions

<-- Left shift by 3 positions

B

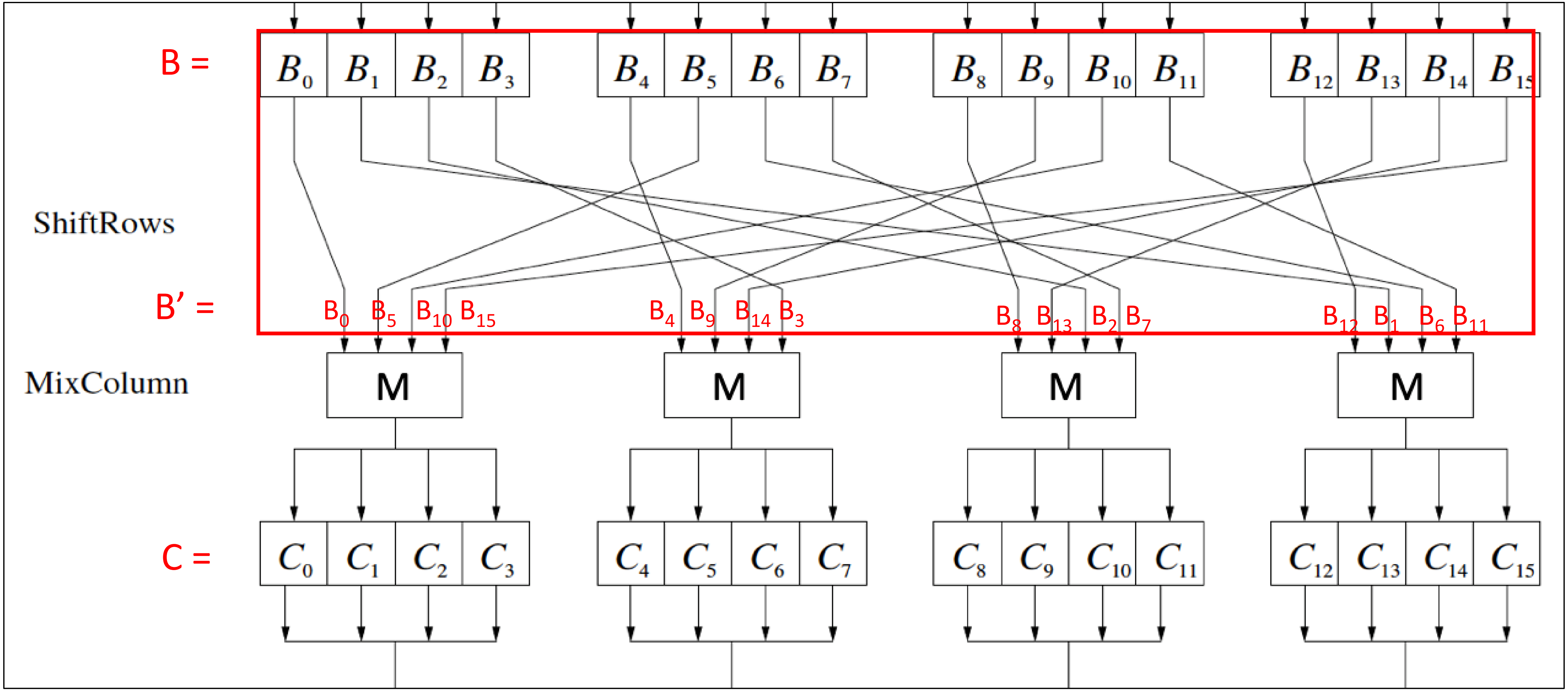
B_0 ,B_1, B_2 B_3, B_4 B_5, B_6 B_7, B_8 B_9, B_{10} B_{11}, B_{12} B_{13}, B_{14} B_{15}

ShiftRow

B’

B_0 B_5, B_{10} B_{15}, B_4 B_9, B_{14} B_3, B_8 B_{13}, B_2 B_7, B_{12} B_1, B_6 B_{11}

ShiftRows: B → B’



MixColumn: B' --> C

B'

B_0	B_4	B_8	B_{12}
B_5	B_9	B_{13}	B_1
B_{10}	B_{14}	B_2	B_6
B_{15}	B_3	B_7	B_{11}

$\xrightarrow{\text{MixColumn}}$

C

C_0	C_4	C_8	C_{12}
C_1	C_5	C_9	C_{13}
C_2	C_6	C_{10}	C_{14}
C_3	C_7	C_{11}	C_{15}

- A constant matrix M is used.
- MixColumn is a matrix multiplication:

C = M*B'

$$\underbrace{\begin{bmatrix} C_0 & C_4 & C_8 & C_{12} \\ C_1 & C_5 & C_9 & C_{13} \\ C_2 & C_6 & C_{10} & C_{14} \\ C_3 & C_7 & C_{11} & C_{15} \end{bmatrix}}_{C} = \underbrace{\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}}_{M} * \underbrace{\begin{bmatrix} B_0 & B_4 & B_8 & B_{12} \\ B_5 & B_9 & B_{13} & B_1 \\ B_{10} & B_{14} & B_2 & B_6 \\ B_{15} & B_3 & B_7 & B_{11} \end{bmatrix}}_{B'}$$

MixColumn: B' --> C

- You can also think the MixColumn layer as a column-wise operation:
  1. The first column of C is M * the first column of B'.
  2. The second column of C is M * the second column of B'.
  3. The third column of C is M * the third column of B'.
  4. The fourth column of C is M * the fourth column of B'.

$$\underbrace{\begin{bmatrix} C_0 & C_4 & C_8 & C_{12} \\ C_1 & C_5 & C_9 & C_{13} \\ C_2 & C_6 & C_{10} & C_{14} \\ C_3 & C_7 & C_{11} & C_{15} \end{bmatrix}}_{C} = \underbrace{\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}}_{M} * \underbrace{\begin{bmatrix} B_0 & B_4 & B_8 & B_{12} \\ B_5 & B_9 & B_{13} & B_1 \\ B_{10} & B_{14} & B_2 & B_6 \\ B_{15} & B_3 & B_7 & B_{11} \end{bmatrix}}_{B'}$$

$$\begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} * \begin{bmatrix} B_0 \\ B_5 \\ B_{10} \\ B_{15} \end{bmatrix}$$

MixColumn: B’ --> C

$$\underset{C}{\begin{bmatrix} C_0 & C_4 & C_8 & C_{12} \\ C_1 & C_5 & C_9 & C_{13} \\ C_2 & C_6 & C_{10} & C_{14} \\ C_3 & C_7 & C_{11} & C_{15} \end{bmatrix}} = \underset{M}{\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}} * \underset{B'}{\begin{bmatrix} B_0 & B_4 & B_8 & B_{12} \\ B_5 & B_9 & B_{13} & B_1 \\ B_{10} & B_{14} & B_2 & B_6 \\ B_{15} & B_3 & B_7 & B_{11} \end{bmatrix}}$$

$$\begin{bmatrix} C_4 \\ C_5 \\ C_6 \\ C_7 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} * \begin{bmatrix} B_4 \\ B_9 \\ B_{14} \\ B_3 \end{bmatrix}$$

$$\underset{C}{\begin{bmatrix} C_0 & C_4 & C_8 & C_{12} \\ C_1 & C_5 & C_9 & C_{13} \\ C_2 & C_6 & C_{10} & C_{14} \\ C_3 & C_7 & C_{11} & C_{15} \end{bmatrix}} = \underset{M}{\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}} * \underset{B'}{\begin{bmatrix} B_0 & B_4 & B_8 & B_{12} \\ B_5 & B_9 & B_{13} & B_1 \\ B_{10} & B_{14} & B_2 & B_6 \\ B_{15} & B_3 & B_7 & B_{11} \end{bmatrix}}$$

$$\begin{bmatrix} C_8 \\ C_9 \\ C_{10} \\ C_{11} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} * \begin{bmatrix} B_8 \\ B_{13} \\ B_2 \\ B_7 \end{bmatrix}$$

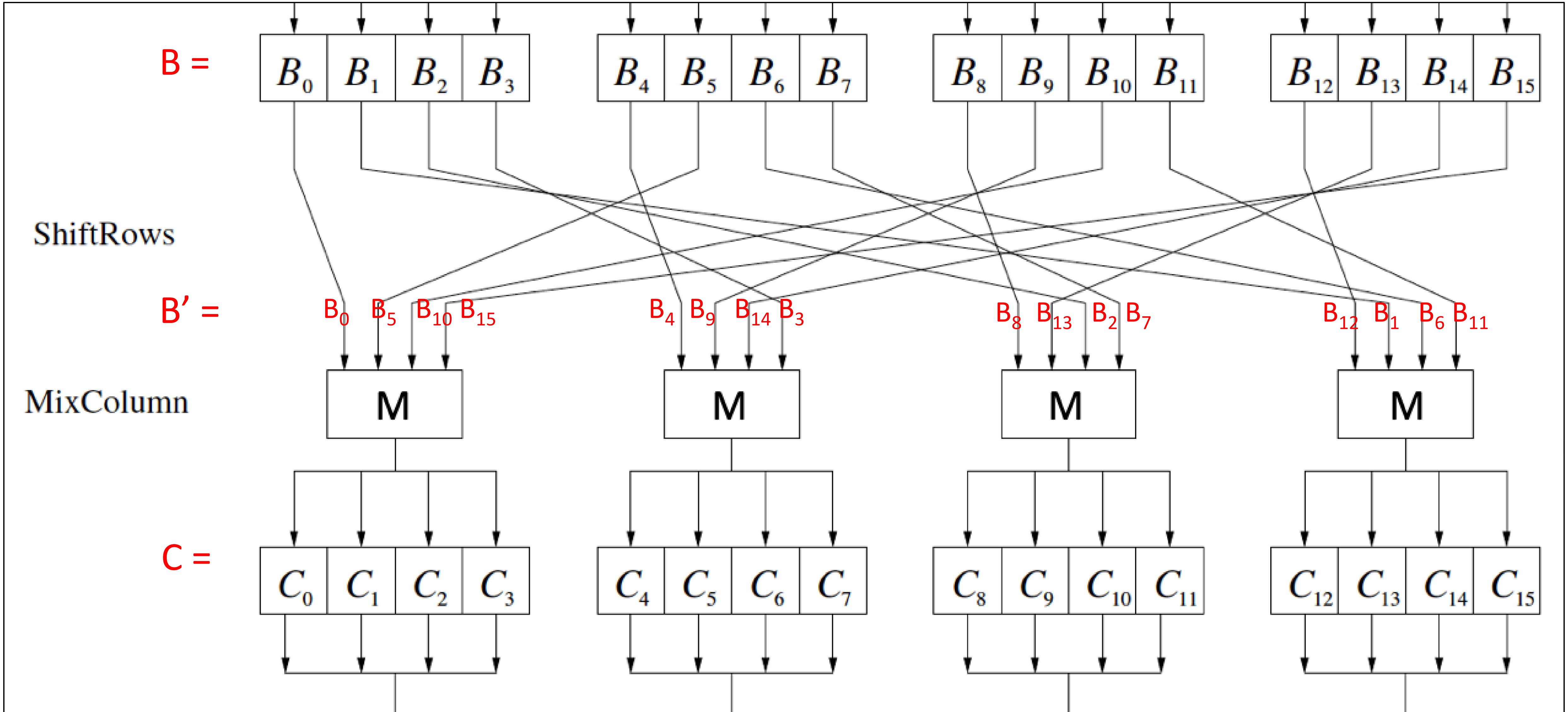
MixColumn: B' --> C

- Compute the first four bytes in C: C_0, C_1, C_2, C_3

$$\underset{C}{\begin{bmatrix} C_0 & C_4 & C_8 & C_{12} \\ C_1 & C_5 & C_9 & C_{13} \\ C_2 & C_6 & C_{10} & C_{14} \\ C_3 & C_7 & C_{11} & C_{15} \end{bmatrix}} = \underset{M}{\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix}} * \underset{B'}{\begin{bmatrix} B_0 & B_4 & B_8 & B_{12} \\ B_5 & B_9 & B_{13} & B_1 \\ B_{10} & B_{14} & B_2 & B_6 \\ B_{15} & B_3 & B_7 & B_{11} \end{bmatrix}}$$

$$\begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{matrix} 02 * B_0 + 03 * B_5 + 01 * B_{10} + 01 * B_{15} \\ 01 * B_0 + 02 * B_5 + 04 * B_{10} + 01 * B_{15} \\ 01 * B_0 + 01 * B_5 + 02 * B_{10} + 03 * B_{15} \\ 03 * B_0 + 01 * B_5 + 01 * B_{10} + 02 * B_{15} \end{matrix}$$

MixColumn: B' --> C



- Compute C_0 when
  - B_0 = 00
  - B_5 = A4
  - B_{10} = 8C
  - B_{15} = 00

$$\begin{bmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} * \begin{bmatrix} B_0 \\ B_5 \\ B_{10} \\ B_{15} \end{bmatrix}$$

- Compute C_0 when $B_0 = 25$, $B_5 = A4$, $B_{10} = 8C$, $B_{15} = 61$.
- $C_0 = 02*B_0+03*B_5+01*B_{10}+01*B_{15}$

  $=02*25+03*A4+01*8C+01*61$
- $02*25 = x * (x^5+x^2+1)$

  $=x^6+x^3+x$
- $03*A4 = (x+1)*(x^7+x^5+x^2)$

  $=x^8+x^7+x^6+x^5+x^3+x^2$

  $=(x^4+x^3+x+1)+x^7+x^6+x^5+x^3+x^2$

  $=x^7+x^6+x^5+x^4+x^2+x+1$
- $01*8C = 8C = x^7+x^3+x^2$
- $01*61 = 61 = x^6+x^5+1$

$$\begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix} = \begin{pmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{pmatrix} \begin{pmatrix} B_0 \\ B_5 \\ B_{10} \\ B_{15} \end{pmatrix}$$

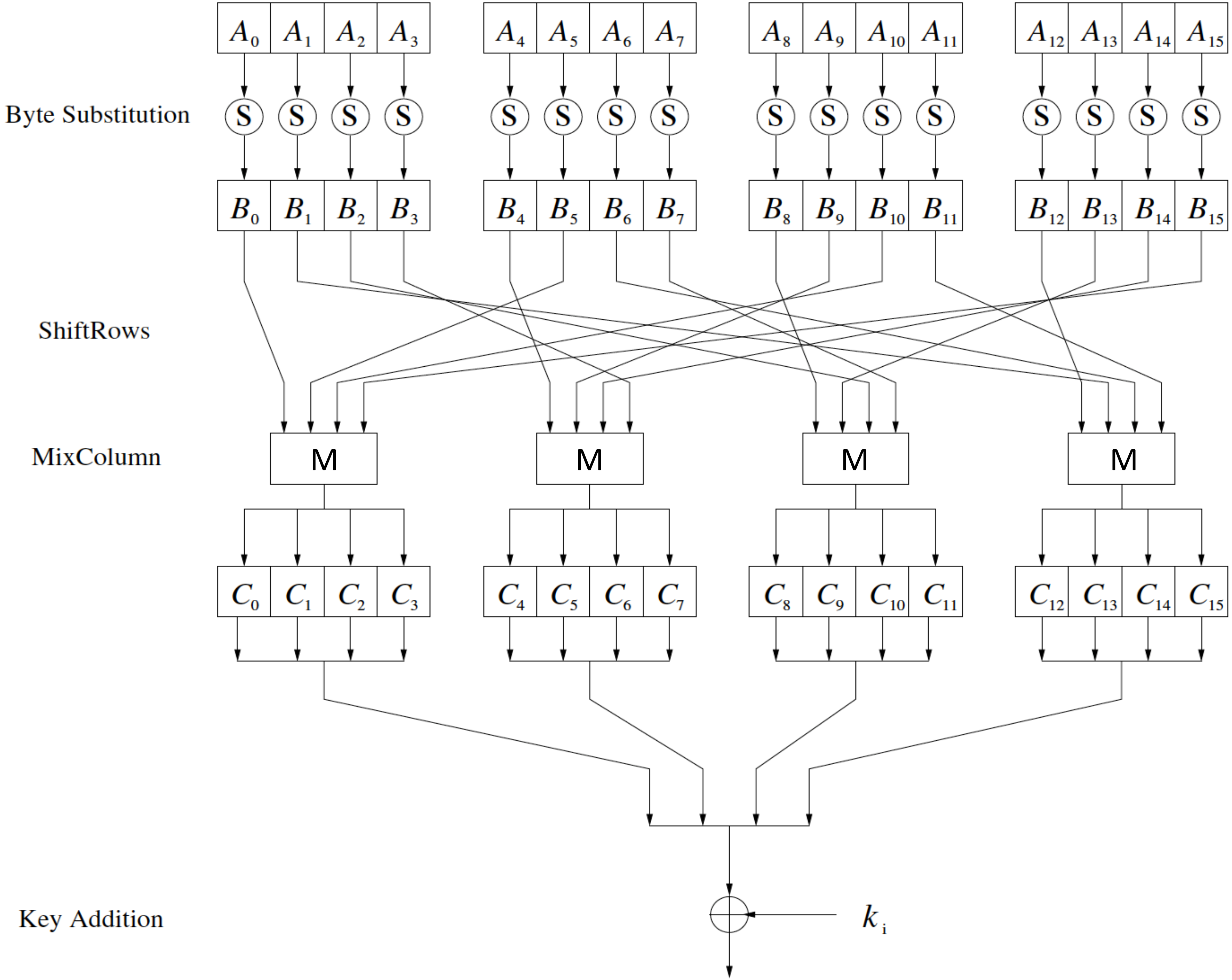
$$\begin{array}{rl} & x^6+x^3+x \\ & x^7+x^6+x^5+x^4+x^2+x+1 \\ & x^7+x^3+x^2 \\ + & x^6+x^5+1 \\ \hline \end{array}$$

$C_0 = x^6+x^4 = (0, 1, 0, 1, 0,0,0,0) = 50$

AES Encryption Round

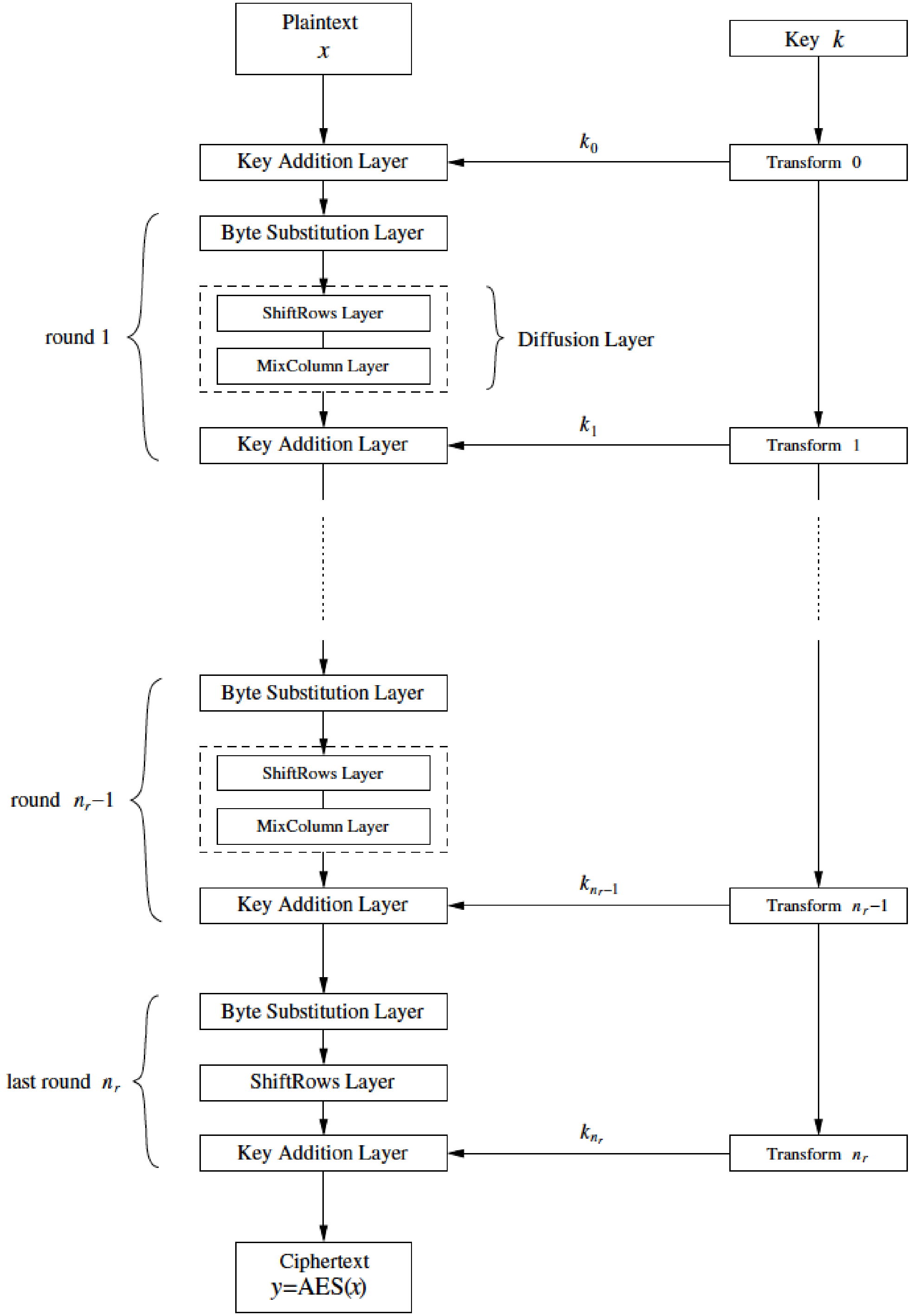
02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02

M



Key Addition Layer

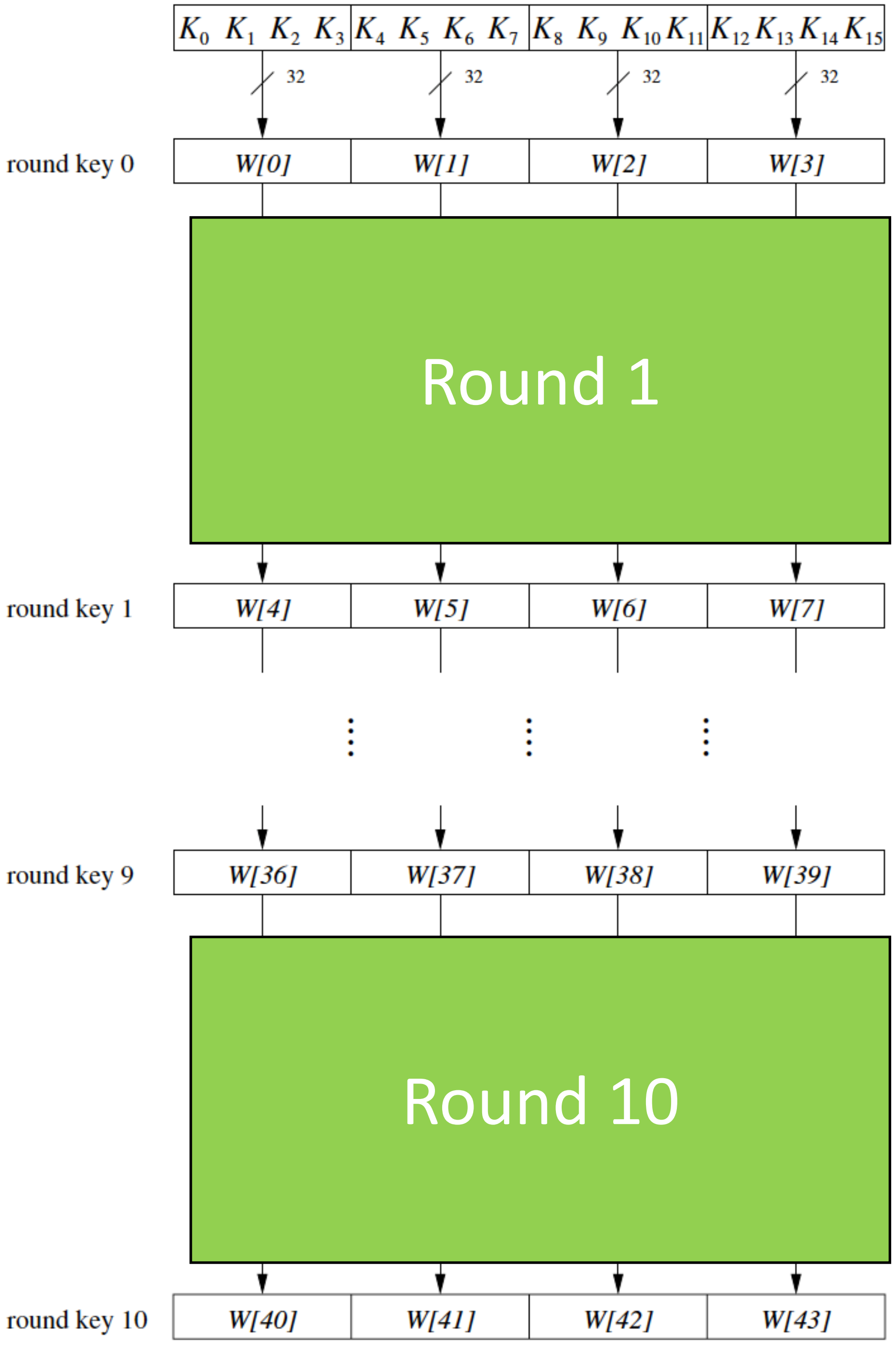
- The two inputs to the Key Addition layer are the current 16-byte state matrix and a subkey which also consists of 16 bytes (128 bits).
- The two inputs are combined through a bitwise **XOR** operation. Note that the XOR operation is equal to addition in the Galois field GF(2) .



Key Schedule for 128-Bit Key AES

Key Schedule is word-oriented.
128 bits = 4 words

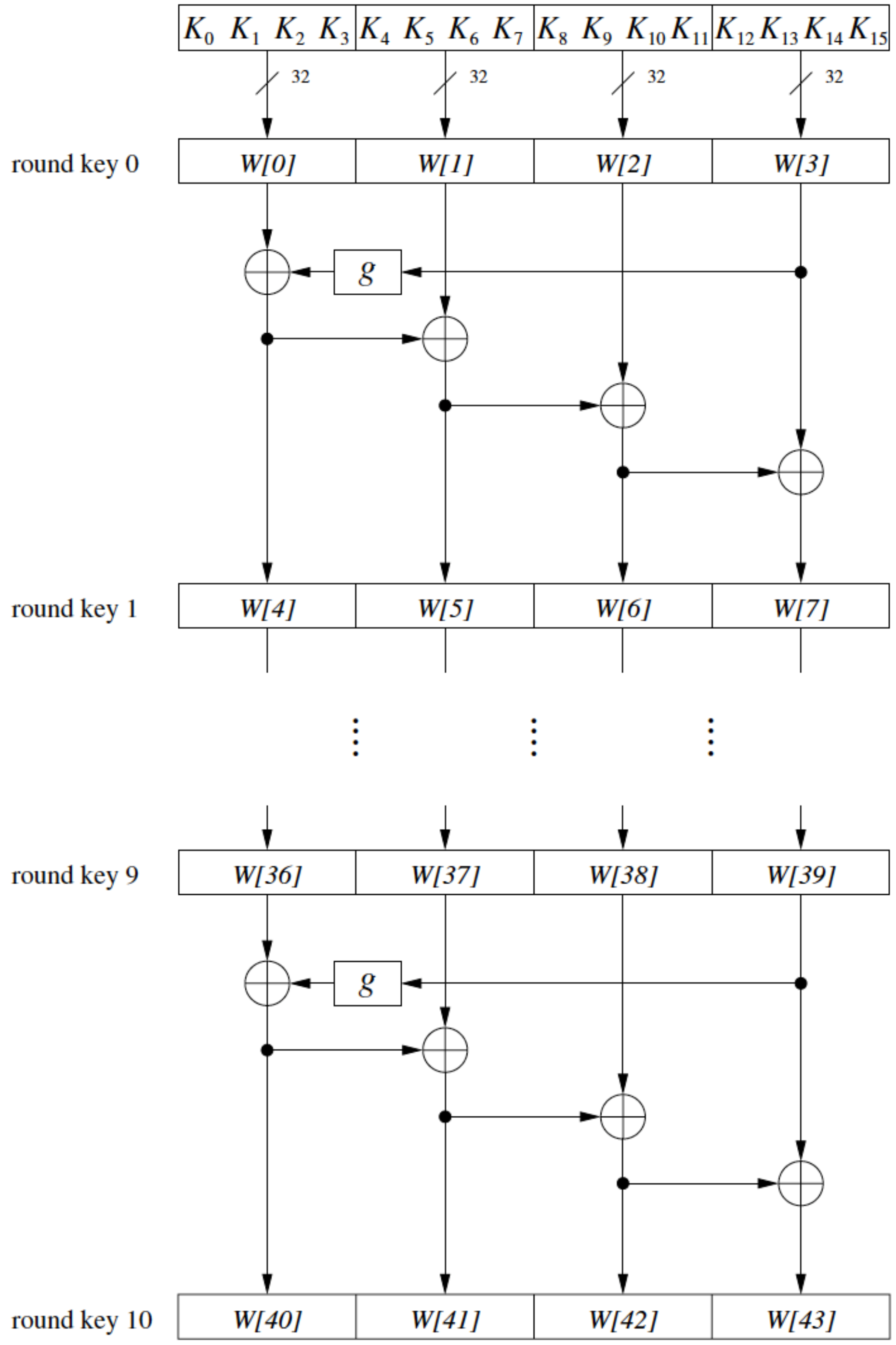
Initial Key: W[0], W[1], W[2], W[3].
Round key1: W[4], W[5], W[6], W[7]
Round key2: W[8], W[9], W[10], W[11],
....
Round key10: W[40], W[41], W[42], W[43]



Key Schedule for 128-Bit Key AES

Key Schedule is word-oriented.
128 bits = 4 words

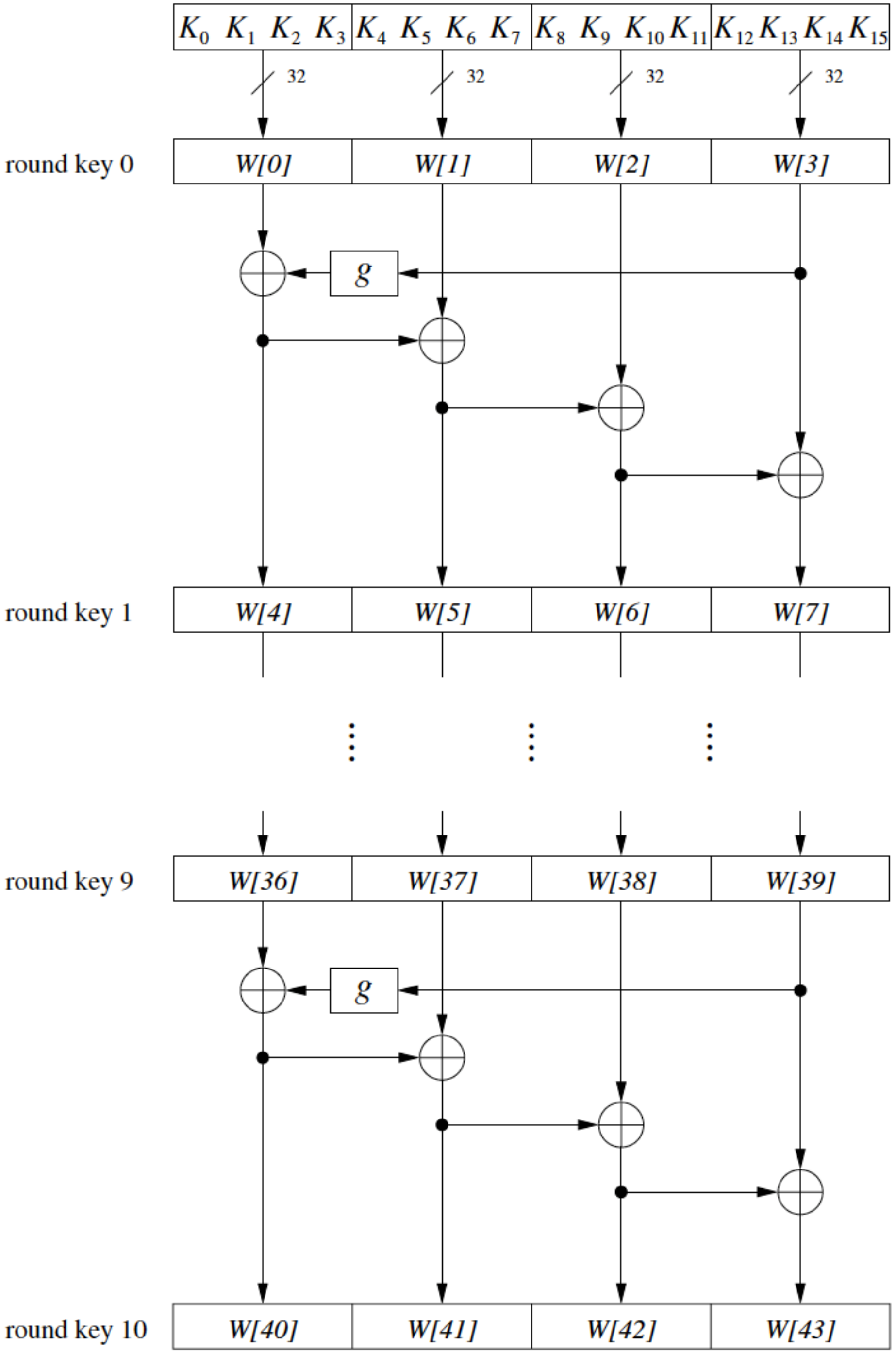
Initial Key: W[0], W[1], W[2], W[3].
Round key1: W[4], W[5], W[6], W[7]
Round key2: W[8], W[9], W[10], W[11],
....
Round key10: W[40], W[41], W[42], W[43]

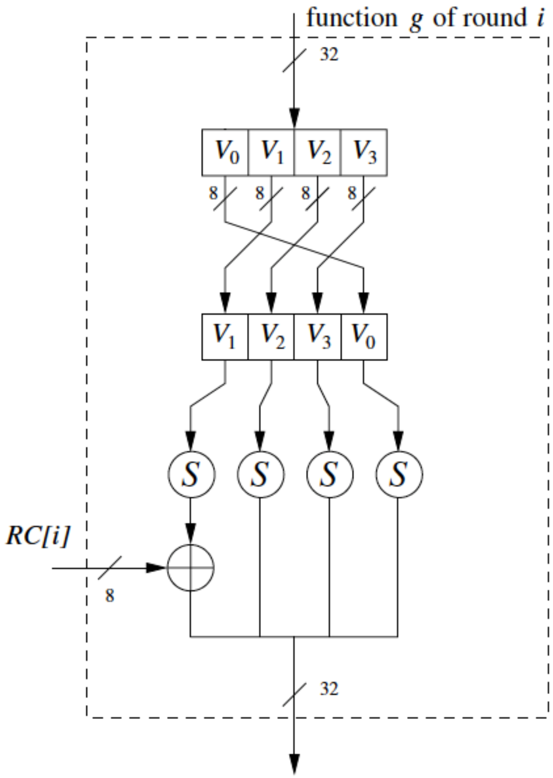


Key Schedule for 128-Bit Key AES

The leftmost word of a subkey
W[4i] , where i = 1, . . . , 10, is computed as:
W[4i] =W[4(i− 1)]+g(W[4i− 1])

Here g() is a nonlinear function with a four-byte input and output. The remaining three words of a subkey are computed recursively as:
W[4i+ j] = W[4i+ j− 1]+W[4(i− 1)+ j],
where i = 1, . . . , 10 and j = 1, 2, 3.





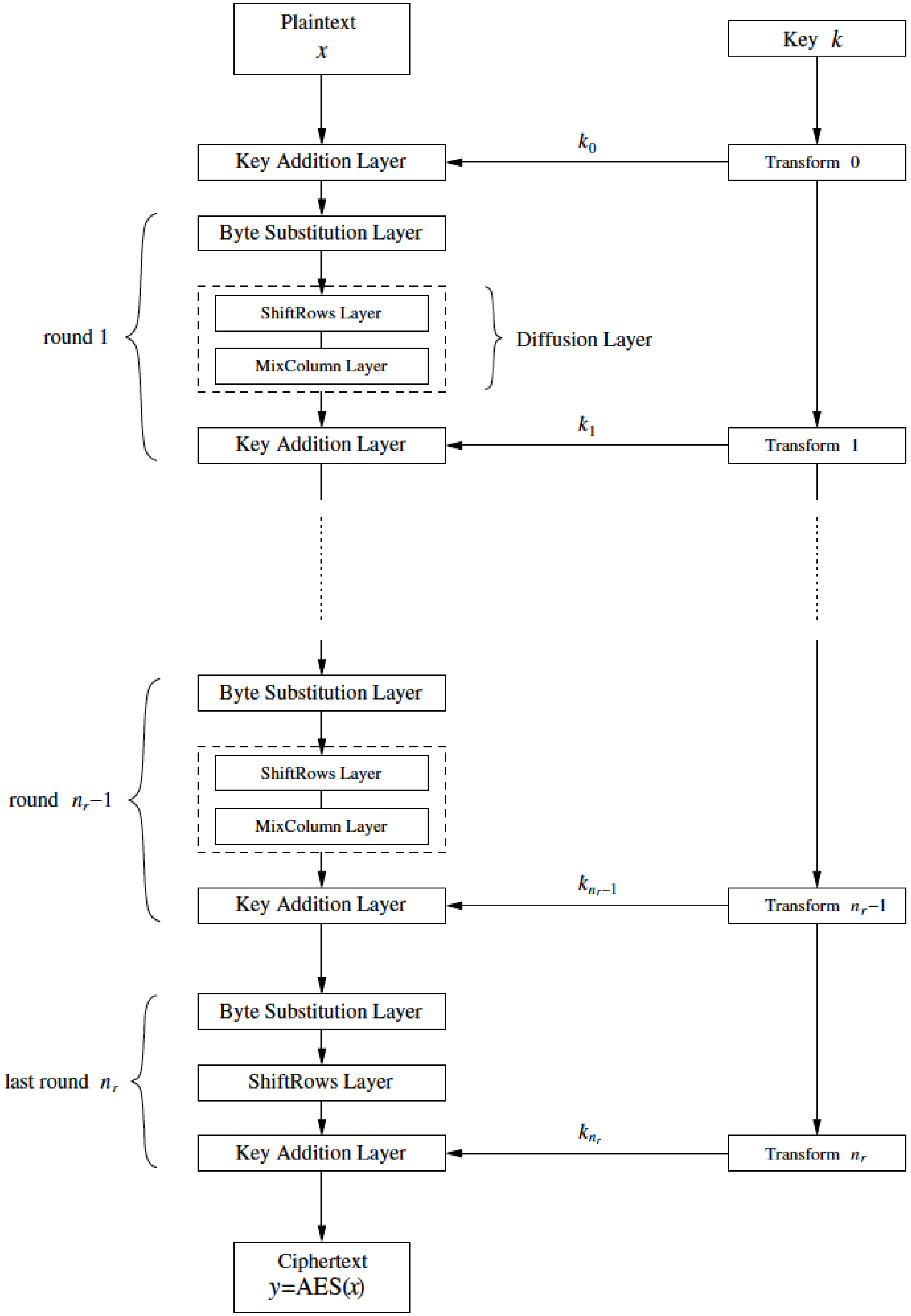
The g-function

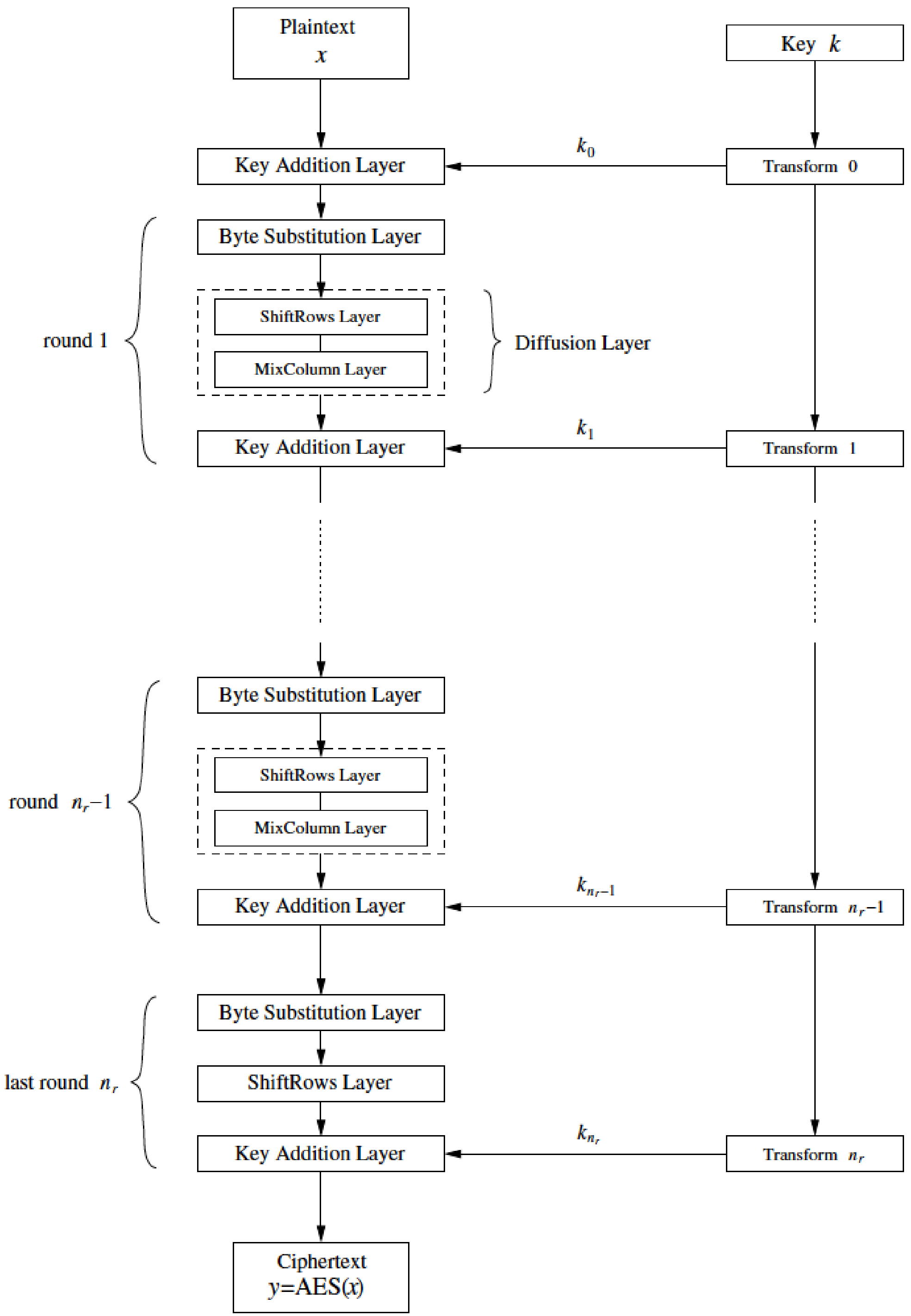
- The function g() rotates its four input bytes, performs a byte-wise S-Box substitution, and adds a round coefficient RC to it.
- The round coefficient is an element of the Galois field GF(2^8) , i.e, an 8-bit value. It is only added to the leftmost byte in the function g() .
- The round coefficients vary from round to round according to the following rule:

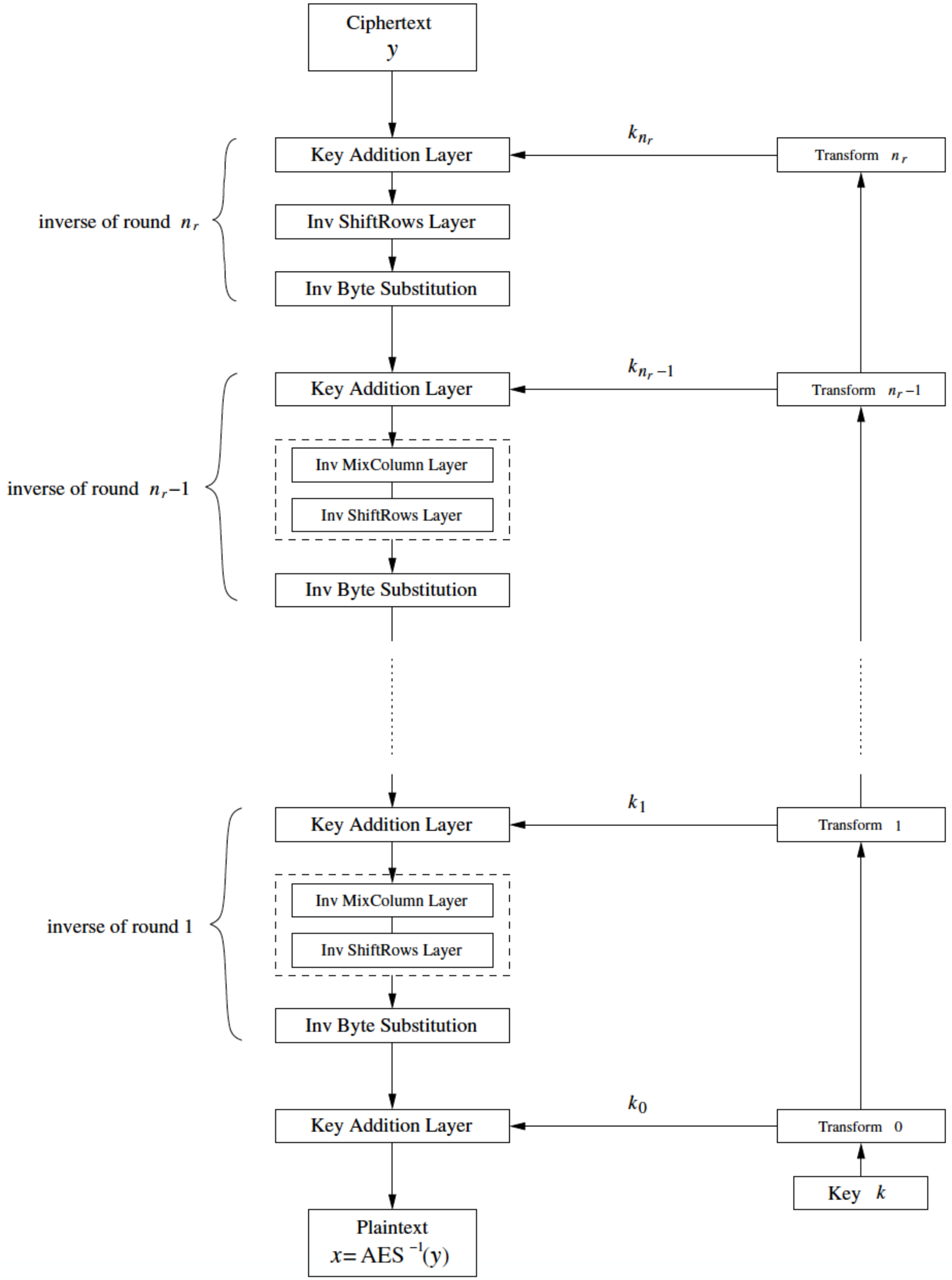
$$\begin{aligned} RC[1] &= x^0 = (0000\,0001)_2, \\ RC[2] &= x^1 = (0000\,0010)_2, \\ RC[3] &= x^2 = (0000\,0100)_2, \\ &\vdots \\ RC[10] &= x^9 = (0011\,0110)_2. \end{aligned}$$

Decryption

- Because AES is not based on a Feistel network, all layers must actually be inverted.
  - The **Byte Substitution** layer becomes **the Inv Byte Substitution** layer.
  - The **ShiftRows** layer becomes the **Inv ShiftRows** layer.
  - The **MixColumn** layer becomes **Inv MixColumn** layer.


Plaintext
x
Key k
Key Addition Layer
k0
Transform 0
Byte Substitution Layer
round 1
ShiftRows Layer
MixColumn Layer
Diffusion Layer
Key Addition Layer
k1
Transform 1
Byte Substitution Layer
round nr−1
ShiftRows Layer
MixColumn Layer
Key Addition Layer
knr−1
Transform nr−1
Byte Substitution Layer
last round nr
ShiftRows Layer
Key Addition Layer
knr
Transform nr
Ciphertext
y=AES(x)

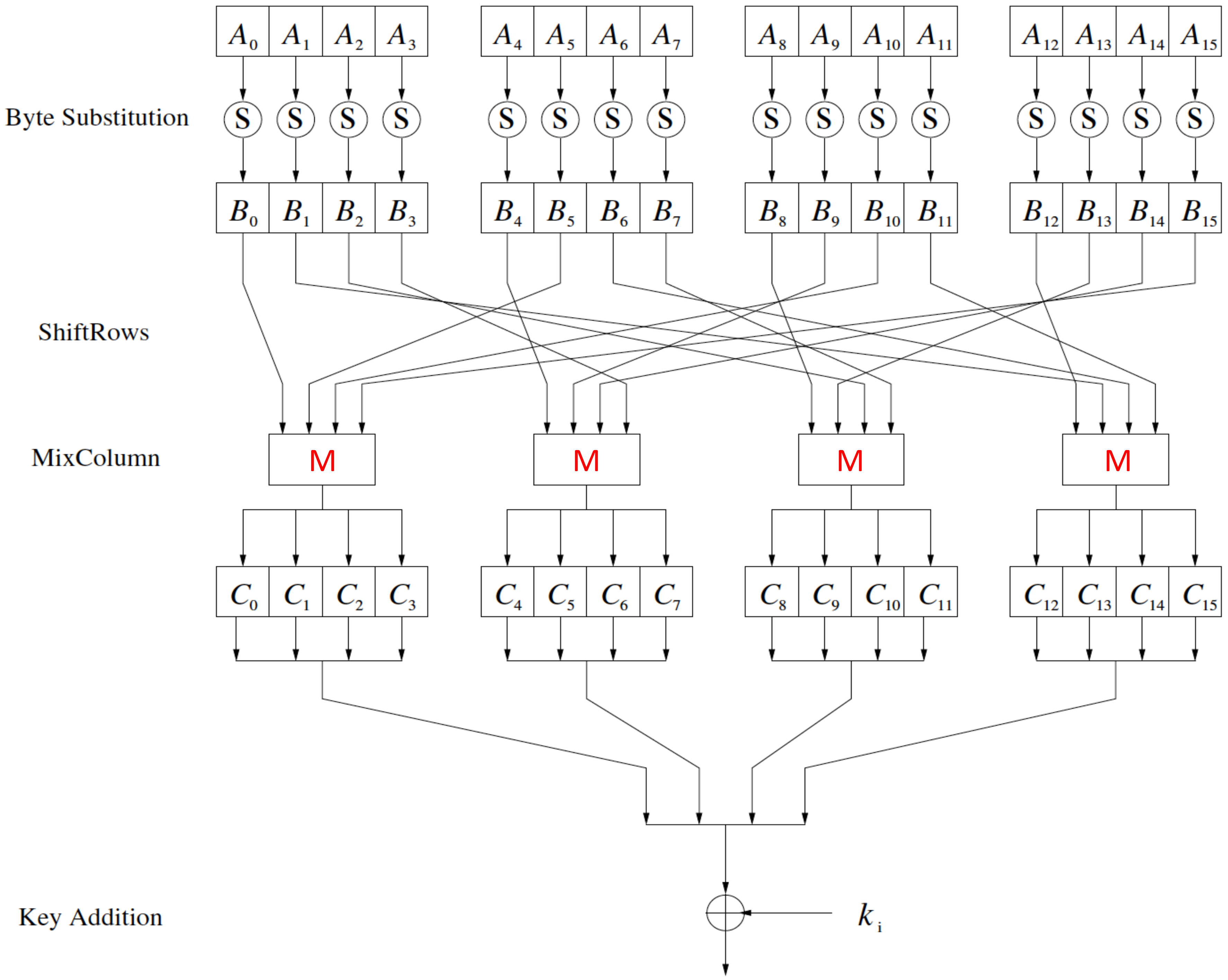

Plaintext
x
Key k
Key Addition Layer
k_0
Transform 0
Byte Substitution Layer
round 1
ShiftRows Layer
MixColumn Layer
Diffusion Layer
Key Addition Layer
k_1
Transform 1
Byte Substitution Layer
round $n_r - 1$
ShiftRows Layer
MixColumn Layer
Key Addition Layer
$k_{n_r - 1}$
Transform $n_r - 1$
Byte Substitution Layer
last round n_r
ShiftRows Layer
Key Addition Layer
k_{n_r}
Transform n_r
Ciphertext
y=AES(x)


Ciphertext
y
Key Addition Layer
k_{n_r}
Transform n_r
inverse of round n_r
Inv ShiftRows Layer
Inv Byte Substitution
Key Addition Layer
$k_{n_r - 1}$
Transform $n_r - 1$
inverse of round $n_r - 1$
Inv MixColumn Layer
Inv ShiftRows Layer
Inv Byte Substitution
Key Addition Layer
k_1
Transform 1
inverse of round 1
Inv MixColumn Layer
Inv ShiftRows Layer
Inv Byte Substitution
Key Addition Layer
k_0
Transform 0
Key k
Plaintext
$x = AES^{-1}(y)$

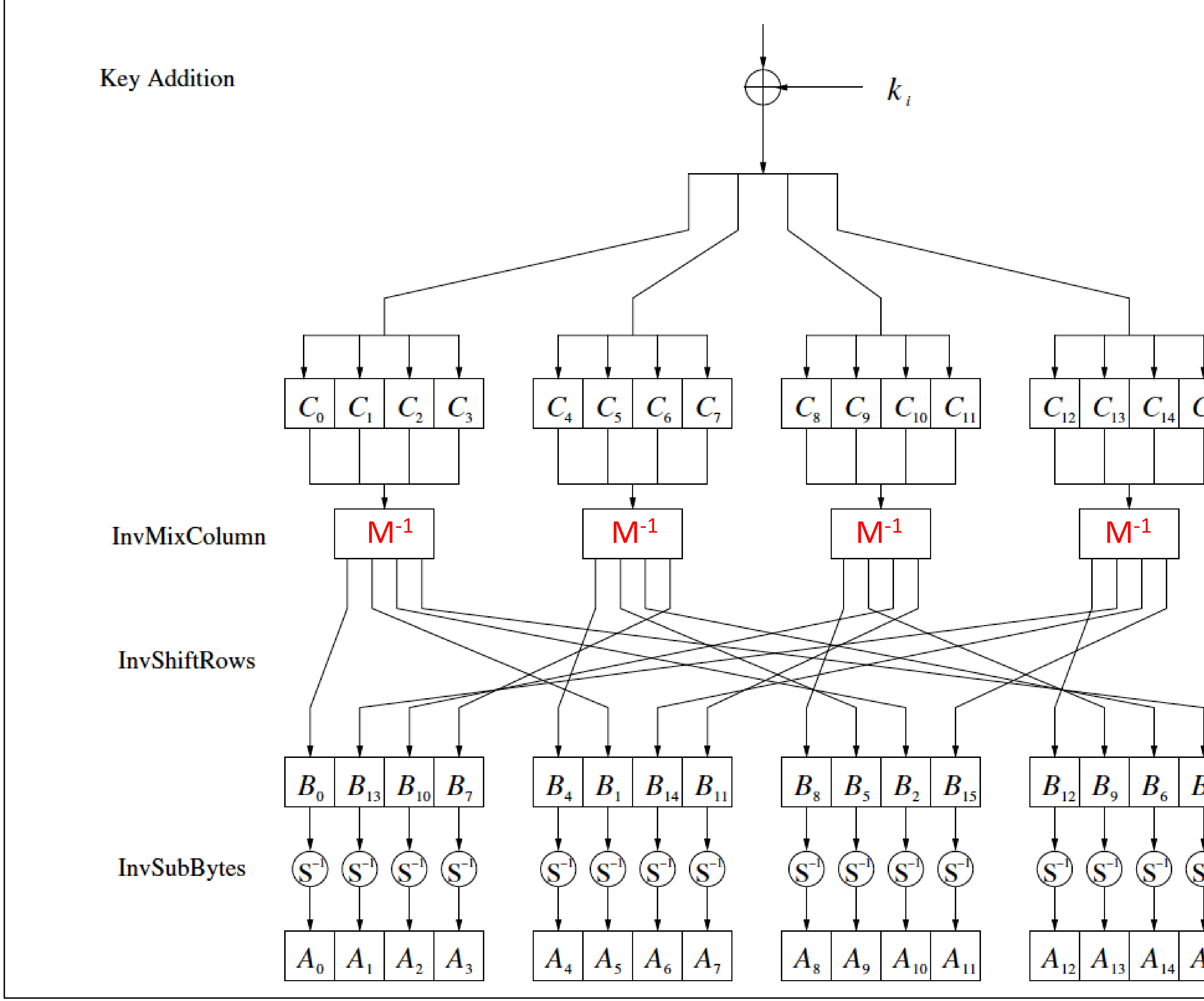
Review: AES Encryption Round Function

M =

02	03	01	01
01	02	03	01
01	01	02	03
03	01	01	02



AES Decryption Round Function



$$M^{-1} = \begin{pmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{pmatrix}$$

Inverse ShiftRows Sublayer

B_0	B_4	B_8	B_{12}
B_1	B_5	B_9	B_{13}
B_2	B_6	B_{10}	B_{14}
B_3	B_7	B_{11}	B_{15}

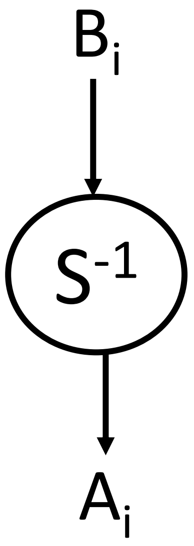
$\Rightarrow$

B_0	B_4	B_8	B_{12}	no shift
B_{13}	B_1	B_5	B_9	$\longrightarrow$ one position right shift
B_{10}	B_{14}	B_2	B_6	$\longrightarrow$ two positions right shift
B_7	B_{11}	B_{15}	B_3	$\longrightarrow$ three positions right shift

Inverse Byte Substitution Layer

x \ 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
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

Inverse AES S-Box



Decryption Key Schedule

- Since the first decryption round needs the last subkey, the second decryption round needs the second-to-last subkey and so on, we need the subkey in reversed order.
- In practice this is mainly achieved by computing the entire key schedule first and storing all 11 (13 or 15) subkeys, depending on the number or rounds AES is using (which in turn depends on the three key lengths supported by AES)