HUST

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI

HANOI UNIVERSITY OF SCIENCE AND TECHNOLOGY

ONE LOVE. ONE FUTURE.

TRƯỜNG ĐẠI HỌC
BÁCH KHOA HÀ NỘI
HANOI UNIVERSITY
OF SCIENCE AND TECHNOLOGY

LÝ THUYẾT MẬT MÃ

Cryptography Theory

ET3310

PGS. TS. Đỗ Trọng Tuấn
Trường Điện - Điện tử * Đại học Bách Khoa Hà Nội

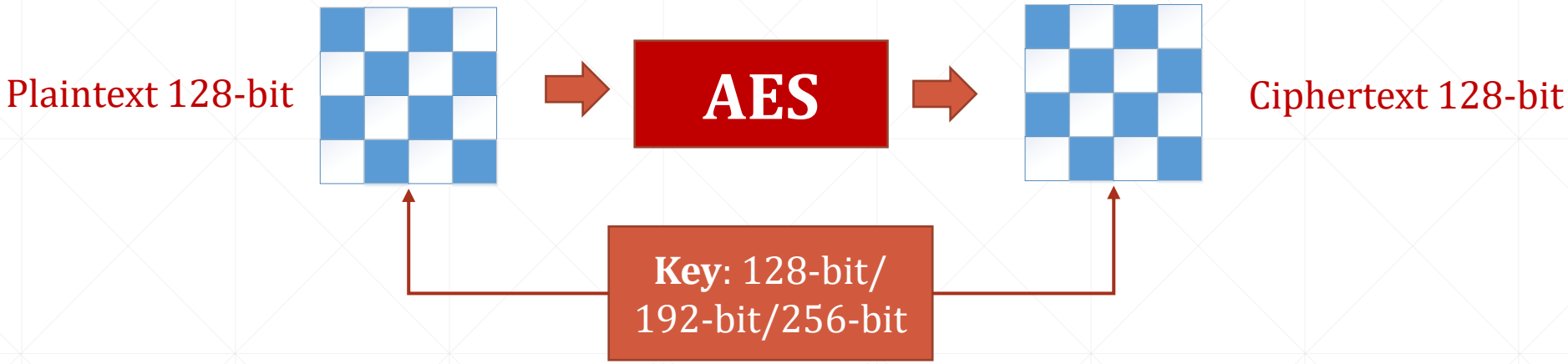
ONE LOVE. ONE FUTURE.

Advanced Encryption Standard

Contents

1. Introduction to AES cipher
2. Transformations
3. Key Expansion
4. Ciphers
5. Examples
6. Analysis of AES

1 Introduction to AES cipher



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.

1 Introduction to AES cipher

❖ Criteria

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

- *Security*
- *Cost*
- *Implementation*

1 Introduction to AES cipher

❖ 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.

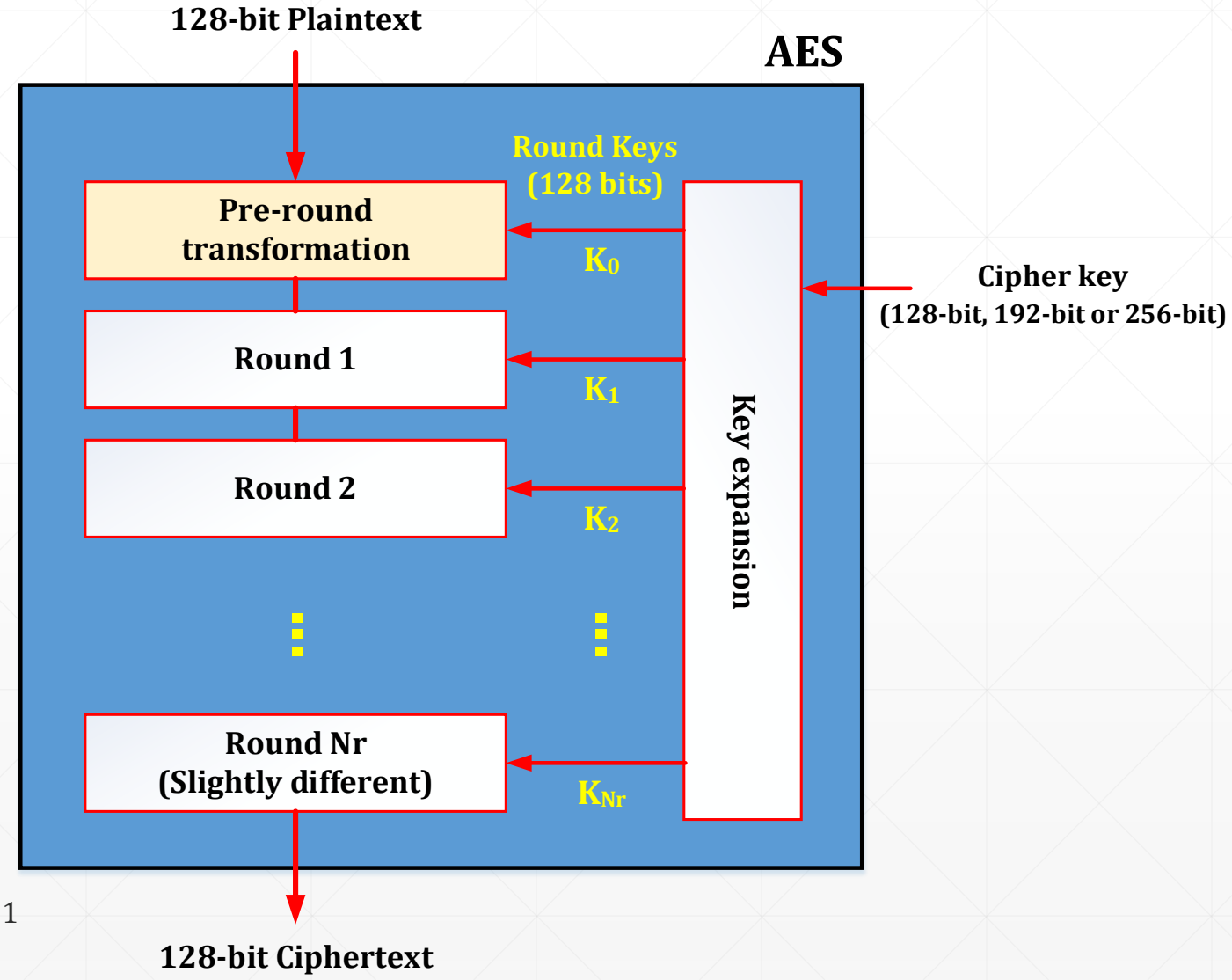
AES has defined three versions, with 10, 12, and 14 rounds. Each version uses a different cipher key size (128, 192, or 256), but the round keys are always 128 bits.

No.	AES Cipher	Round Nr
1	AES-128	10
2	AES-192	12
3	AES-256	14

1 Introduction to AES cipher

❖ Rounds

General design of AES encryption cipher

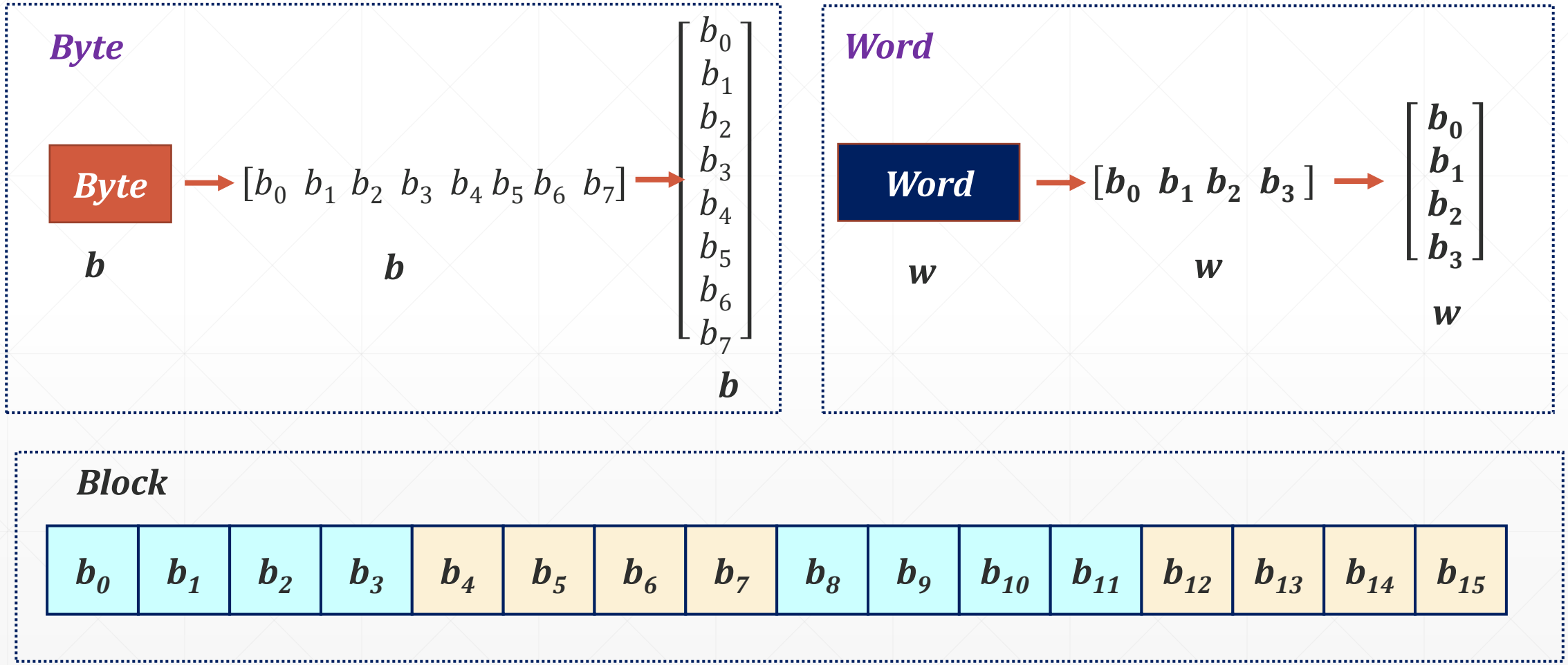


Relationship between number of rounds and cipher key size.

Nr	Key size
10	128
12	192
14	256

1 Introduction to AES cipher

❖ Data Units



1 Introduction to AES cipher

❖ Data Units

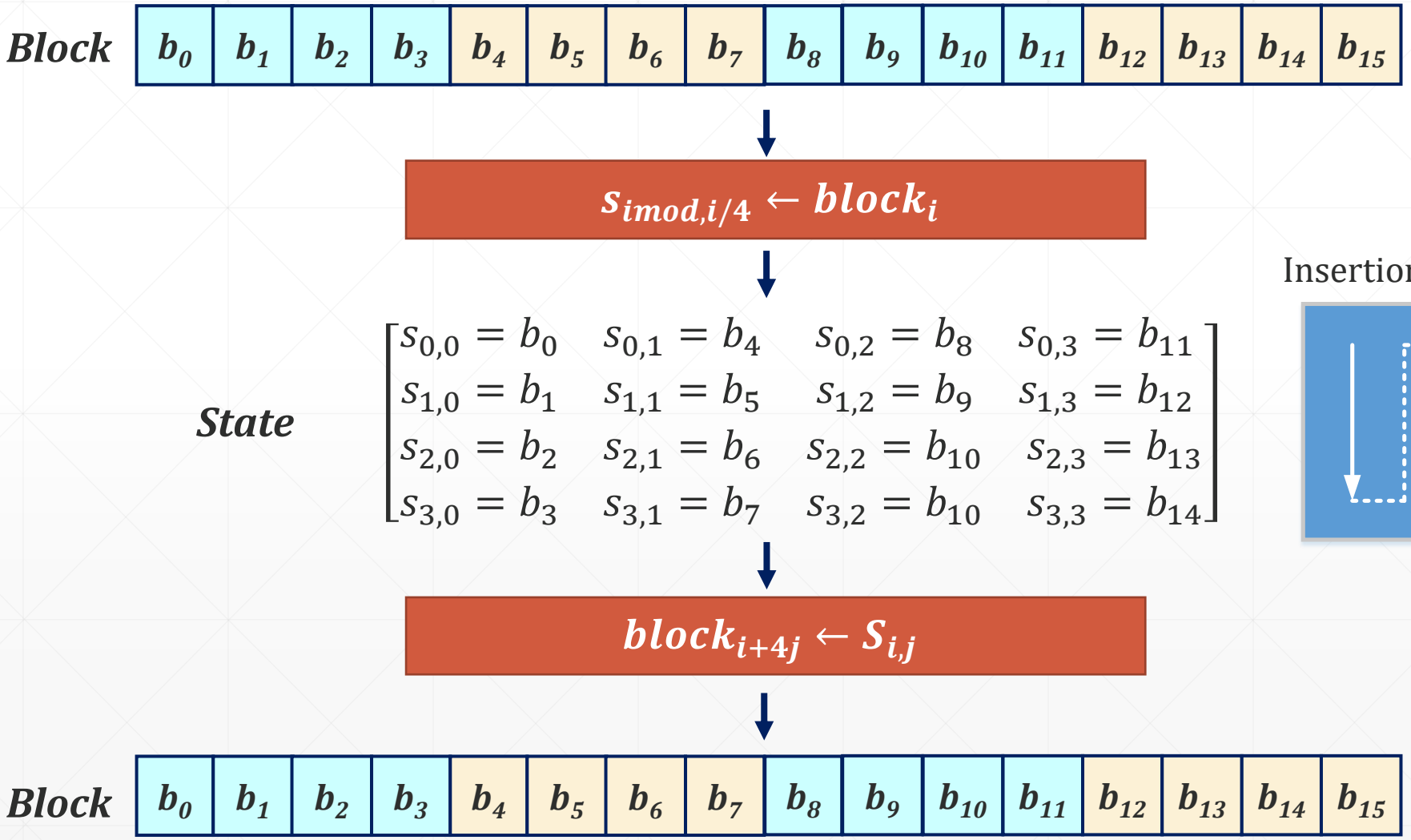
State

$$S \rightarrow \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} \rightarrow [w_0 \; w_1 \; w_2 \; w_3]$$

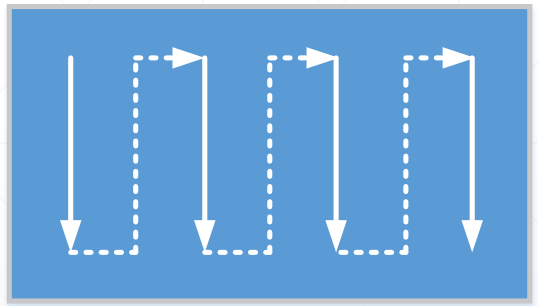
1 Introduction to AES cipher

❖ Data Units

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



Insertion and extraction flow

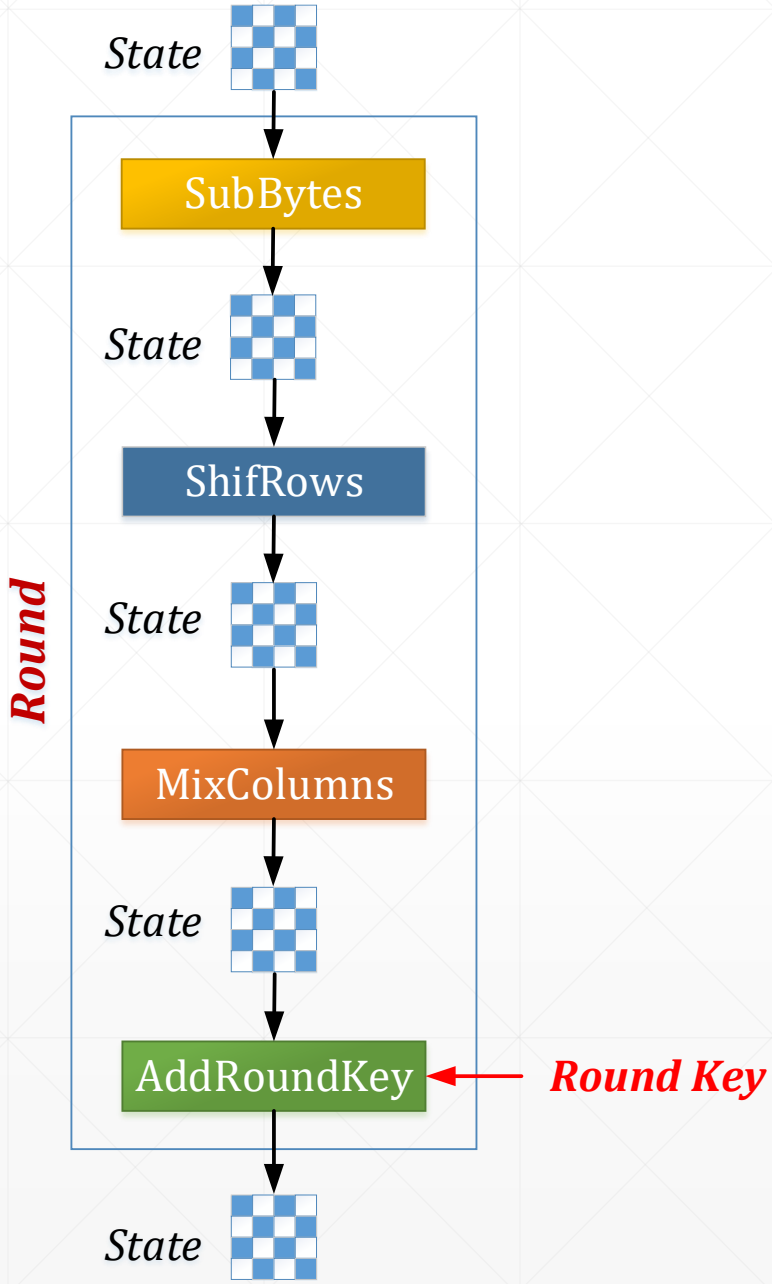


1 Introduction to AES cipher

❖ Structure of Each Round

Structure of each round at the encryption site

- *One AddRoundKey is applied before the first round.*
- *The third transformation is missing in the last round.*



2 Transformation

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*

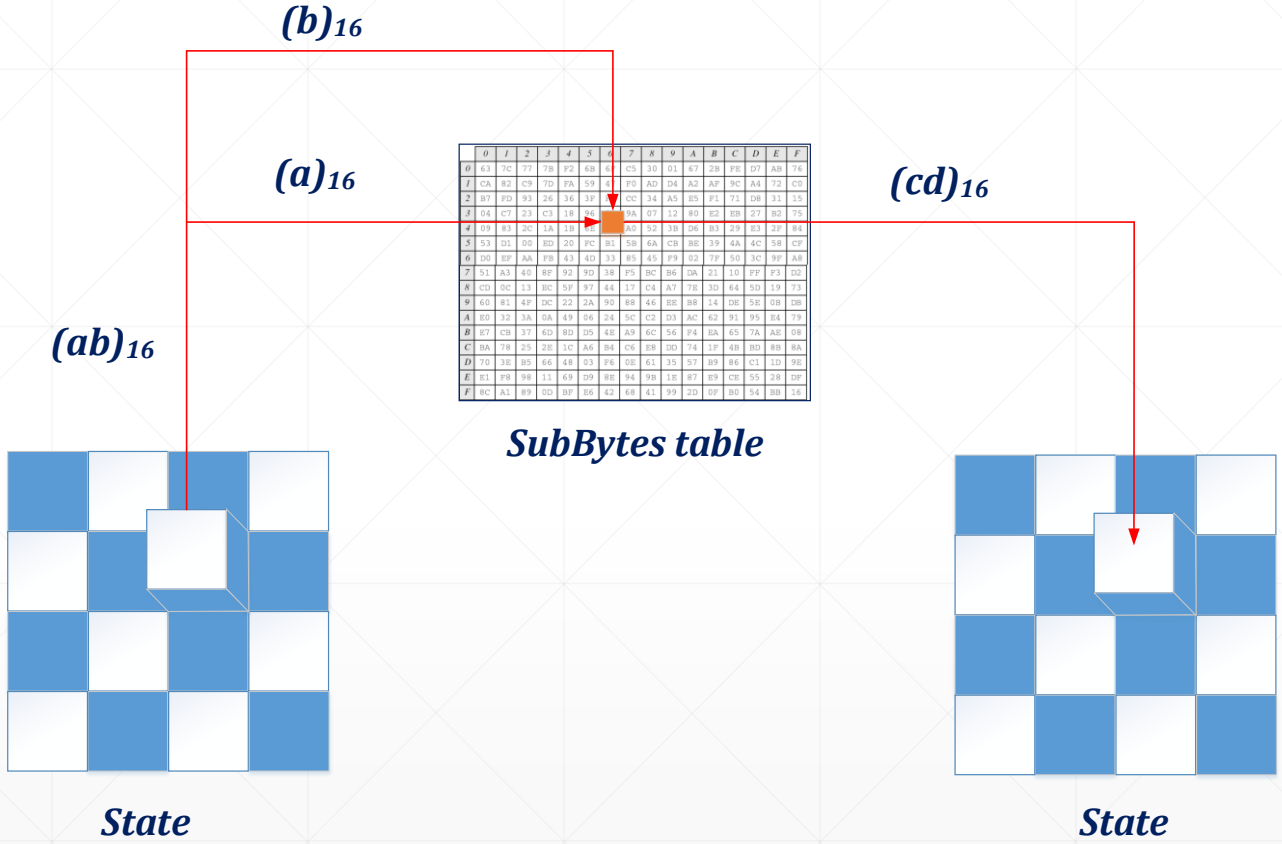
The first transformation, SubBytes, is used at the encryption site. To substitute a byte, we interpret the byte as two hexadecimal digits.

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

2 Transformation

❖ Substitution

- ***SubBytes***



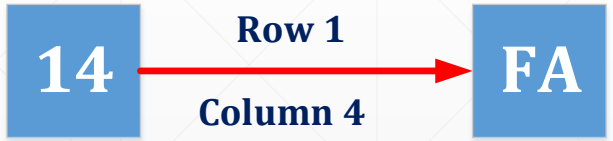
2 Transformation

❖ Substitution

▪ *SubBytes*

The values in the state is substituted by another values according to a lookup table called *SubBytes Table*

Example



SubBytes 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	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

2 Transformation

❖ Substitution

- ***InvSubBytes***

Example

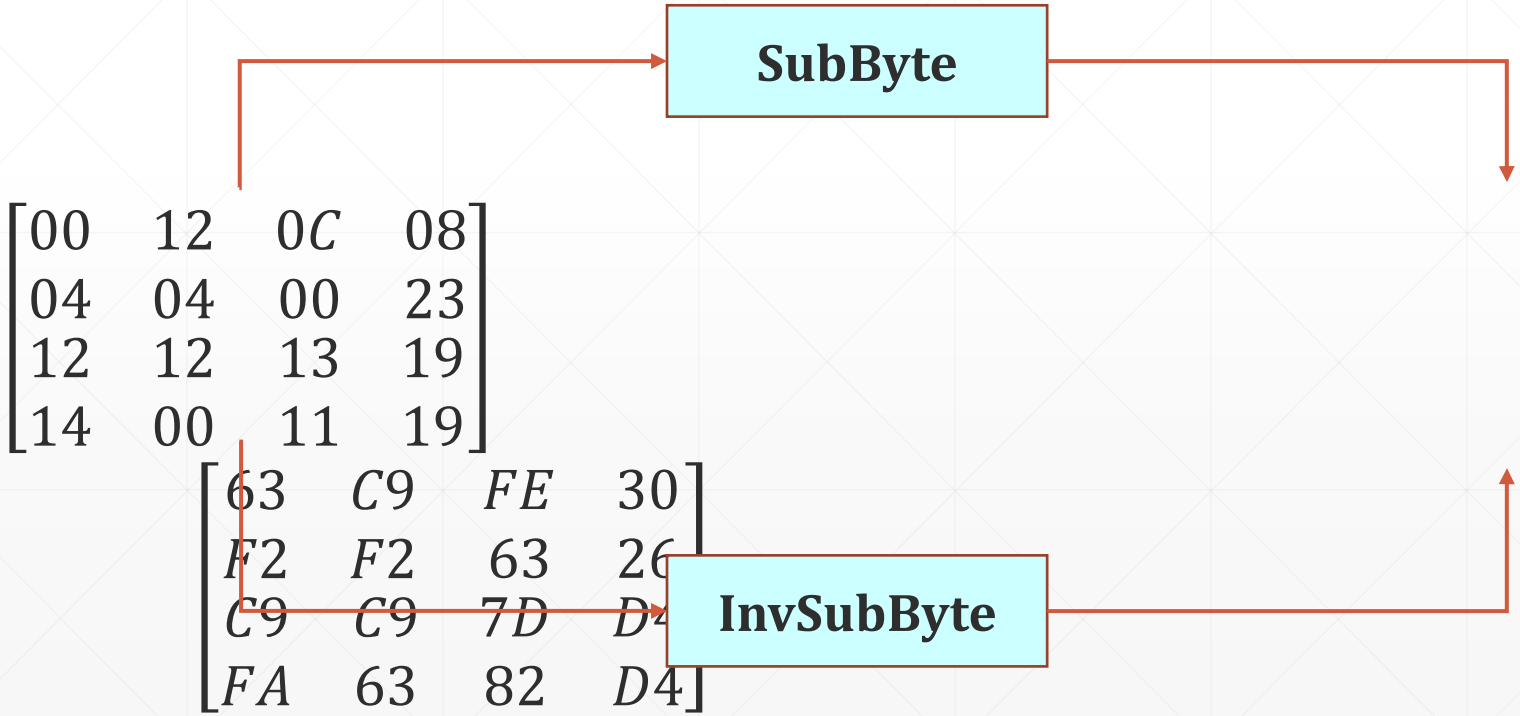
InvSubBytes Table

	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
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

2 Transformation

❖ Substitution

Example: This 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.



2 Transformation

❖ Substitution

- *Transformation Using the GF(2^8) Field*

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

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

$$Where: y = (01100011)_2 = x^6 + x^5 + x^1 + 1$$

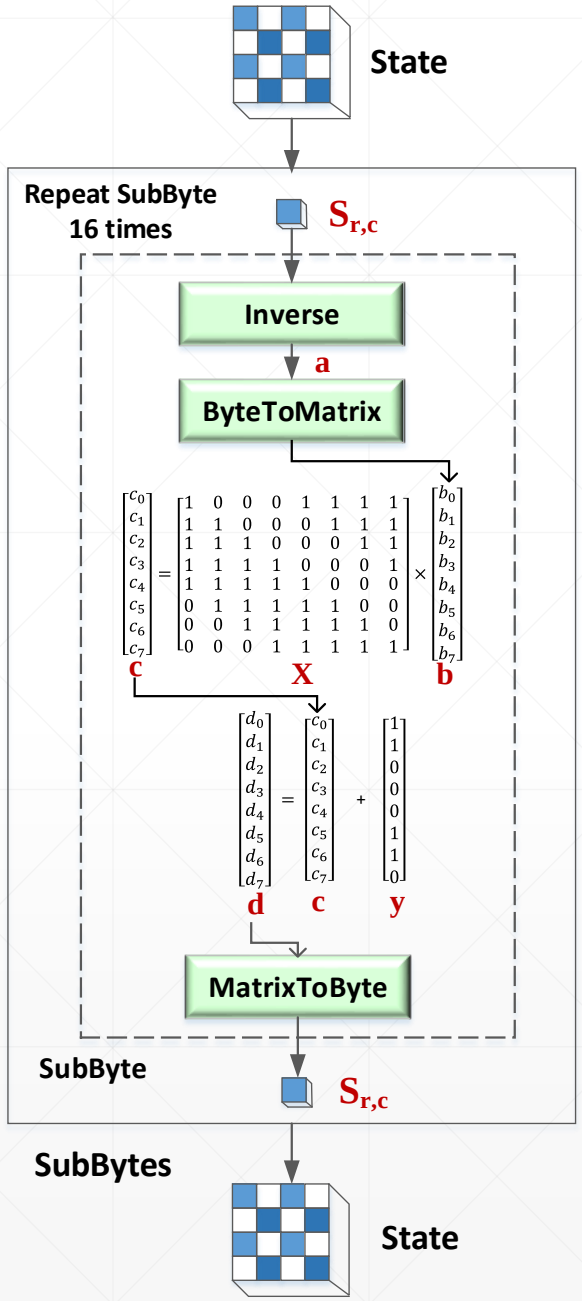
- ***Note***

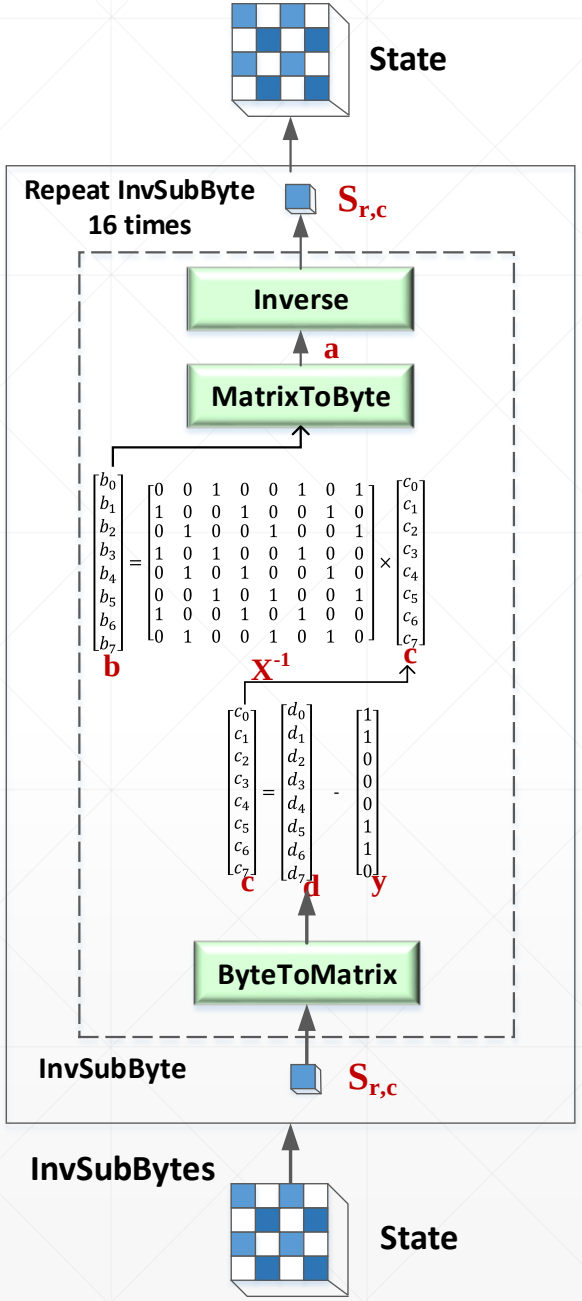
The SubBytes and InvSubBytes transformations are inverses of each other

2 Transformation

❖ Substitution

SubBytes and InvSubBytes processes





2 Transformation

❖ Substitution

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:***
   - *The multiplicative inverse of 0C in GF(2^8) field is B0, which means b is (1011 0000).*
   - *Multiplying matrix X by this matrix results in c = (1001 1101).*
   - *The result of XOR operation is d = (1111 1110), which is FE in hexadecimal.*
2. ***InvSubByte:***
   - *The result of XOR operation is c = (1001 1101).*
   - *The result of multiplying by matrix X^{-1} is (1101 0000) or B0.*
   - *The multiplicative inverse of B0 is 0C.*

2 Transformation

❖ Substitution

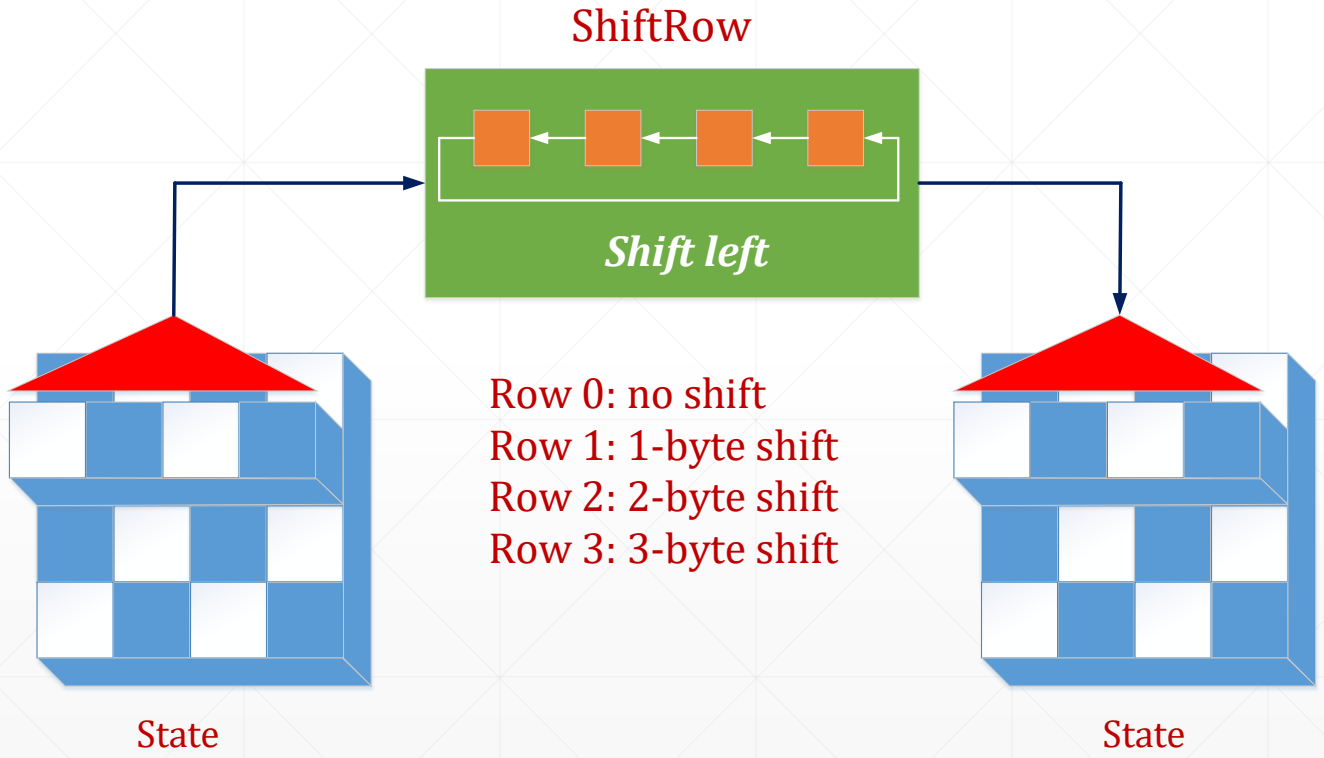
```
SubBytes (S)
{
for (r=0 to 3)
        for (c=0 to 3)
                S_{r,c}= SubBytes(S_{r,c})
}
```

```
SubBytes(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
}
```

2 Transformation

❖ Permutation

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



ShiftRows transformation

2 Transformation

❖ Permutation

▪ *InvShiftRows*

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

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

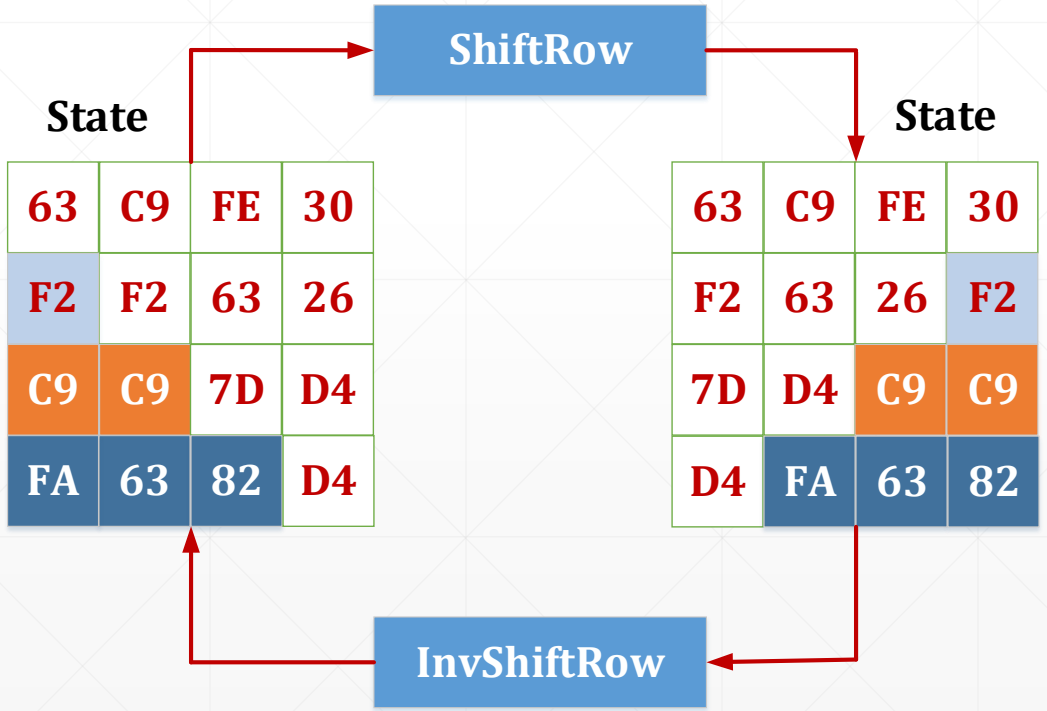
```
ShiftRow(row,n) // n is the number of bytes to be shifted
        {
                CopyRow(row, t)         // t is a temporary row
                for (c = 0 to 3)
                        row(c-n)mod4 ← tc
        }
```

2 Transformation

❖ Permutation

Example: This figure shows how a state is transformed using ShiftRows transformation.

The figure also shows that InvShiftRows transformation creates the original state.



2 Transformation

❖ 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.

$$\begin{matrix} ax + \\ ex + \\ ix + \\ mx + \end{matrix} \quad \begin{matrix} by + \\ fy + \\ jy + \\ ny + \end{matrix} \quad \begin{matrix} cz + \\ gz + \\ kz + \\ oz + \end{matrix} \quad \begin{matrix} dt \\ ht \\ lt \\ pt \end{matrix} \rightarrow \begin{bmatrix} \square \\ \square \\ \square \\ \square \end{bmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ t \end{bmatrix}$$

New matrix **Constant Matrix** Old Matrix

2 Transformation

❖ Mixing

$$\begin{matrix} ax + \\ ex + \\ ix + \\ mx + \end{matrix} \quad \begin{matrix} by + \\ fy + \\ jy + \\ ny + \end{matrix} \quad \begin{matrix} cz + \\ gz + \\ kz + \\ oz + \end{matrix} \quad \begin{matrix} dt \\ ht \\ lt \\ pt \end{matrix} \quad = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ t \end{bmatrix}$$

New matrix **Constant Matrix** Old Matrix

Constant matrices used by MixColumns and InvMixColumns

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

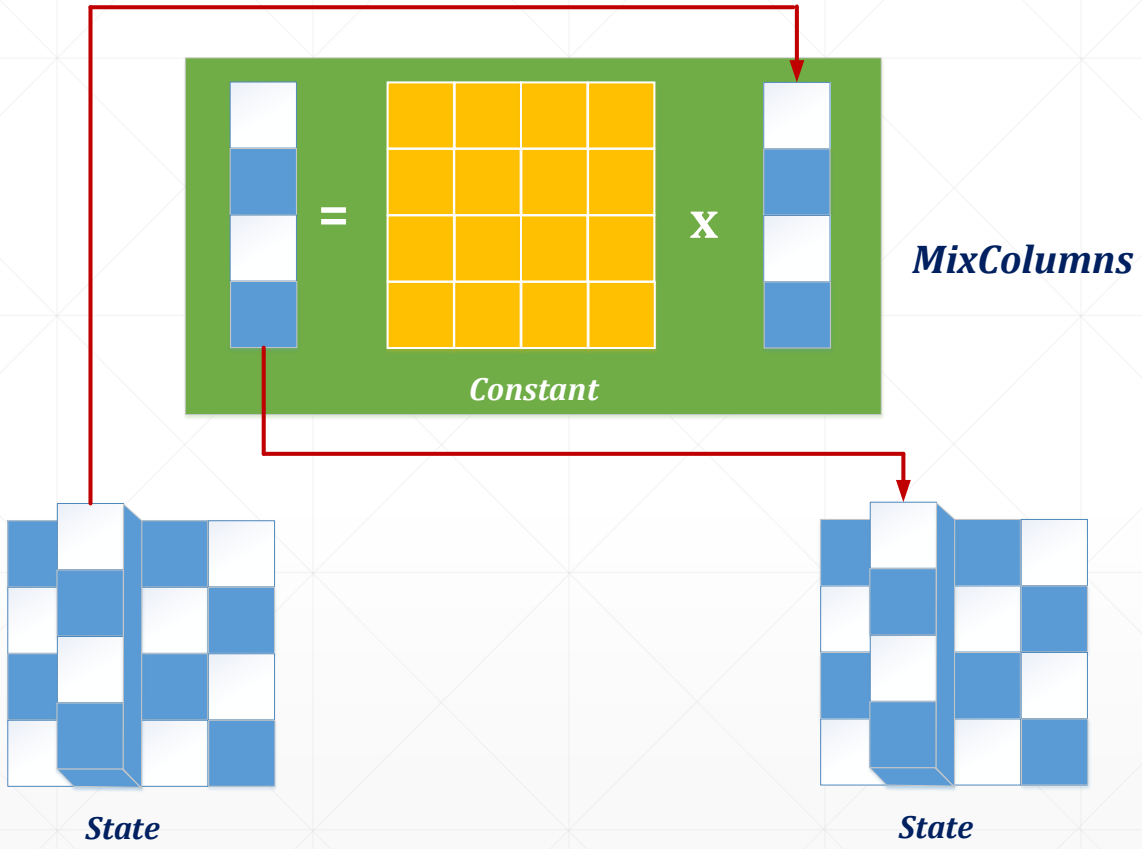
C C^{-1}

2 Transformation

❖ Mixing

▪ *MixColumns*

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



MixColumns transformation

2 Transformation

❖ Mixing

- ***InvMixColumns***

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

NOTE: The MixColumns and InvMixColumns transformations are inverses of each other.

2 Transformation

❖ Mixing

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

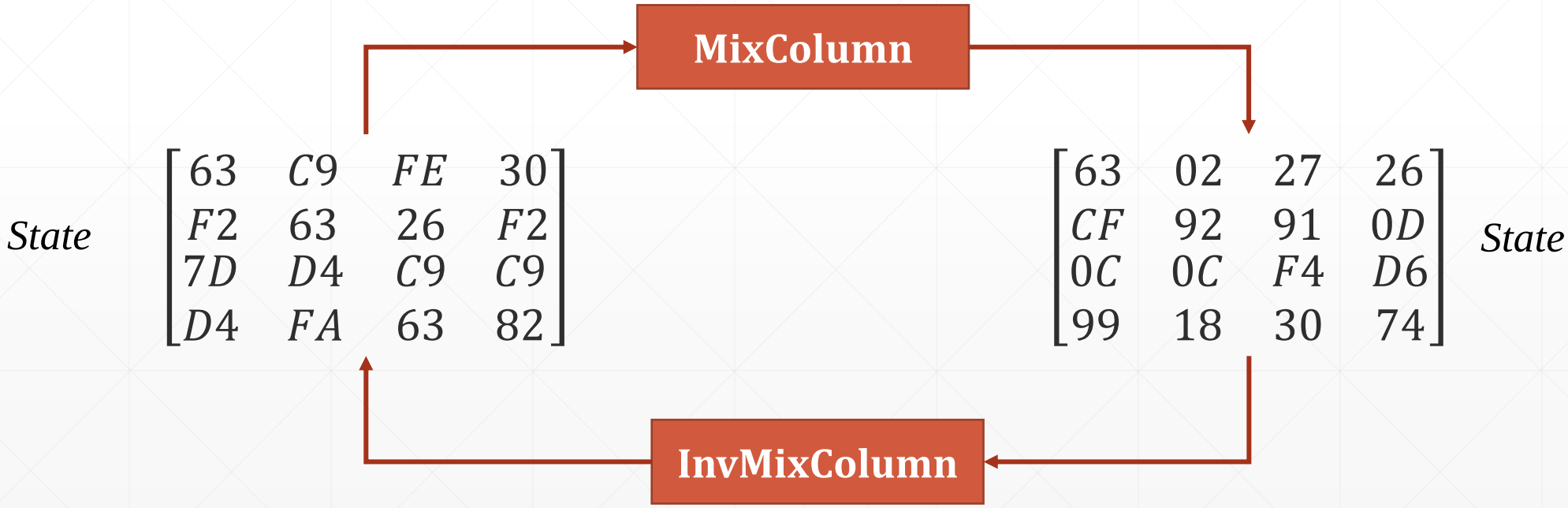
```
mixcolumn(col)
{
        CopyColumn(col, t)                  // t is a temporary column
        col_0 ← (0x02)* t_0 ⊕ (0x03 *t_1) ⊕ t_2 ⊕ t_3
        col_1 ← t_0 ⊕ (0x02)* t_1 ⊕ (0x03 *t_2) ⊕ t_3
        col_2 ← t_0 ⊕ t_1 ⊕ (0x02)* t_2 ⊕ (0x03) *t_3
        col_3 ← (0x03*t_0) ⊕ t_1 ⊕ t_2 ⊕ (0x02)*t_3
}
```

2 Transformation

❖ Mixing

Example: Figure below 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 example



2 Transformation

❖ 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.

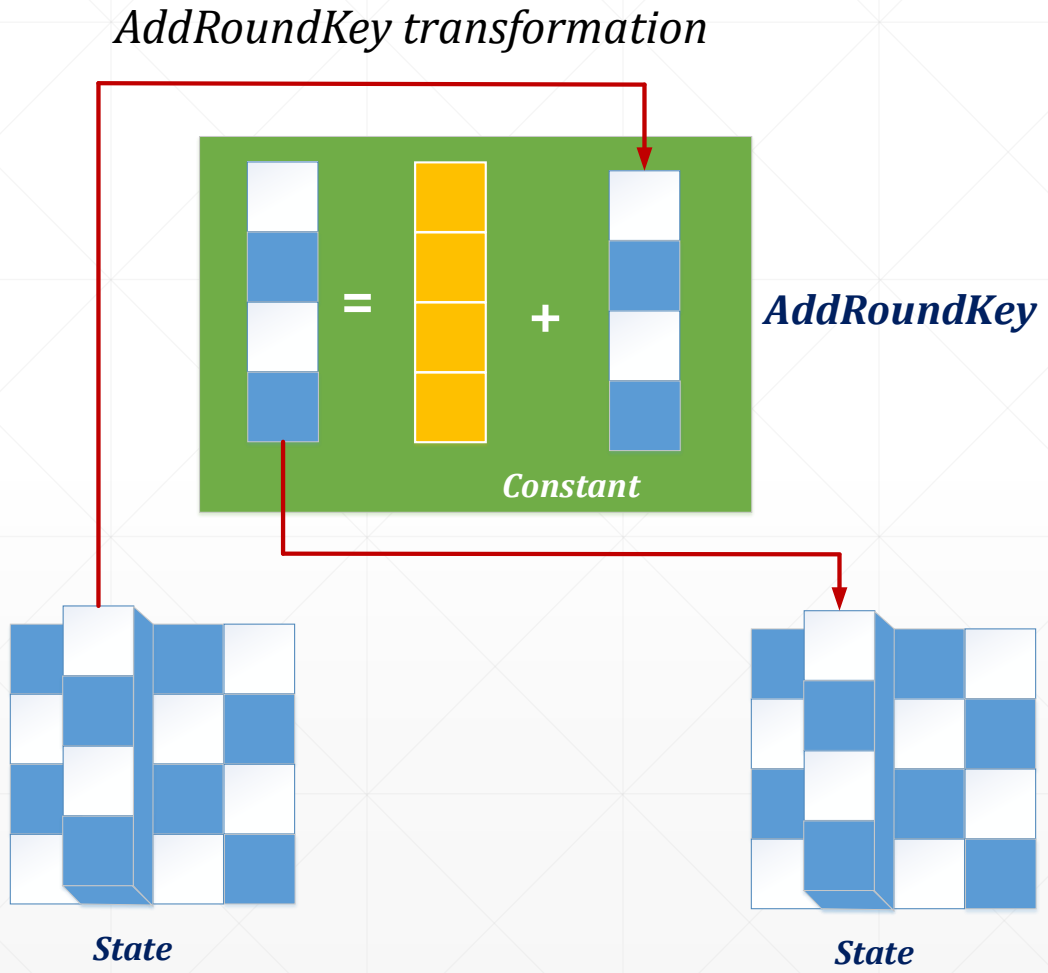
NOTE: The AddRoundKey transformation is the inverse of itself.

2 Transformation

❖ Key Adding

- ***AddRoundKey***

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



3 Key Expansion

- *To create round keys for each round, AES uses a keyexpansion 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.*

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

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

3 Key Expansion

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.

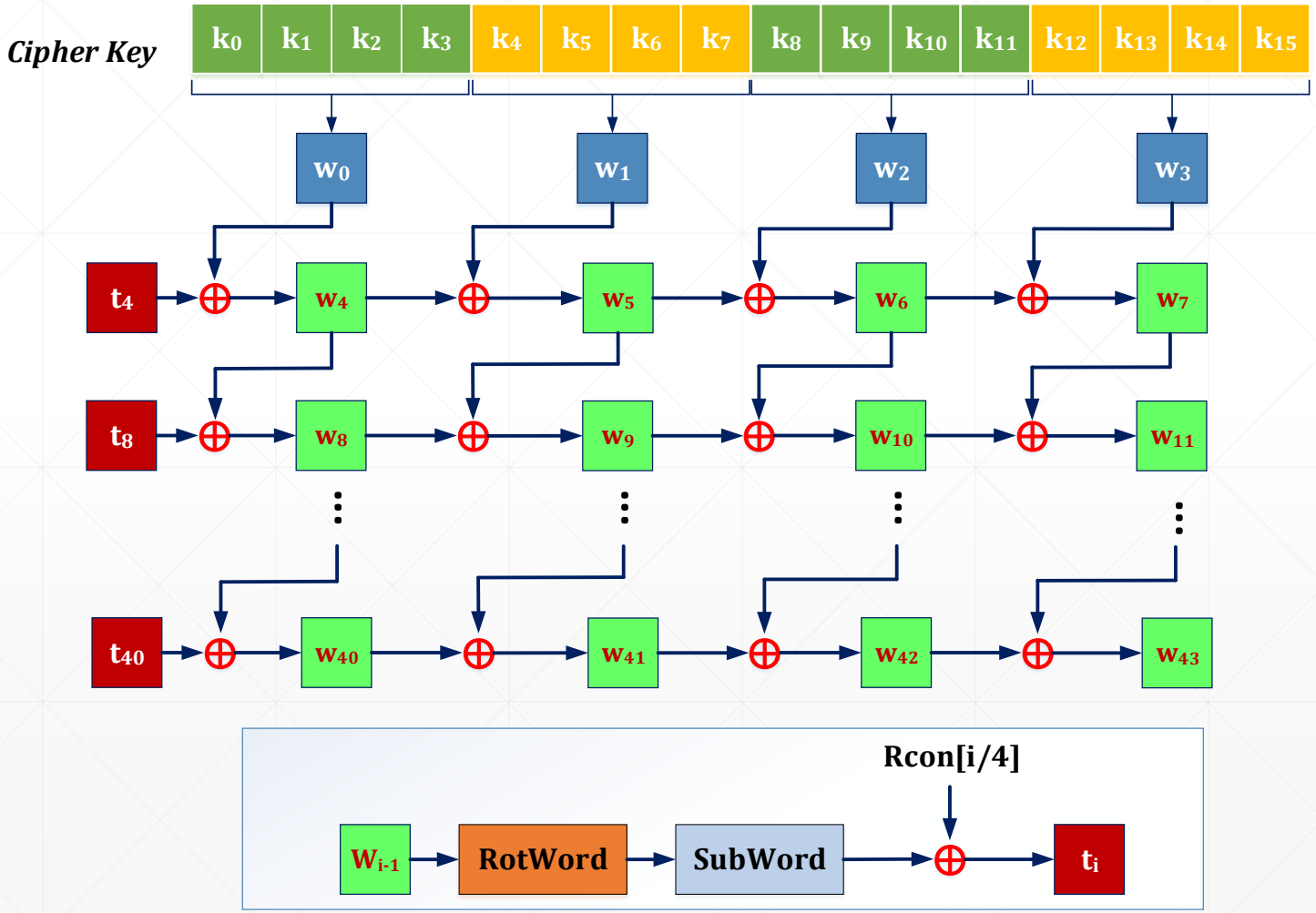
No.	AES Cipher	Round Nr
1	AES-128	10
2	AES-192	12
3	AES-256	14

Words for each round

Round	Words			
Pre-round	W_0	W_1	W_2	W_3
1	W_4	W_5	W_6	W_7
2	W_8	W_9	W_{10}	W_{11}
…	…	…	…	…
Nr	W_{4Nr}	W_{4Nr+1}	W_{4Nr+2}	W_{4Nr+3}

3 Key Expansion

❖ *Key Expansion in AES-128*



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

3 Key Expansion

❖ *Key Expansion in AES-128*

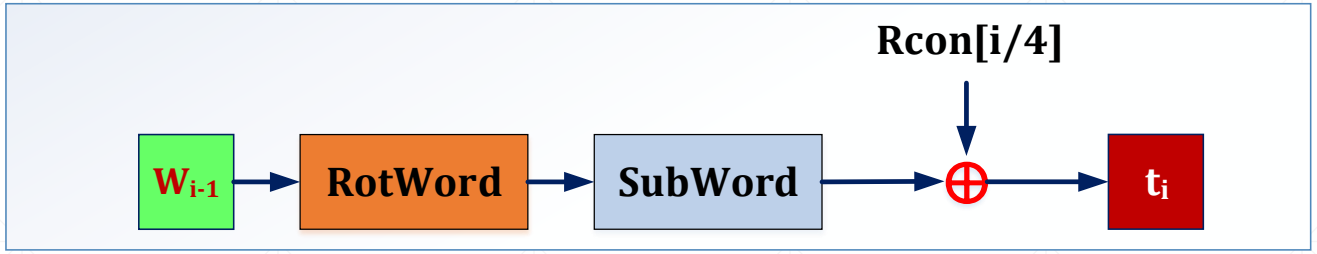
The process is as follows:

1. *The first 4 words (w_0,w_1,w_2,w_3) are made from the cipher key. The cipher key is thought of as an array of 16 bytes (k_0 to k_{15}). The first 4 bytes (k_0 to k_3) become w_0; the next 4 bytes (k_4 to k_7) become w_1; and so on. In other words, the concatenation of the words in this group replicates the cipher key.*
2. *The rest of the words (w_i for i=4 to 43) are made as follows:*
   a. *If (i mod 4) ≠ 0, $w_i = w_{i-1} \oplus w_{i-4}$. This means each word is made from the one at the left and the one at the top.*
   b. *If (I mod 4) = 0, $w_i = t \oplus w_{i-4}$. Here t, a temporary word, is the result of applying 2 routines, SubWord and RotWord, on w_{i-1} and XORing the result with a round constants, Rcon. In other words, we have:*

$$t = SubWord(RotWord(w_{i-1})) \oplus RCON_{i/4}$$

3 Key Expansion

❖ *Key Expansion in AES-128*



RotWord:

The RotWord (rotate word) routine is similar to the ShiftRows transformation, but it is applied to only one row. The routine takes a word as an array of 4 bytes and shifts each byte to the left with wrapping.

SubWord:

The SubWord (Substitute word) rountine is similar to the SubBytes transformation, but it is applied only to 4 bytes. The rountine takes each byte in the word and substitutes another byte for it.

3 Key Expansion

❖ *Key Expansion in AES-128*

RCON Constant (Hexa)

Round 1 $RCON_0$	*Round 2* $RCON_1$	*Round 3* $RCON_2$	*Round 4* $RCON_3$	*Round 5* $RCON_4$	*Round 6* $RCON_5$	*Round 7* $RCON_6$	*Round 8* $RCON_7$	*Round 9* $RCON_8$	*Round 10* $RCON_9$
01	**02**	**04**	**08**	**10**	**20**	**40**	**80**	**1B**	**36**
00	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**
00	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**
00	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**

Each round constant, Rcon, is a 4-byte value in which the rightmost three bytes are always zero.

3 Key Expansion

❖ *Key Expansion in AES-128*

The key-expansion routine can either use the above 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, prime $= x^8 + x^4 + x^3 + x + 1$)

$$RC_1 \rightarrow x^{1-1} = x^0 \quad \text{mod prime} \quad = 1$$
$$\rightarrow 0000\ 0001 \rightarrow (01)_{16}$$
$$RC_2 \rightarrow x^{2-1} = x^1 \quad \text{mod prime} \quad = x$$
$$\rightarrow 0000\ 0010 \rightarrow (02)_{16}$$
$$RC_3 \rightarrow x^{3-1} = x^2 \quad \text{mod prime} \quad = x^2$$
$$\rightarrow 0000\ 0100 \rightarrow (04)_{16}$$
$$RC_4 \rightarrow x^{4-1} = x^3 \quad \text{mod prime} \quad = x^3$$
$$\rightarrow 0000\ 1000 \rightarrow (08)_{16}$$
$$RC_5 \rightarrow x^{5-1} = x^4 \quad \text{mod prime} \quad = x^4$$
$$\rightarrow 0001\ 0000 \rightarrow (10)_{16}$$
$$RC_6 \rightarrow x^{6-1} = x^5 \quad \text{mod prime} \quad = x^5$$
$$\rightarrow 0010\ 0000 \rightarrow (20)_{16}$$
$$RC_7 \rightarrow x^{7-1} = x^6 \quad \text{mod prime} \quad = x^6$$

3 Key Expansion

❖ *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)) xor RConi/4  // t is a temporary word
             wi <- t+ wi-4
                }
         }
 }
```

3 Key Expansion

❖ *Key Expansion in AES-128*

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.

3 Key Expansion

❖ *Key Expansion in AES-128*

This table shows how the keys for each round are calculated assuming that the 128-bit cipher key agreed upon by Alice and Bob is $(24\ 75\ A2\ B3\ 34\ 75\ 56\ 88\ 31\ E2\ 12\ 00\ 13\ AA\ 54\ 87)_{16}$.

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	$w_{40} = DBF92E26$	$w_{41} = D538D2D2$	$w_{42} = F49B88C0$	$w_{43} = 0DDB4F40$

3 Key Expansion

❖ *Key Expansion in AES-128*

The concept of weak keys, as we discussed for DES in Chapter 3, 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.

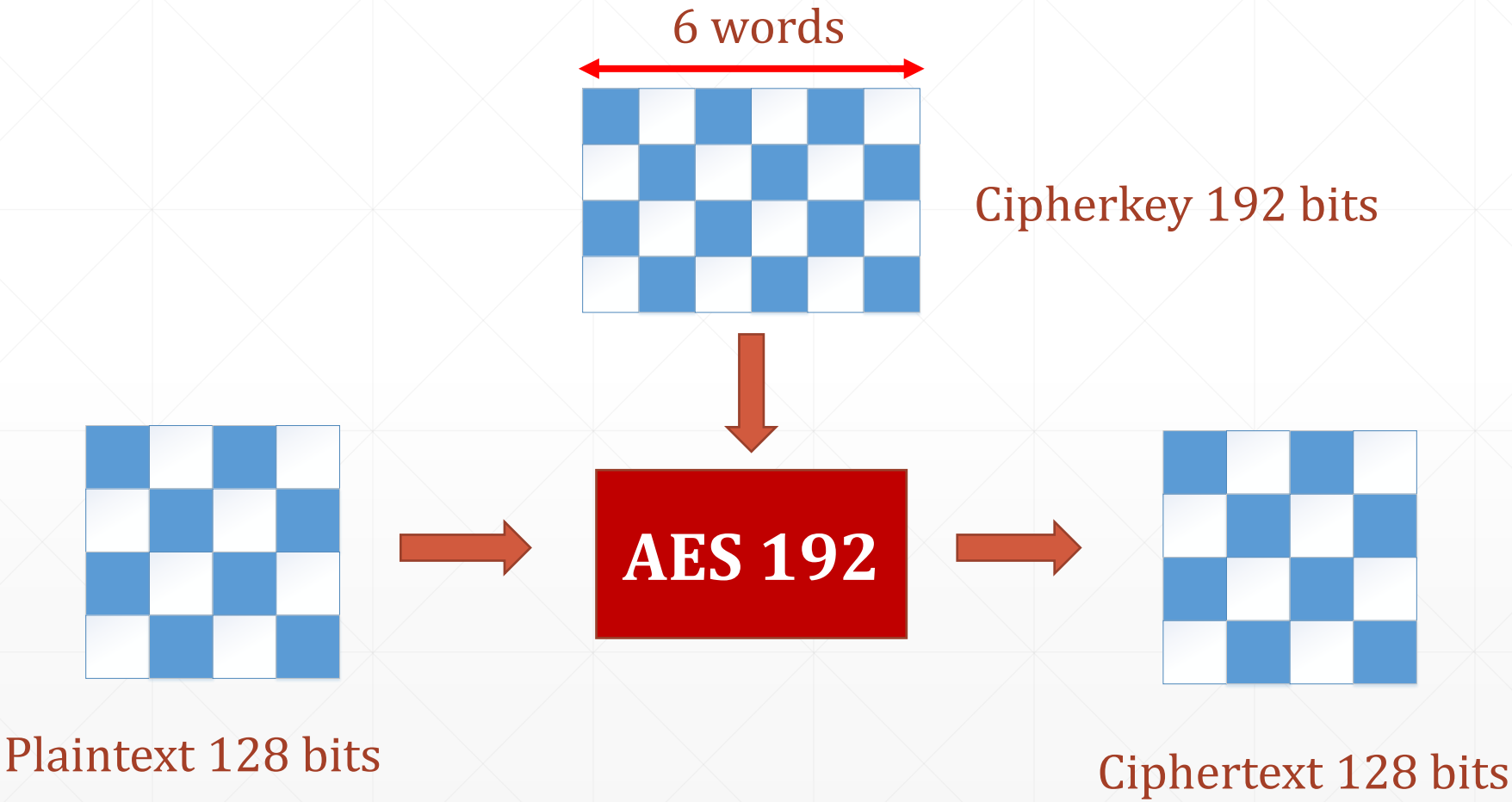
3 Key Expansion

- *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:*

1. ***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, $w_i \leftarrow w_{i-1} + w_{i-6}$; otherwise, $w_i \leftarrow t + w_{i-6}$*
2. ***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, $w_i \leftarrow w_{i-1} + w_{i-8}$; otherwise, $w_i \leftarrow t + w_{i-8}$*
   c. *If i mod 4 = 0, but i mod 8 ≠ 0, then $w_i = SubWord(w_{i-1}) + w_{i-8}$*

3 Key Expansion

❖ ***Key Expansion in AES-192***



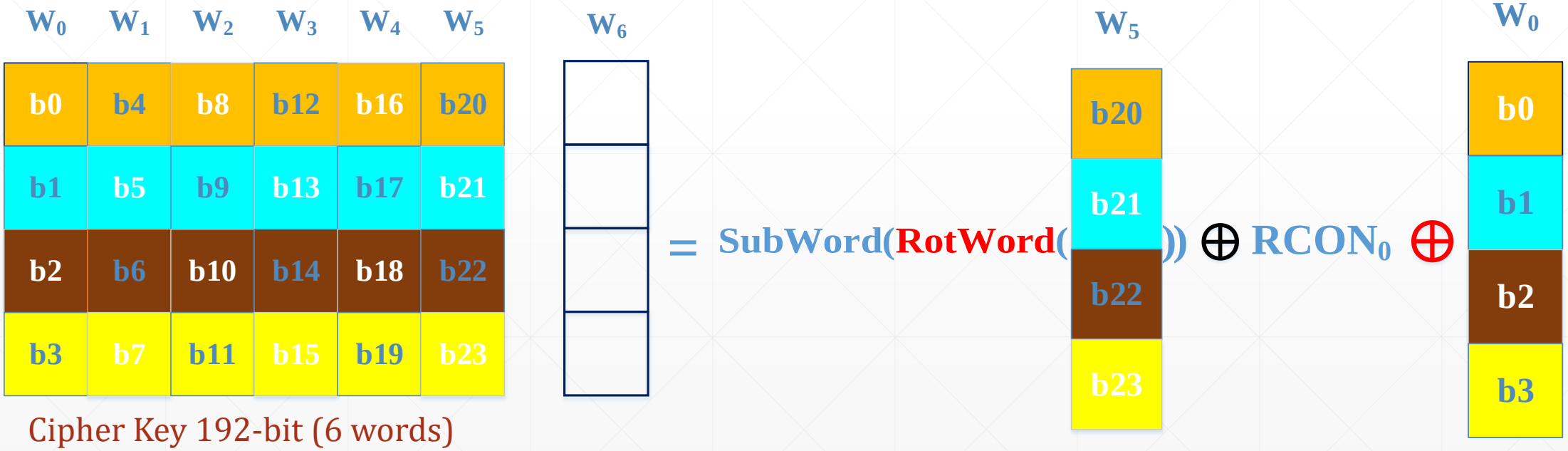
3 Key Expansion

❖ *Key Expansion in AES-192*

Words for each round

Round	Words			
Pre-round	W_0	W_1	W_2	W_3
1	W_4	W_5	W_6	W_7
2	W_8	W_9	W_{10}	W_{11}
...	...	...	...	...
12	W_{48}	W_{49}	W_{50}	W_{51}

➢ *Finding W_6*



Cipher Key 192-bit (6 words)

3 Key Expansion

❖ *Key Expansion in AES-192*

➢ *Finding* W_6

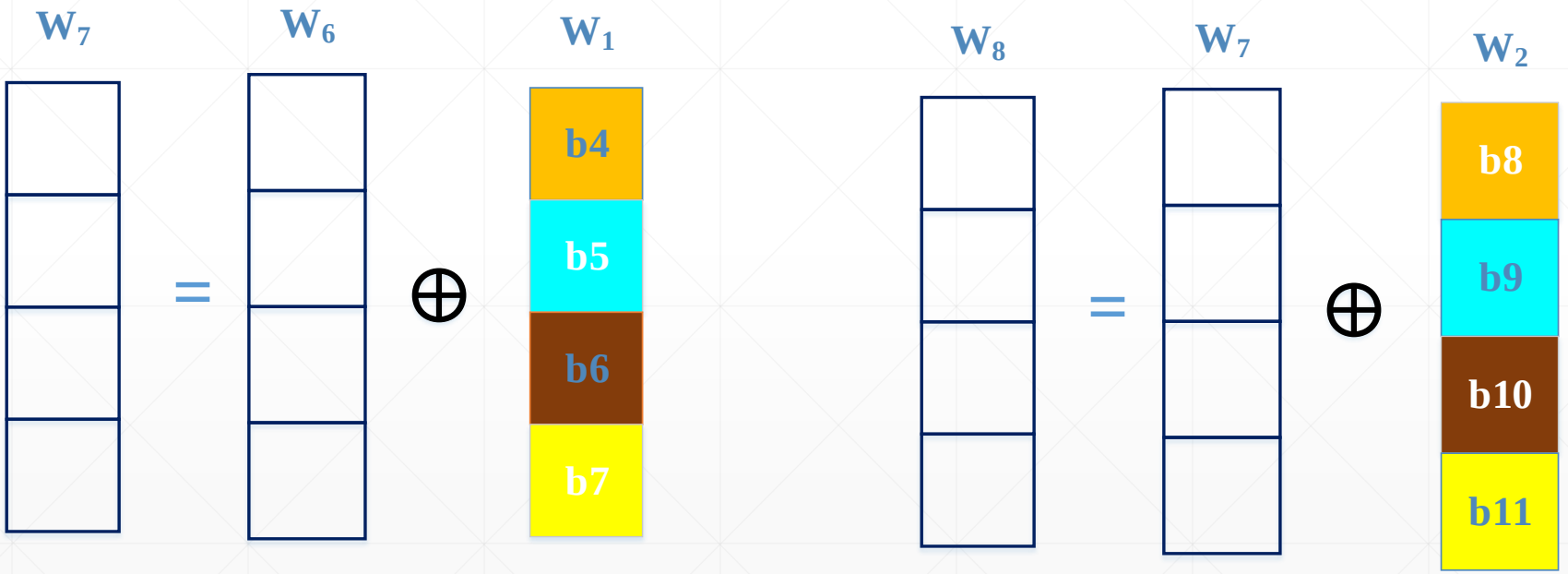
RCON Constant (Hexa)

Round 1 **RCON$_0$**	*Round 2* **RCON$_1$**	*Round 3* **RCON$_2$**	*Round 4* **RCON$_3$**	*Round 5* **RCON$_4$**	*Round 6* **RCON$_5$**	*Round 7* **RCON$_6$**	*Round 8* **RCON$_7$**	*Round 9* **RCON$_8$**	*Round 10* **RCON$_9$**
01	**02**	**04**	**08**	**10**	**20**	**40**	**80**	**1B**	**36**
00	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**
00	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**
00	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**	**00**

3 Key Expansion

❖ *Key Expansion in AES-192*

➢ *Finding W_7,W_8*

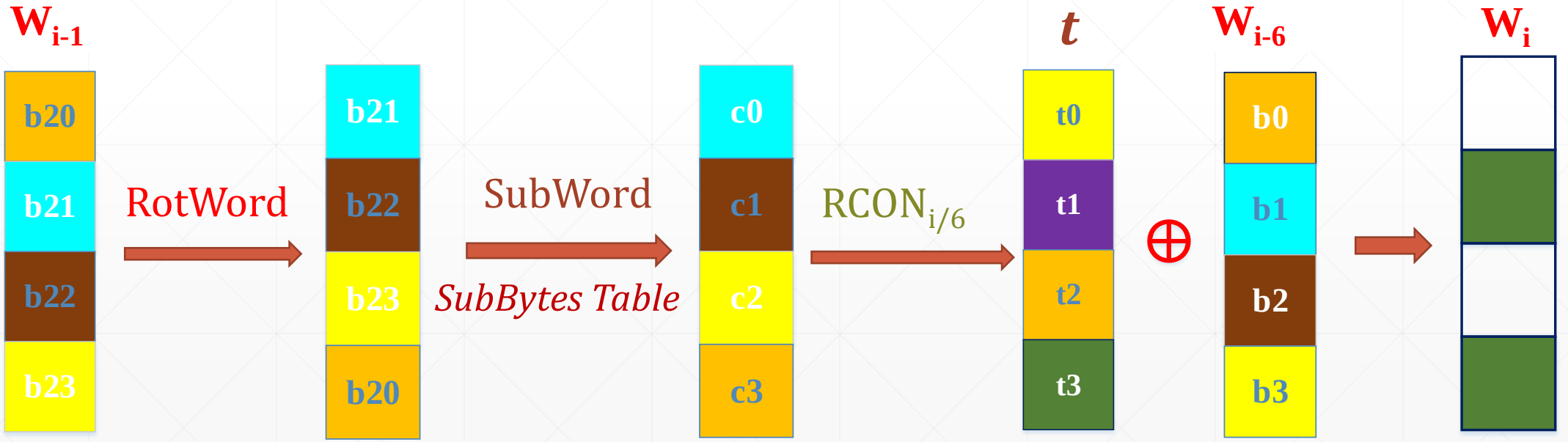


3 Key Expansion

❖ *Key Expansion in AES-192*

➢ *Generality*

- W_i = ? (for i=0:51)
  - If (i mod 6==0)

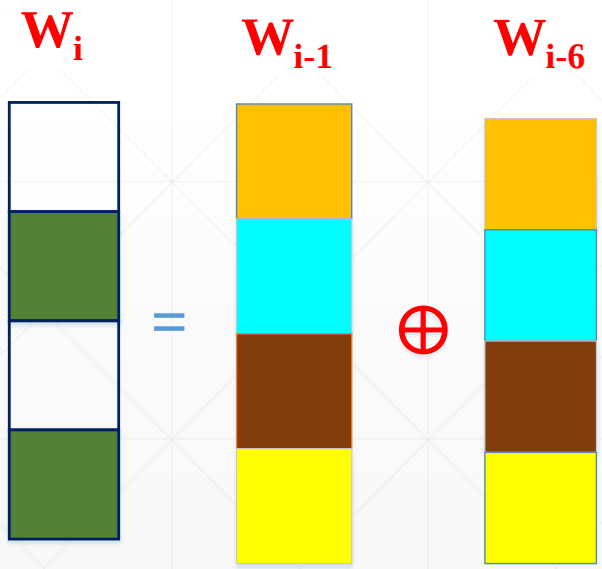


3 Key Expansion

❖ *Key Expansion in AES-192*

➢ *Generality*

- W_i = ? (for i=0:51)
  - else



3 Key Expansion

❖ *Key Expansion in AES-192*

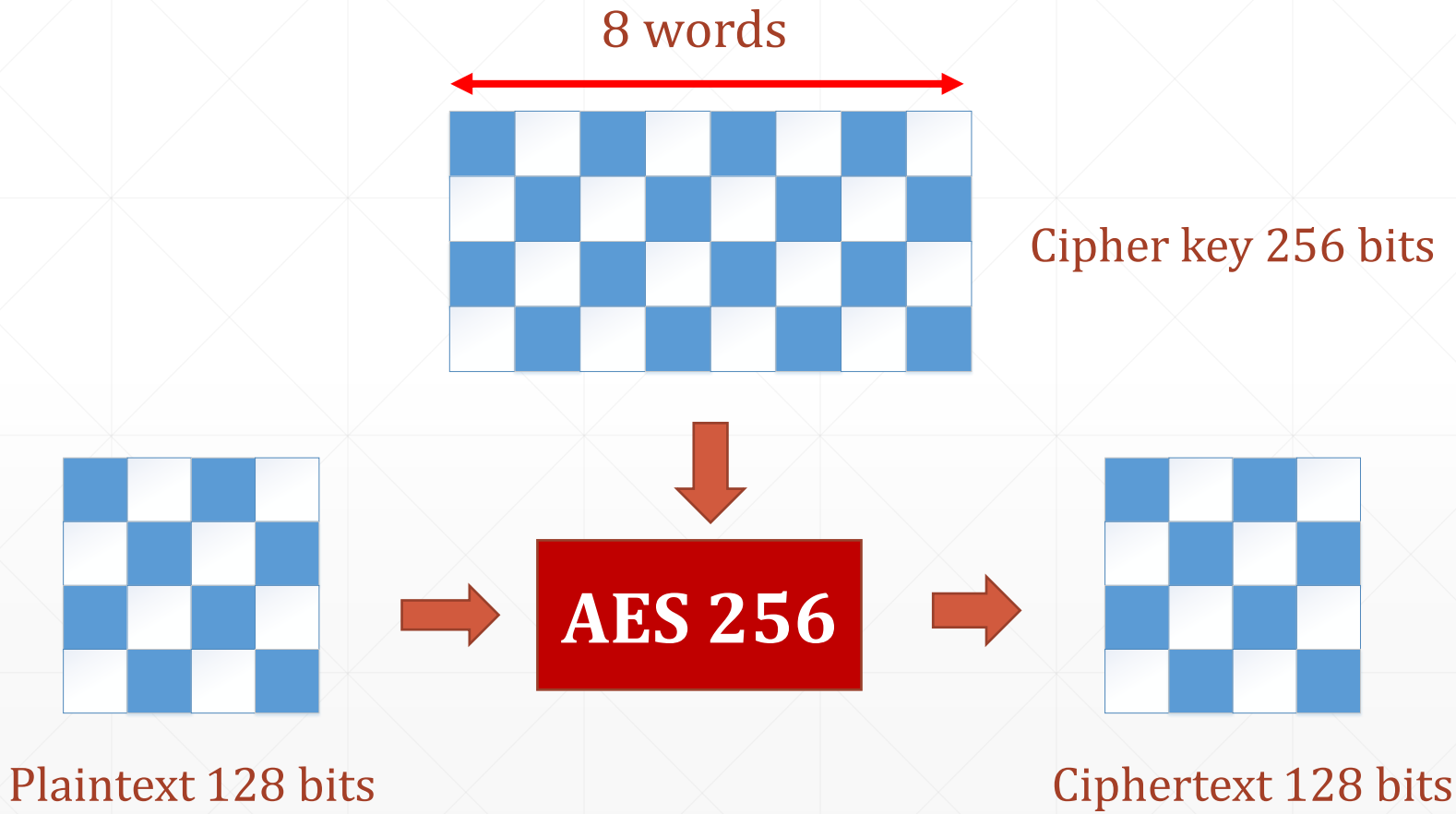
➢ *Generality*

Words for each round

Round	Words			
Pre-round	W_0	W_1	W_2	W_3
1	W_4	W_5	W_6	W_7
2	W_8	W_9	W_{10}	W_{11}
…	…	…	…	…
12	W_{48}	W_{49}	W_{50}	W_{51}

3 Key Expansion

❖ ***Key Expansion in AES-256***



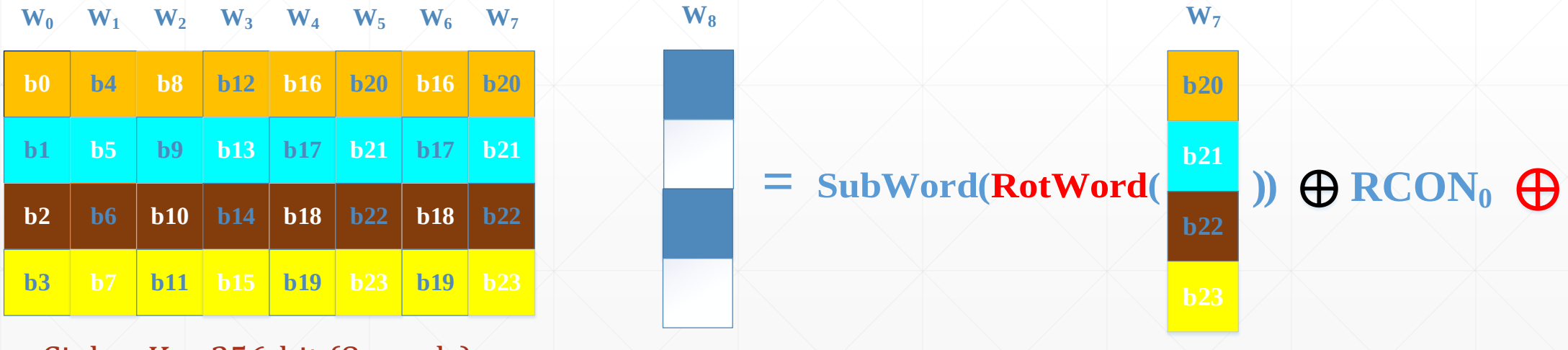
3 Key Expansion

❖ *Key Expansion in AES-256*

➢ *Finding W_8*

Words for each round

Round	Words			
Pre-round	W_0	W_1	W_2	W_3
1	W_4	W_5	W_6	W_7
2	W_8	W_9	W_{10}	W_{11}
...	...	...	...	...
14	W_{56}	W_{57}	W_{58}	W_{59}

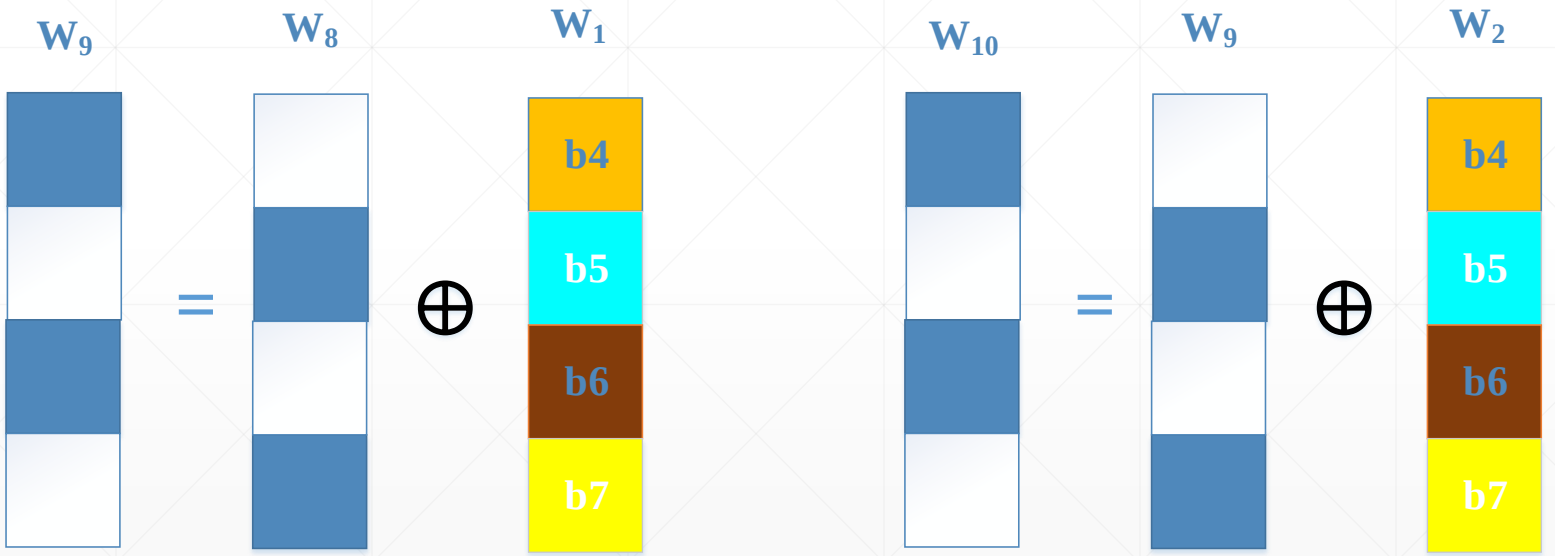


Cipher Key 256-bit (8 words)

3 Key Expansion

❖ *Key Expansion in AES-256*

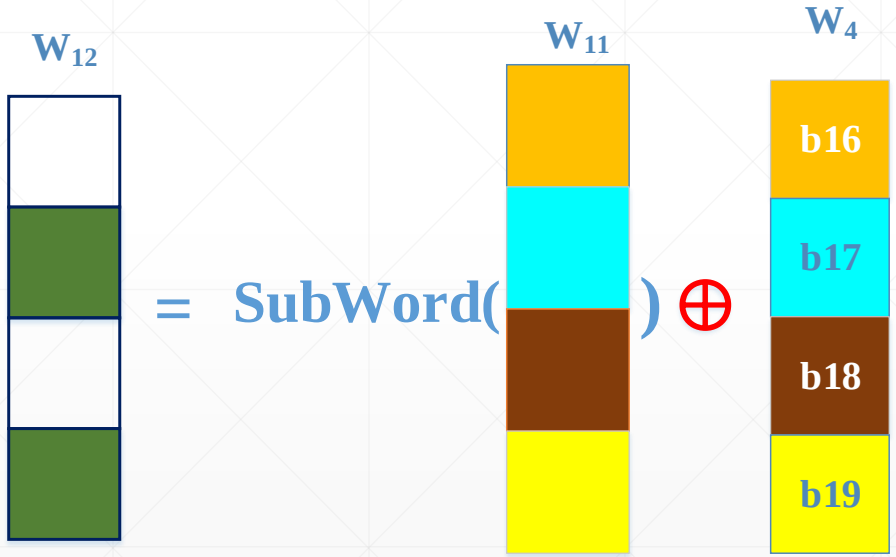
➢ *Finding W_9, W_{10}*



3 Key Expansion

❖ *Key Expansion in AES-256*

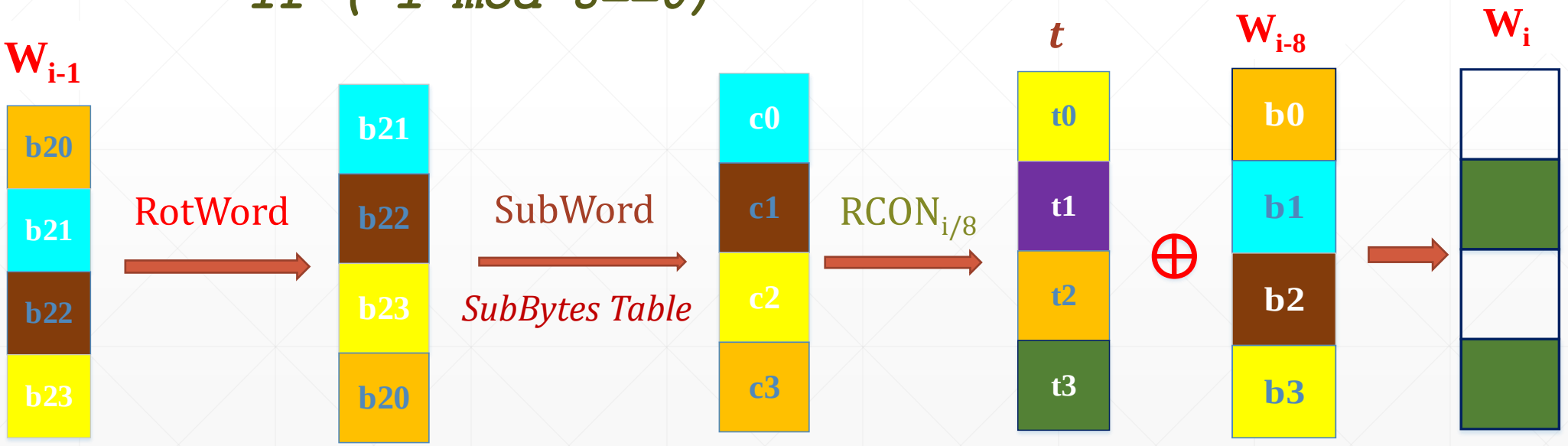
➢ *Finding W_{12}*



3 Key Expansion

❖ *Key Expansion in AES-256*

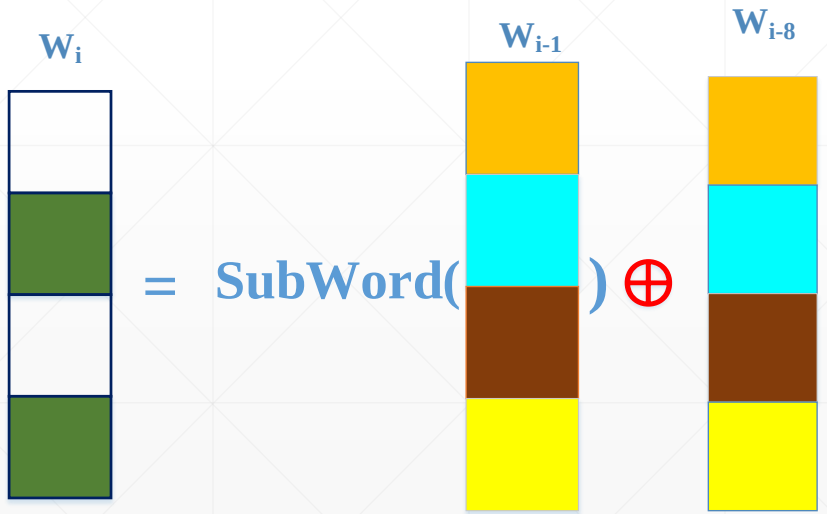
- *Generality*
  - W_i = ? (for i=0:59)
    - If (i mod 8==0)



3 Key Expansion

❖ *Key Expansion in AES-256*

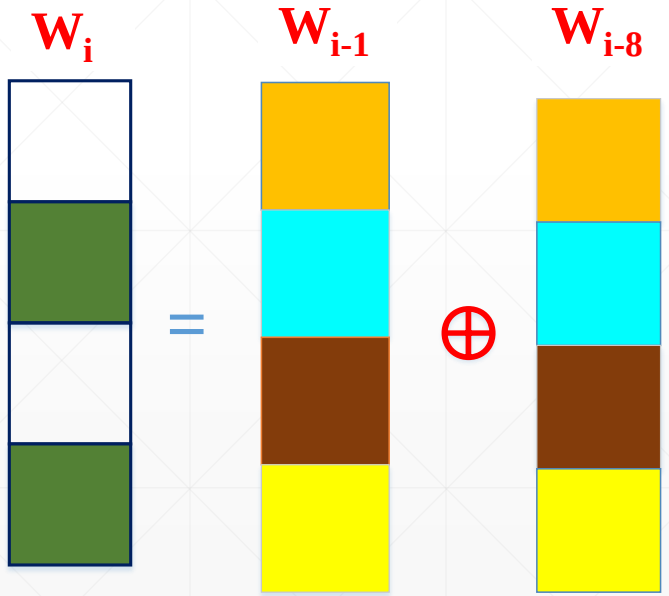
- ➢ *Generality*
  - W_i = ? (for i=0:59)
    - elseif (i mod 4==0)&&(i mod 8~=0)



3 Key Expansion

❖ *Key Expansion in AES-256*

- ➢ *Generality*
  - W_i = ? (for i=0:59)
    - else



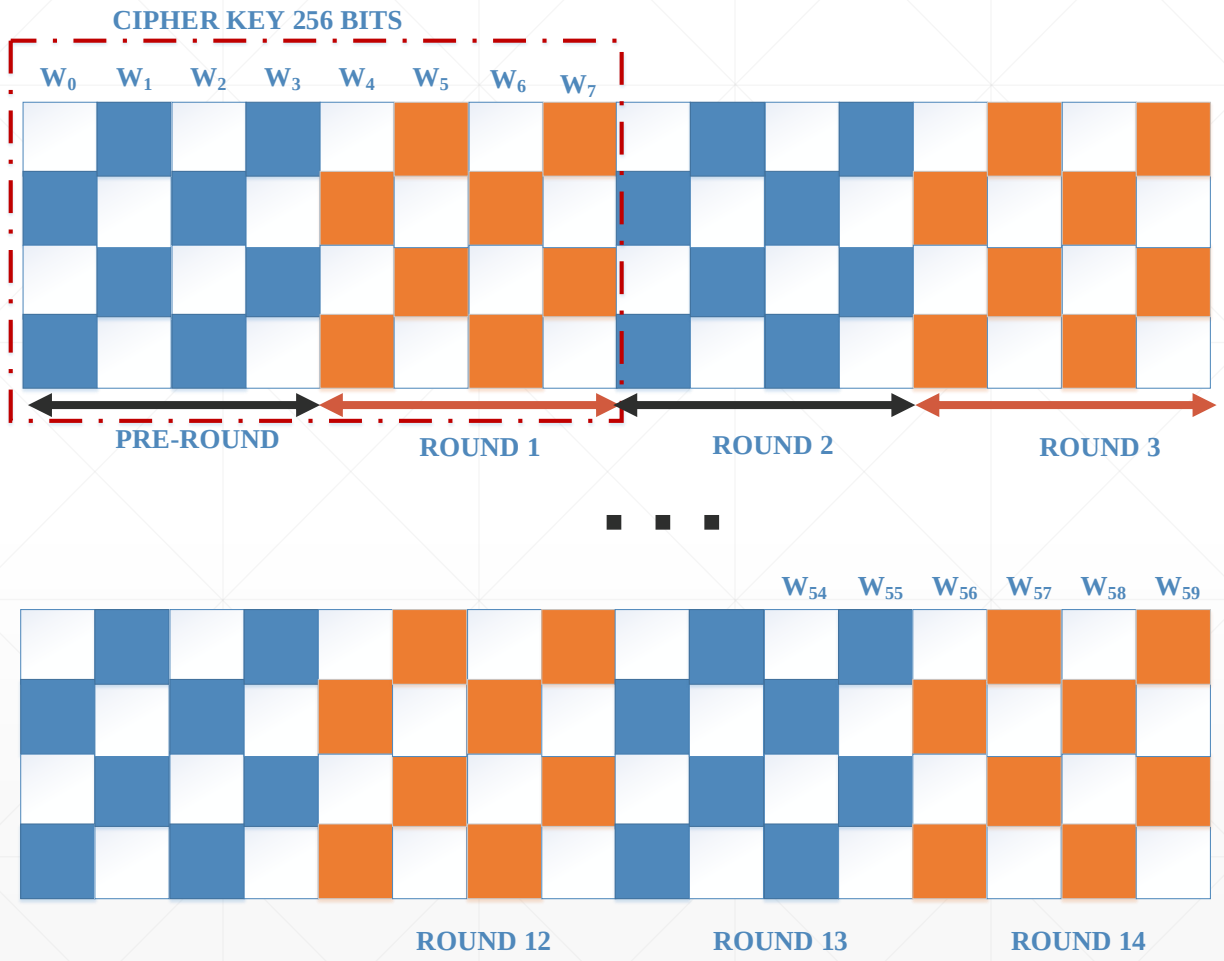
3 Key Expansion

❖ *Key Expansion in AES-256*

➢ *Generality*

Words for each round

Round	Words			
Pre-round	W_0	W_1	W_2	W_3
1	W_4	W_5	W_6	W_7
2	W_8	W_9	W_{10}	W_{11}
…	…	…	…	…
14	W_{56}	W_{57}	W_{58}	W_{59}



4 Ciphers

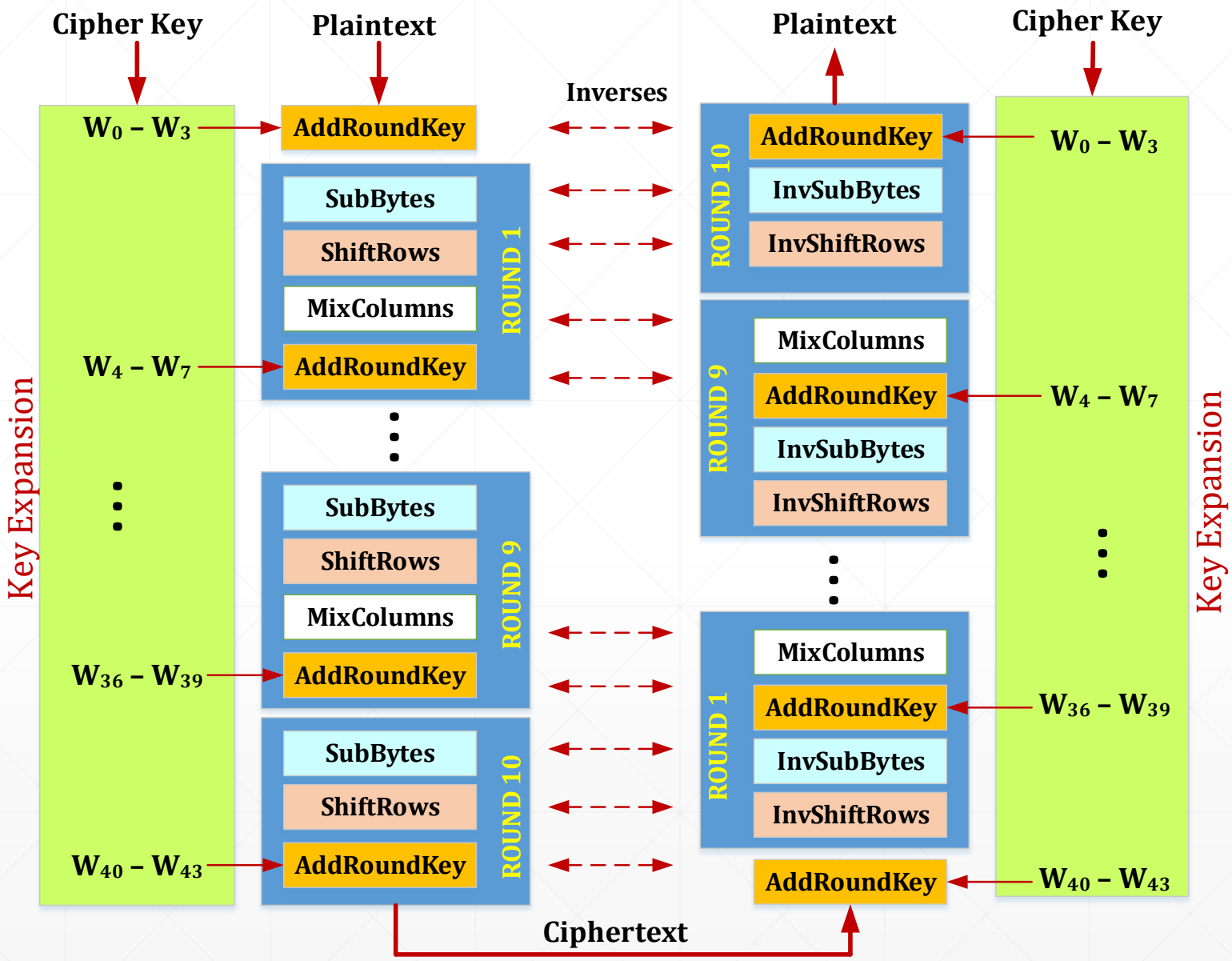
- AES uses four types of transformations for encryption and decryption.
- In the standard, the encryption algorithm is referred to as the cipher and the decryption algorithm as the inverse cipher.

- ***Topics discussed in this section:***
  - Original Design
  - Alternative Design

4 Ciphers

❖ *Original Design*

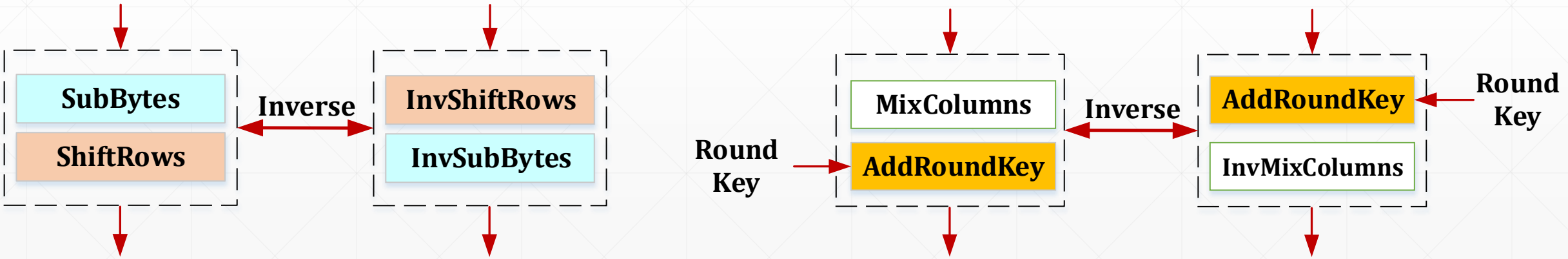
Ciphers and inverse ciphers of the original design



Ciphers

❖ *Alternative Design*

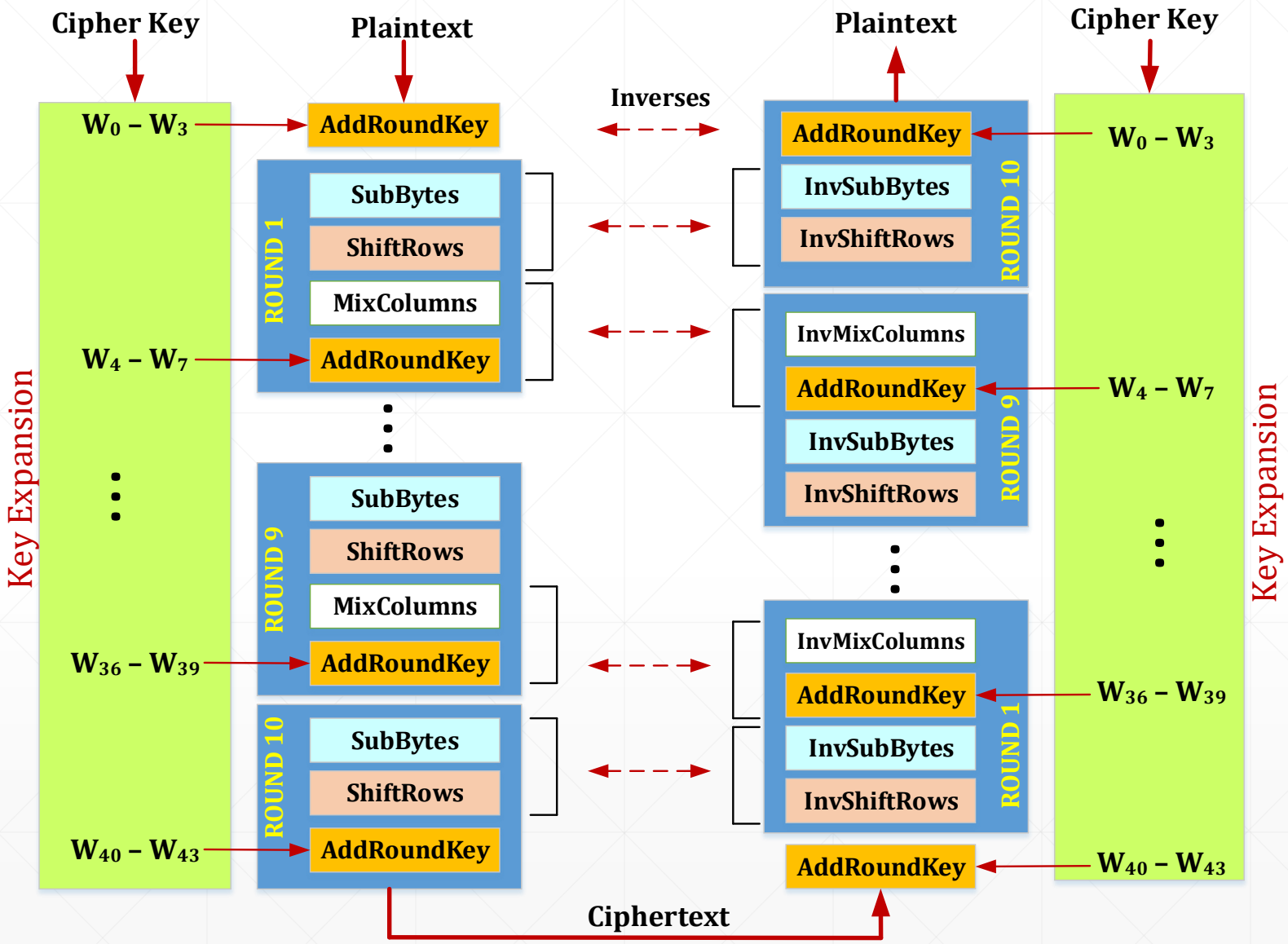
- In this version, the transformation in the reverse cipher are rearranged to make the order of transformations the same in the cipher and reverse cipher.
- In this design, invertibility is provided for a pair of transformations, not for each single transformation.



Ciphers

❖ *Alternative Design*

Cipher and reverse cipher in alternate design



5 Examples

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

Example 5.1: 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

5 Examples

Example 5.1: (Continue)

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

5 Examples

Example 5.1: (Continue)

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

5 Examples

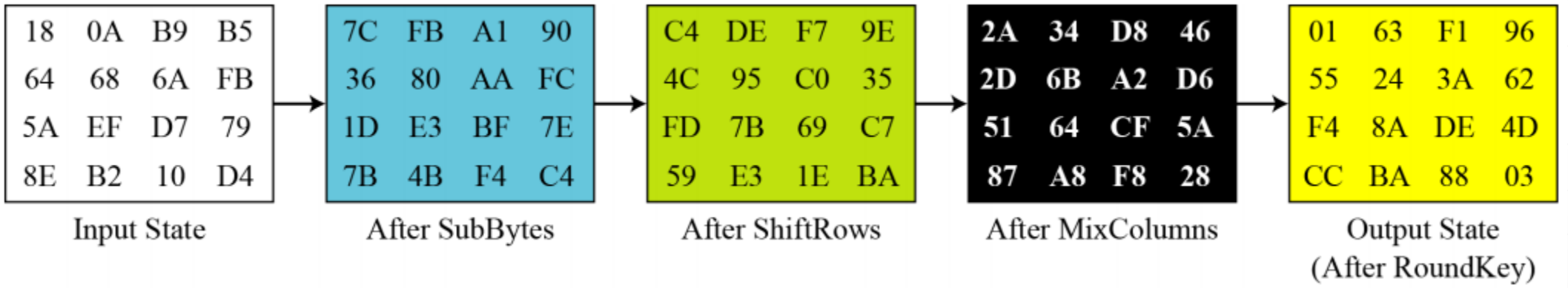
Example 5.1: (Continue)

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

5 Examples

Example 5.2: This Figure shows the state entries in one round, round 7, in Example 5.1

States in a single round



5 Examples

Example 5.3: One may be curious to see the result of encryption when the plaintext is made of all 0s. Using the cipher key in Example 5.1 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

The avalanche effect

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

5 Examples

Example 5.4: 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

6 Analysis of AES

AES was designed after DES. Most ofthe 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.

HUST

hust.edu.vn fb.com/dhbkhn

THANK YOU !