

# 第五章 对称密码体制

5.1 分组密码概述

5.2 数据加密标准DES

5.3 高级加密标准AES

# 本节主要内容


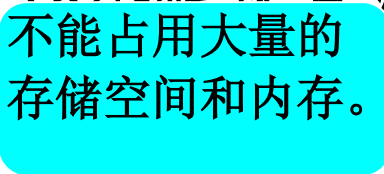
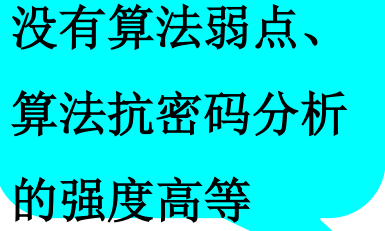
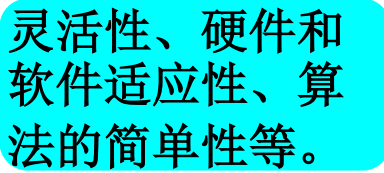

- ◆ 算法描述
- ◆ 基本运算
- ◆ 基本加密变换
- ◆ AES的解密
- ◆ 密钥扩展

## § 5-3 高级加密标准 (AES)

### 背景

从各方面来看，DES已走到了它生命的尽头。其56比特密钥实在太小，虽然三重DES可以解决密钥长度的问题，但是DES的设计主要针对硬件实现，而今在许多领域，需要用软件方法来实现它，在这种情况下，它的效率相对较低。鉴于此，1997年4月15日美国国家标准和技术研究所（NIST）发起征集AES（AES—Advanced Encryption Standard）算法的活动，并成立了AES工作组。目的是为了确定一个非保密的、公开披露的、全球免费使用的加密算法，用于保护下一世纪政府的敏感信息。也希望能够成为保密和非保密部门公用的数据加密标准（DES）。

# AES候选算法的产生

- 1997年9月12日：美国国家标准与技术研究所NIST提出征集该算法的公告；基本要求是：比三重DES快、至少与三重DES一样安全、数据分组长度为128比特、密钥长度为128/192/256比特。
- 1998年8月20日：NIST召开了第一次候选大会，并公布了15个候选算法。
- 1999年3月：NIST从15个候选算法中公布了5个进入第二轮。这5个算法是：Serpent, Twofish, M2000, RC6, 和 XTEA。
- 2000年10月2日：以安全性、性能、大小、实现特性为标准而最终选出了AES算法。
- 2001年：正式发布AES标准。

# 一、算法描述

- Rijndael算法是由两位比利时的密码专家Joan Daemen 和Vincent Rijmen发明的，它很快而且所需的内存不多，这个算法非常可靠；
- Rijndael算法是针对差分分析和线性分析提出来的，是一个分组迭代密码，具有可变的分组长度和密钥长度；
- 典型的SPN结构（不属于Feistel结构）。
- 加密、解密相似但不对称。
- 有较好的数学理论作为基础；结构简单、速度快。
- Rijndael汇聚了安全性能、效率、可实现性和灵活性等优点。

# AES分组长度、密钥长度、轮数的关系

$N_r$	$N_b=128$ 位
$N_k=128$ 位	10
$N_k=192$ 位	12
$N_k=256$ 位	14

$N_b$ : 分组长度;  $N_k$ : 密钥长度;  $N_r$ : 轮数

# NIST对Rijndael算法的评估准则及结论

- **一般安全性：** 没有已知的攻击方法能攻破Rijndael。
- **软件执行：** 利于在8位、16位及DSP等各种平台上执行，并且Rijndael的密钥安装速度很快。
- **受限空间环境：** 对RAM和ROM的要求很低。
- **硬件执行：** 执行速度快；
- **对执行的攻击：** 有利于抵抗**能量攻击**和**计时攻击**。
- **加密与解密：** 加解密函数不同，但执行加密与解密的速度差不多。
- **密钥灵活性：** 在加密前用特定密钥产生所有的子密钥。
- **其他的功能和灵活性：** 支持分组和密钥长度为128、192和256位。
- **指令级并行执行的潜力：** 有很好的并行执行能力。

## 二、基本运算

- ◆ 字节代替SubBytes：用一个S盒完成分组中的按字节的代替。
- ◆ 行移位ShiftRows：简单的置换。
- ◆ 列混淆MixColumns：在域上的算术特性的代替。
- ◆ 轮密钥加AddRoundKey：利用当前分组和扩展密钥的一部分进行按位异或。



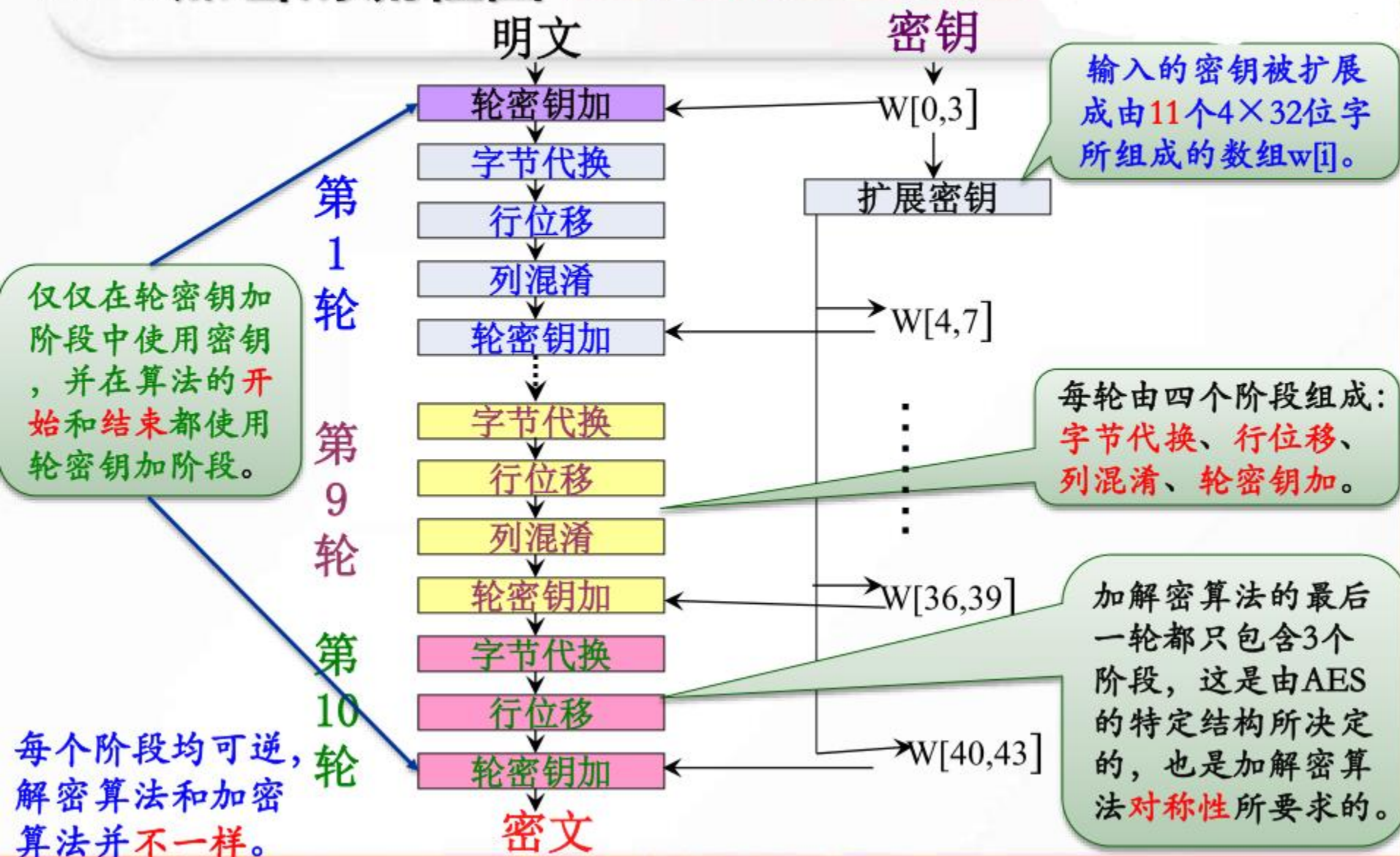
# 5个数据度量单位

- 位：0或1；
- 字节：8位二进制数；
- 字：4个字节，32位；
- 分组：AES的分组是128位；
- 态：16个字节，表示成4\*4的矩阵。态的每一行或每一列都是一个字。

$N_k=128$ 位

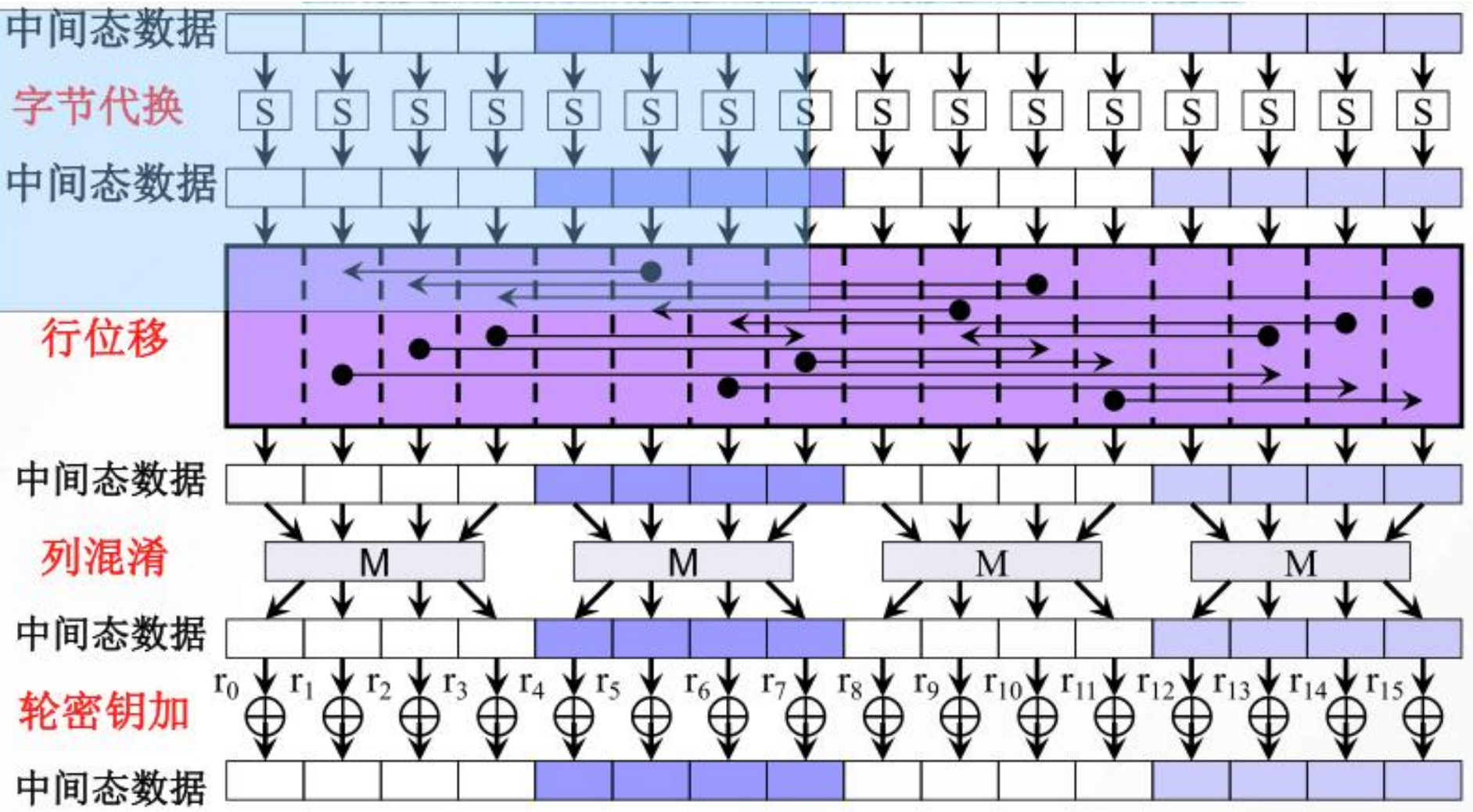
# AES加密的流程图

AES的结构的一个显著特征是不同于DES的结构。



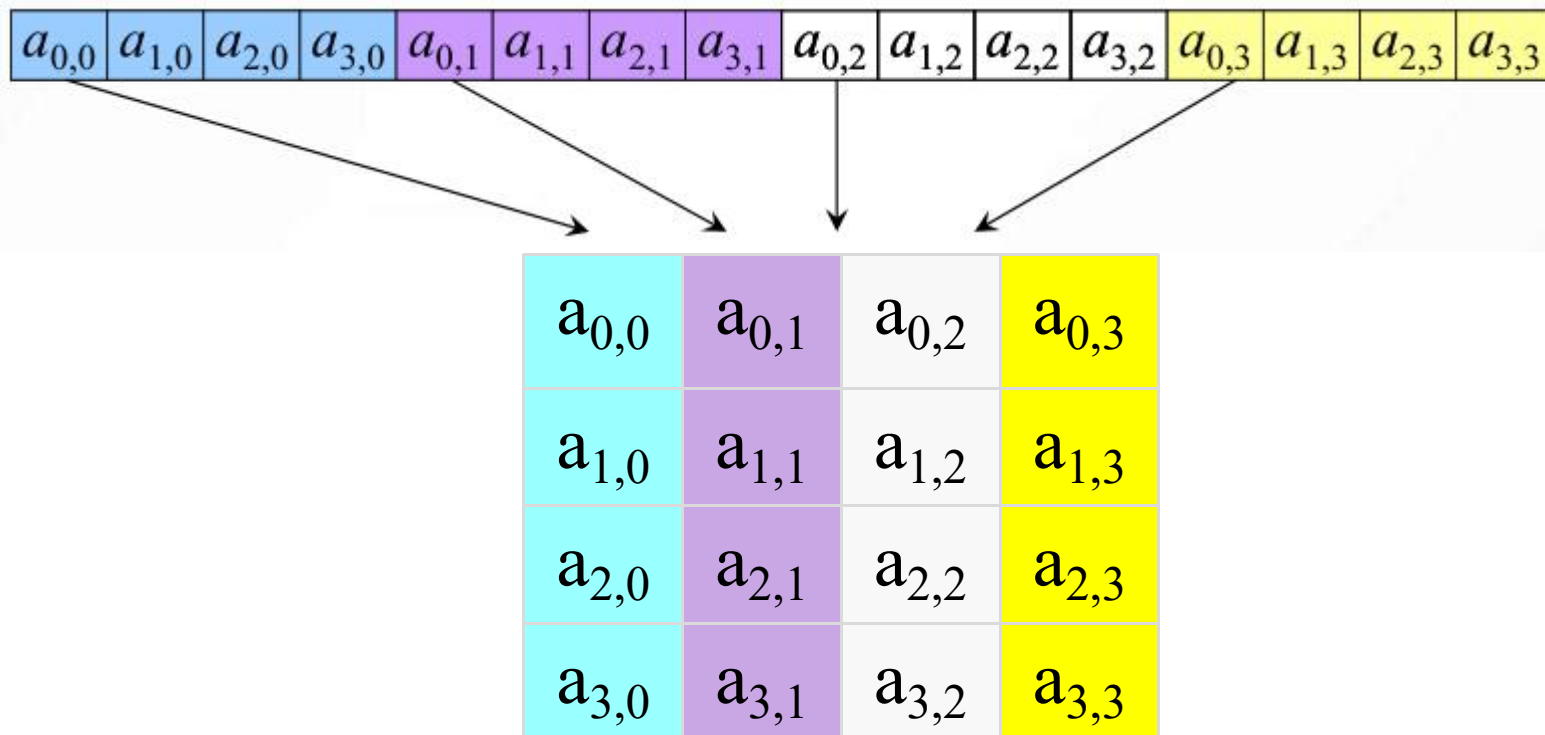
# AES轮函数加密流程图

注：每个方格代表一个字节，四个方格为一组。



# AES的分组

AES首先将明文按字节分成列组。前4个字节组成第一列，接下来的4个字节组成第二列，以此类推，如下图所示。如果块为128位，那么就可组成一个 $4 \times 4$ 的矩阵。





# 1.有限域GF (2<sup>8</sup>) 上的字节运算

➤ 加法 “+”：字节的按位异或运算

$$(a_7a_6a_5a_4a_3a_2a_1a_0) + (b_7b_6b_5b_4b_3b_2b_1b_0) = (c_7c_6c_5c_4c_3c_2c_1c_0)$$

其中， $c_i = a_i \oplus b_i$  (i=0, 1,...,7) 。

## ➤ 乘法 “.”

$$c(x) = a(x) \cdot b(x) = a(x) \times b(x) \bmod m(x)$$

$$a(x) = a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$$

$$b(x) = b_7x^7 + b_6x^6 + b_5x^5 + b_4x^4 + b_3x^3 + b_2x^2 + b_1x + b_0$$

$$c(x) = c_7x^7 + c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x + c_0$$

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

- 是普通多项式乘法，但系数运算可看作比特的乘法和异或运算，即看作域 $\{0,1\}$ 上的运算。

在有限域GF(2<sup>8</sup>)上的多项式运算遵循以下规则：

- 遵循普通多项式运算规则；
- 各项的系数运算以2为模，即遵循GF(2)上的运算规则；
- 如果乘法运算的结果是次数大于7的多项式，则必须将其除以既约多项式 $m(x)=x^8+x^4+x^3+x+1$

## 例子—加法

➤  $(01110011) + (10010101)$

$$\begin{aligned} & (x^6 + x^5 + x^4 + x + 1) + (x^7 + x^4 + x^2 + 1) \\ &= x^7 + x^6 + x^5 + 2x^4 + x^2 + x + 2 \\ &= x^7 + x^6 + x^5 + x^2 + x \end{aligned}$$

所以  $(01110011) + (10010101) = (11100110)$



## 例子—乘法

$$(01110011) \cdot (10010101)$$

$$(x^6 + x^5 + x^4 + x + 1) \cdot (x^7 + x^4 + x^2 + 1)$$

$$= x^{13} + x^{12} + x^{11} + x^{10} + x^9 + 3x^8 + 2x^7 + 2x^6 + 2x^5 + 2x^4 + x^3 + x^2 + x + 1$$

$$= x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^3 + x^2 + x + 1$$

$$= (x^5 + x^4 + x^3 + x^2 + 1)(x^8 + x^4 + x^3 + x + 1) + (x^6 + x^5 + x^4)$$

$$\text{所以, } (01110011) \cdot (10010101) = (01110000)$$

在GF(2<sup>n</sup>)上，对于n次多项式p(x)有：

$$x^n \bmod p(x) = p(x) - x^n \quad or$$

$$x^n \bmod p(x) = p(x) + x^n$$

$$(p(x) - x^n) \bmod p(x)$$

$$= p(x) \bmod p(x) - x^n \bmod p(x)$$

$$= 0 - x^n \bmod p(x)$$

$$= -x^n \bmod p(x)$$

$$= x^n \bmod p(x)$$

# (1) 乘法的实际计算方法

$$\begin{aligned}x^8 \bmod m(x) &= x^8 \bmod (x^8 + x^4 + x^3 + x + 1) \\&= m(x) - x^8 = x^4 + x^3 + x + 1\end{aligned}$$

$$\begin{aligned}x \cdot a(x) &= x \cdot (a_7x^7 + a_6x^6 + a_5x^5 + a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0) \\&= (a_7x^8 + a_6x^7 + a_5x^6 + a_4x^5 + a_3x^4 + a_2x^3 + a_1x^2 + a_0x) \bmod m(x)\end{aligned}$$

# 讨论

$$(a_7x^8 + a_6x^7 + a_5x^6 + a_4x^5 + a_3x^4 + a_2x^3 + a_1x^2 + a_0x) \bmod m(x)$$

(1)  $a_7 = 0$

$$x \cdot a(x) = a_6x^7 + a_5x^6 + a_4x^5 + a_3x^4 + a_2x^3 + a_1x^2 + a_0x$$

$$x \cdot a(x) = (a_6 a_5 a_4 a_3 a_2 a_1 a_0 0)$$

(2)  $a_7 = 1$

$$a_7x^8 \bmod m(x) = x^8 \bmod m(x) = m(x) - x^8 = x^4 + x^3 + x + 1$$

$$x \cdot a(x) = (x^4 + x^3 + x + 1) + (a_6x^7 + a_5x^6 + a_4x^5 + a_3x^4 + a_2x^3 + a_1x^2 + a_0x)$$

$$x \cdot a(x) = (00011011) \oplus (a_6 a_5 a_4 a_3 a_2 a_1 a_0 0)$$

多项式乘以x的结果:

$$(00000010) \bullet (a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0)$$

$$= \begin{cases} (a_6 a_5 a_4 a_3 a_2 a_1 a_0 0) & a_7 = 0 \\ (a_6 a_5 a_4 a_3 a_2 a_1 a_0 0) \oplus (00011011) & a_7 = 1 \end{cases}$$

# 再看前面的例子

$$(01110011) \cdot (10010101) = (01110000),$$

$$\text{即}(73)_H \cdot (95)_H = (70)_H$$

等价于：  $(01110011) \cdot (10010101) =$   
 $[(00000000\mathbf{1}) + (000000\mathbf{1}0) + (000\mathbf{1}0000) + (00\mathbf{1}00000) + (0\mathbf{1}000000)] \cdot (10010101)$

乘以  $x, a_7=1$

$$\begin{aligned} &= (0000000\mathbf{1}) \cdot (10010101) \oplus (000000\mathbf{1}0) \cdot (10010101) \\ &\oplus (000\mathbf{1}0000) \cdot (10010101) \oplus (00\mathbf{1}00000) \cdot (10010101) \\ &\oplus (0\mathbf{1}000000) \cdot (10010101) \end{aligned}$$

乘以  $x^4, a_7=1$

$$\begin{aligned} &= (10010101) \oplus [(00101010) \oplus (000\mathbf{1}1011)] \oplus (000\mathbf{1}0000) \cdot \\ &(\mathbf{1}0010101) \oplus (00100000) \cdot (10010101) \oplus (01000000) \cdot (10010101) \end{aligned}$$

$$\begin{aligned} &= \\ &(10010101) \oplus (00110001) \oplus [(10001000) \oplus (000\mathbf{1}1011)] \oplus (00100000) \cdot \\ &(10010101) \oplus (01000000) \cdot (10010101) \end{aligned}$$

下一步

$x^3$

$$(00010000) \cdot (10010101)$$

乘以 $x, a^7=1$

$$=(00001000) \cdot [(00101010) \oplus (00011011)]$$

$x^2$

$$=(00001000) \cdot (00110001)$$

乘以 $x, a^7=0$

$$=(00000100) \cdot (01100010)$$

乘以 $x, a^7=1$

$$=(00000010) \cdot (11000100)$$

$$=(10001000) \oplus (00011011)$$

返回

$$= (10010101) \oplus (00110001) \oplus (10010011) \oplus [(00100110) \oplus \underline{(00011011)}] \oplus (01000000) \cdot (10010101)$$

$$= (10010101) \oplus (00110001) \oplus (10010011) \oplus (00111101) \oplus (01111010)$$

$$= (01110000)$$

$$= (70)$$



## (2) 基于表操作的有限域乘法运算

- 所有的运算都在 $GF(2^8)$ 域上进行，最后的结果也都在 $GF(2^8)$ 上；
- 如果 $g \in GF(2^8)$ 是 $GF(2^8)$ 域的生成元，则 $g$ 的不同次幂 $g^p$ 将产生域的所有255个( $p=0 \sim 254$ )非零元素。
- $g^{255} = \{01\} = g^0$ ;
- 认为每一个域元素对应的 $p$ 值是一个对数值，这提供了一种将乘法转换为加法的方法；对于两个域元素 $a = g^\alpha, b = g^\beta$ , 有 $ab = g^{\alpha+\beta}$ , 当 $\alpha+\beta$ 大于十进制的255时，取mod 255运算。

- ▶ 十六进制的“03”是 $GF(2^8)$ 的一个生成元，用一个基于“03”建立起来的所有域元素的幂的对数表，就能通过查表找出任意一个域元素的基于生成元的幂指数。
- ▶ 反对数表给出了一个基于生成元{03}的幂指数表示的域元素结果。

# 例子

➤ 查对数表:

$$\{73\} = \{03\}^{(15)}, \quad \{95\} = \{03\}^{(16)}$$

$$\text{则 } \{73\} \cdot \{95\} = \{03\}^{(15)} \cdot \{03\}^{(16)}$$

$$= \{03\}^{(15) + (16)} = \{03\}^{(2B)}$$

➤ 查反对数表:

$\{03\}^{(2B)}$  表示的域元素为  $\{70\}$

$$\text{说明 } \{73\} \cdot \{95\} = \{70\}$$

# 对数表— $\{xy\}=\{03\}^L(xy)$

L(xy)		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0		00	19	01	32	02	1a	c6	4b	c7	1b	68	33	ee	df	03
	1	64	04	e0	0e	34	8d	81	ef	4c	71	08	c8	f8	69	1c	c1
	2	7d	c2	1d	b5	f9	b9	27	6a	4d	e4	a6	72	9a	c9	09	78
	3	65	2f	8a	05	21	0f	e1	24	12	f0	82	45	35	93	da	8e
	4	96	8f	db	bd	36	d0	ce	94	13	5c	d2	f1	40	46	83	38
	5	66	dd	fd	30	bf	06	8b	62	b3	25	e2	98	22	88	91	10
	6	7e	6e	48	c3	a3	b6	1e	42	3a	6b	28	54	fa	85	3d	ba
	7	2b	79	0a	15	9b	9f	5e	ca	4e	d4	ac	e5	f3	73	a7	57
	8	af	58	a8	50	f4	ea	d6	74	4f	ae	e9	d5	e7	e6	ad	e8
	9	2c	d7	75	7a	eb	16	0b	f5	59	cb	5f	b0	9c	a9	51	a0
	a	7f	0c	f6	6f	17	c4	49	ec	d8	43	1f	2d	a4	76	7b	b7
	b	cc	bb	3e	5a	fb	60	b1	86	3b	52	a1	6c	aa	55	29	9d
	c	97	b2	87	90	61	be	dc	fc	bc	95	cf	cd	37	3f	5b	d1
	d	53	39	84	3c	41	a2	6d	47	14	2a	9e	5d	56	f2	d3	ab
	e	44	11	92	d9	23	20	2e	89	b4	7c	b8	26	77	99	e3	a5
	f	67	4a	ed	de	c5	31	fe	18	0d	63	8c	80	c0	f7	70	07

# 反对数表—域元素{E}={03}(xy)

E(xy)		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	0	01	03	05	0f	11	33	55	ff	1a	2e	72	96	a1	f8	13	35
	1	5f	e1	38	48	d8	73	95	a4	f7	02	06	0a	1e	22	66	aa
	2	e5	34	5c	e4	37	59	eb	26	6a	be	d9	70	90	ab	e6	31
	3	53	f5	04	0c	14	3c	44	cc	4f	d1	68	b8	d3	6e	b2	cd
	4	4c	d4	67	a9	e0	3b	4d	d7	62	a6	f1	08	18	28	78	88
	5	83	9e	b9	d0	6b	bd	dc	7f	81	98	b3	ce	49	db	76	9a
	6	b5	c4	57	f9	10	30	50	f0	0b	1d	27	69	bb	d6	61	a3
	7	fe	19	2b	7d	87	92	ad	ec	2f	71	93	ae	e9	20	60	a0
	8	fb	16	3a	4e	d2	6d	b7	c2	5d	e7	32	56	fa	15	3f	41
	9	c3	5e	e2	3d	47	c9	40	c0	5b	ed	2c	74	9c	bf	da	75
	a	9f	ba	d5	64	ac	ef	2a	7e	82	9d	bc	df	7a	8e	89	80
	b	9b	b6	c1	58	e8	23	65	af	ea	25	6f	b1	c8	43	c5	54
	c	fc	1f	21	63	a5	f4	07	09	1b	2d	77	99	b0	cb	46	ca
	d	45	cf	4a	de	79	8b	86	91	a8	e3	3e	42	c6	51	f3	0e
	e	12	36	5a	ee	29	7b	8d	8c	8f	8a	85	94	a7	f2	0d	17
	f	39	4b	dd	7c	84	97	a2	fd	1c	24	6c	b4	c7	52	f6	01

# 查表求任意域元素的逆

方法:

- 除了  $\{00\}$  外所有的域元素都有逆;
- $g^{255} = \{01\}$
- 如果  $a = g^\alpha$  则  $a \bullet a^{-1} = 1 = g^{255}$ , 则  $a^{-1} = g^{255-\alpha}$  表示  
为:  $a^{-1} = g^{(ff)-(\alpha)}$

# 例子

➤ 查对数表:

$$\{95\} = \{03\}^{(16)}$$

➤ 其逆为:

$$\{95^{-1}\} = \{03\}^{(ff)-(16)} = \{03\}^{(e9)}$$

➤ 查反对数表:

$$\{03\}^{(e9)} = \{8a\}$$

故  $\{95\}$  的逆为  $\{8a\}$

## 2.字运算—系数在有限域 $GF(2^8)$ 上的多项式运算

AES中，全体字的集合及运算构成一个环： $(R, +, \cdot)$ ，相应的加法和乘法定义为：

设  $R$  是一个非空集合， $R$  上有两个代数运算，一个称为加法，用“+”表示，另一个称为乘法，用“ $\circ$ ”表示。如果下面三个条件成立：

- ①  $(R, +)$  是一个 Abel 群。
- ②  $(R, \circ)$  是一个半群。
- ③ 乘法对加法满足左右分配律：对  $\forall a, b, c \in R$  有

$$a \circ (b + c) = a \circ b + a \circ c$$

$$(b + c) \circ a = b \circ a + c \circ a$$

则称代数系  $(R, \circ, +)$  是一个环。



AES中，环 $(R, +, \cdot)$ 上的加法和乘法定义为：

➤ 加法 (+)

$$(a_3a_2a_1a_0) + (b_3b_2b_1b_0) = (c_3c_2c_1c_0)$$

其中， $c_i = a_i + b_i$  ( $i=0,1,2,3$ ).  $c_i = a_i + b_i$  是  $GF(2^8)$  中的加法运算。

➤ 乘法 ( “.” )

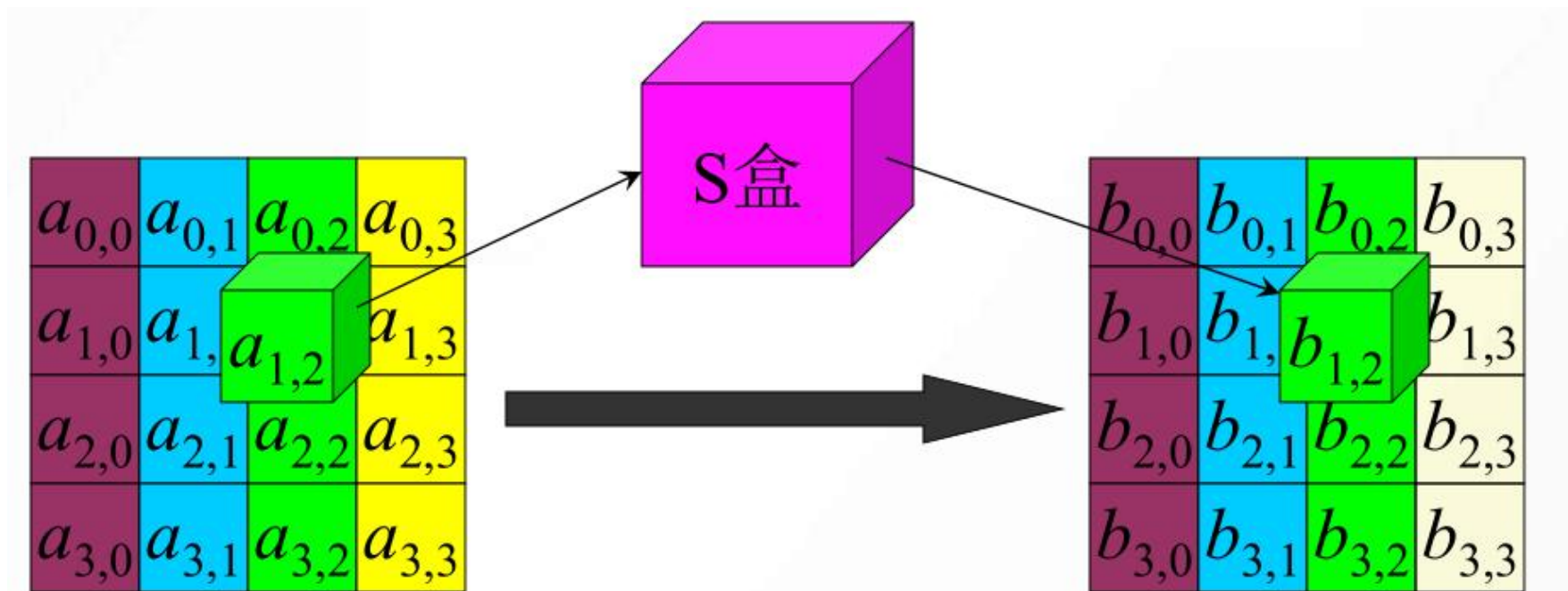
$$(a_3a_2a_1a_0) \cdot (b_3b_2b_1b_0) = (c_3c_2c_1c_0) \quad \text{其中, } c_3x^3 + c_2x^2 + c_1x^1 + c_0 \\ = (a_3x^3 + a_2x^2 + a_1x^1 + a_0)(b_3x^3 + b_2x^2 + b_1x^1 + b_0) \bmod (x^4 + 1)$$

这里的乘法为普通多项式乘法，但系数的运算是  $GF(2^8)$  中的运算。-----字的乘法转化成字节的乘法运算。

# 三、基本加密变换

## 1.S盒变换—SubBytes（字节代换）

- 是一个基于S盒的**非线性置换**，将字节通过**查表**映射为另一个字节。
- 映射方法：把输入字节的高4位作为S盒的行值，低4位作为**列值**，然后取出S盒中对应行和列的元素作为明文的输出。



S盒的替换表

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

# SubBytes变换的两个步骤:

(1) 把S盒中的每个字节（例如9行5列，写成95<sub>H</sub>）映射为它在有限域GF(2<sup>8</sup>)中的乘法逆，“0”映射为它自身。

即 $a * a^{-1} \equiv 1 \pmod{x^8 + x^4 + x^3 + x + 1}$ 。

(2) 对字节（逆元）作如下的仿射变换

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

# 例子

- “95”在GF(2<sup>8</sup>)中的乘法逆为“8a”，二进制表示为“10001010”

$$\begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Diagram illustrating the calculation of the inverse of 95 in GF(2<sup>8</sup>). The result is shown as a binary vector  $\begin{bmatrix} 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}$ , which corresponds to the hexadecimal value "2a".

Annotations:

- $b_0$  points to the first element of the result vector (0).
- $C_0$  points to the first element of the second vector (1).
- $b_7$  points to the last element of the result vector (0).
- $C_7$  points to the last element of the second vector (0).

- 结果为“00101010”，十六进制表示为“2a”

注意：

- S盒变换的第一步是把字节的值用它的乘法逆来代替，是一种非线性变换。
- 由于系数矩阵中每列都含有5个1，这说明改变输入中的任意一位，将影响输出中的5位发生变化。
- 由于系数矩阵中每行都含有5个1，这说明输出中的每一位，都与输入中的5位相关。

# S盒评价

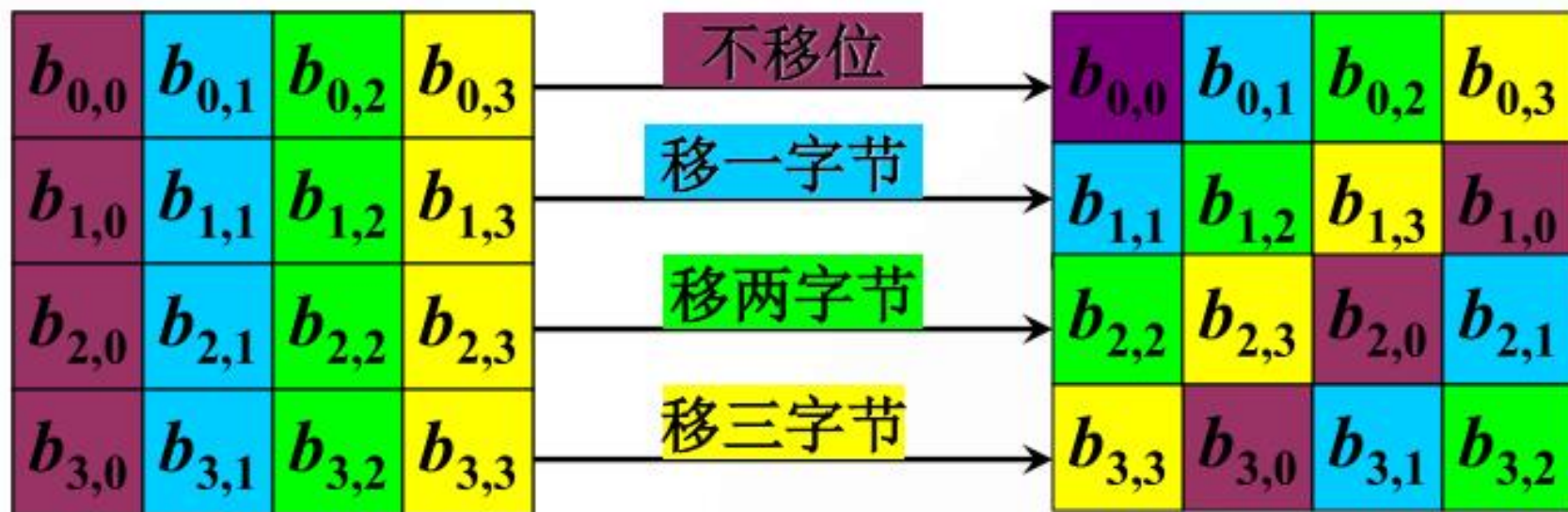
S盒被设计成能防止已有的各种密码分析攻击。Rijndael 的开发者特别寻求在输入位和输出位之间几乎没有相关性的设计，且输出值不能通过利用一个简单的数学函数变换输入值所得。另外，在代码变换中所选择的常量使得在S盒中没有不动点 $[S_{\text{盒}}(a)=a]$ ，也没有“反不动点” $[S_{\text{盒}}(a)=\text{逆}a]$ 。当然，S盒必须是可逆的，即 $\text{逆}S_{\text{盒}}[S_{\text{盒}}(a)]=a$ 。然而， $S_{\text{盒}}(a)=\text{逆}S_{\text{盒}}(a)$ 不成立，在这个意义上S盒不是自逆的。例如， $S_{\text{盒}}(\{95\})=\{2A\}$ ，但 $\text{逆}S_{\text{盒}}(\{95\})=\{AD\}$ 。

解密时使用的  
S盒的替换表



## 2.行移位运算—ShiftRows


行移位操作是作用于S盒的输出，其中，列的4个行**螺旋**地左移，即第0行左移0字节，第1行左移1字节，第2行左移2字节，第3行左移3字节，如下图所示。从该图中可以看出，这使得列完全进行了**重排**，即在移位后的每列中，都包含有未移位前每个列的一个字节。





## 行移位的例子

87	f2	4d	97
ec	6e	4c	90
4a	c3	46	e7
8c	d8	95	a6



87	f2	4d	97
6e	4c	90	ec
46	e7	4a	c3
a6	8c	d8	95

# 行移位评价

行移位虽然简单，但相当有用。由于中间态数据和算法的输入输出数据一样，也是由4列所组成的数组，在加密过程中，明文逐列被复制到中间态数据上，且后面的轮密钥也是逐列应用到中间态数据上，因此，行位移将某个字节从一列移到另一列中，这个变换确保了某列中的4字节被扩展到了4个不同的列。

### 3.列混合变换—MixColumns

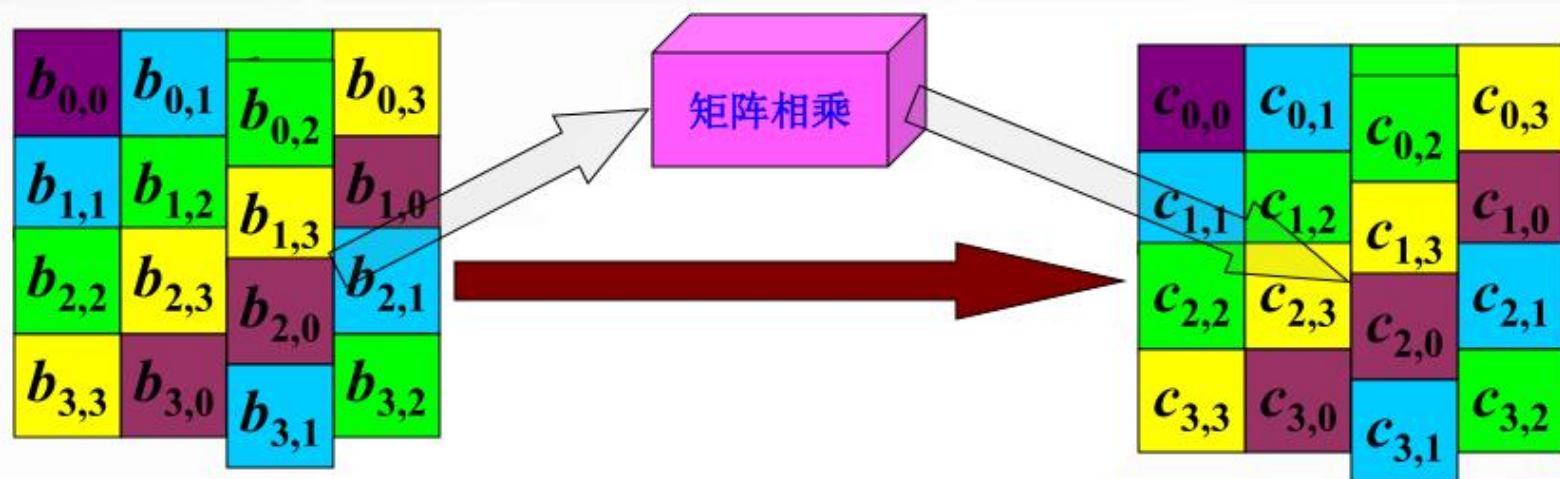
列混合变换就是把状态的每一列转换为一个新的列。实际上，转换就是一个状态列和一个常数矩阵相乘。

变换

$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix}$$

逆变换

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$



$$\begin{vmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{vmatrix} = \begin{vmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{vmatrix} \begin{vmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{vmatrix}$$

举例

$$\begin{vmatrix} 73 \\ 6b \\ ba \\ a7 \end{vmatrix} = \begin{vmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{vmatrix} \begin{vmatrix} 11 \\ 09 \\ 01 \\ 35 \end{vmatrix}$$

## 列混淆的数学基础（不作要求）

$$b(x) \equiv (b_3x^3 + b_2x^2 + b_1x + b_0) \bmod (x^4 + 1)$$

$$a(x) \equiv (a_3x^3 + a_2x^2 + a_1x + a_0) \bmod (x^4 + 1)$$

$$b(x) \equiv c(x)a(x) \bmod (x^4 + 1)$$

$$\therefore x^i \bmod (x^4 + 1) \equiv x^{i \bmod 4}$$

$$\therefore b_0 = a_0c_0 \oplus a_1c_3 \oplus a_2c_2 \oplus a_3c_1$$

$$b_1 = a_0c_1 \oplus a_1c_0 \oplus a_2c_3 \oplus a_3c_2$$

$$b_2 = a_0c_2 \oplus a_1c_1 \oplus a_2c_0 \oplus a_3c_3$$

$$b_3 = a_0c_3 \oplus a_1c_2 \oplus a_2c_1 \oplus a_3c_0$$

在AES中，取

$$c(x) = 03x^3 + 01x^2 + 01x + 02$$

$$\text{则 } c^{-1}(x) = 0Bx^3 + 0Dx^2 + 09x + 0E$$

# 列混淆例子

常数矩阵

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

态

87	f2	4d	97
6e	4c	90	ec
46	e7	4a	c3
a6	8c	d8	95

态

47	40	a3	4c
37	d4	70	9f
94	e4	3a	42
ed	a5	a6	bc

$$02 \bullet 87 \oplus 03 \bullet 6e \oplus 01 \bullet 46 \oplus 01 \bullet a6 = 47$$

$$02 \bullet 87 = (00000010) \bullet (10000111)$$

$$= (00001110) \oplus (00011011) = (00010101)$$

# 列混淆评价

- 列混淆变换的矩阵系数是基于在码字间有最大距离的**线性编码**，这使得在每列的所有字节中有良好的混淆性。列混淆变换和行移位变换使得在经过**二轮**变换后，所有的输出位均与所有的输入位相关。
- 此外，列混淆变换的系数，即 $\{01\}$ ， $\{02\}$ ， $\{03\}$ 是基于算法执行效率考虑的，而逆向列混淆变换中的系数并不是出于效率的考虑，**加密被视为比解密更重要**。

例1：设某轮的输入状态如下表所示，分别求其经过S盒代换、行移位和列混合操作的输出。

输入			
12	2A	21	0B
45	BD	04	C1
23	0A	00	1C
89	11	2A	FC



解：通过查表运算，状态矩阵中S盒代换结果如表右半部分所示。

输入				输出			
12	2A	21	0B	C9	E5	FD	2B
45	BD	04	C1	6E	7A	F2	78
23	0A	00	1C	26	67	63	9C
89	11	2A	FC	A7	82	E5	B0

解：通过行移位，状态矩阵中的输出如表右半部分所示。

输入				输出			
C9	E5	FD	2B	C9	E5	FD	2B
6E	7A	F2	78	7A	F2	78	6E
26	67	63	9C	63	9C	26	67
A7	82	E5	B0	B0	A7	82	E5

下面进行列混合运算：

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{vmatrix} C9 \\ 7A \\ 63 \\ B0 \end{vmatrix} = \begin{vmatrix} D4 \\ 28 \\ BE \\ 22 \end{vmatrix}$$

计算过程：

$$02 \bullet C9 \oplus 03 \bullet 7A \oplus 01 \bullet 63 \oplus 01 \bullet B0 = D4$$

$$01 \bullet C9 \oplus 02 \bullet 7A \oplus 03 \bullet 63 \oplus 01 \bullet B0 = 28$$

$$01 \bullet C9 \oplus 01 \bullet 7A \oplus 02 \bullet 63 \oplus 03 \bullet B0 = BE$$

$$03 \bullet C9 \oplus 01 \bullet 7A \oplus 01 \bullet 63 \oplus 02 \bullet B0 = 22$$

详细过程：

$$02 \bullet C9 = (10010010) \oplus (00011011) = 89$$

$$03 \bullet 7A = (11110100) \oplus (01111010) = 8E$$

$$89 \oplus 8E \oplus 63 \oplus B0 = D4$$

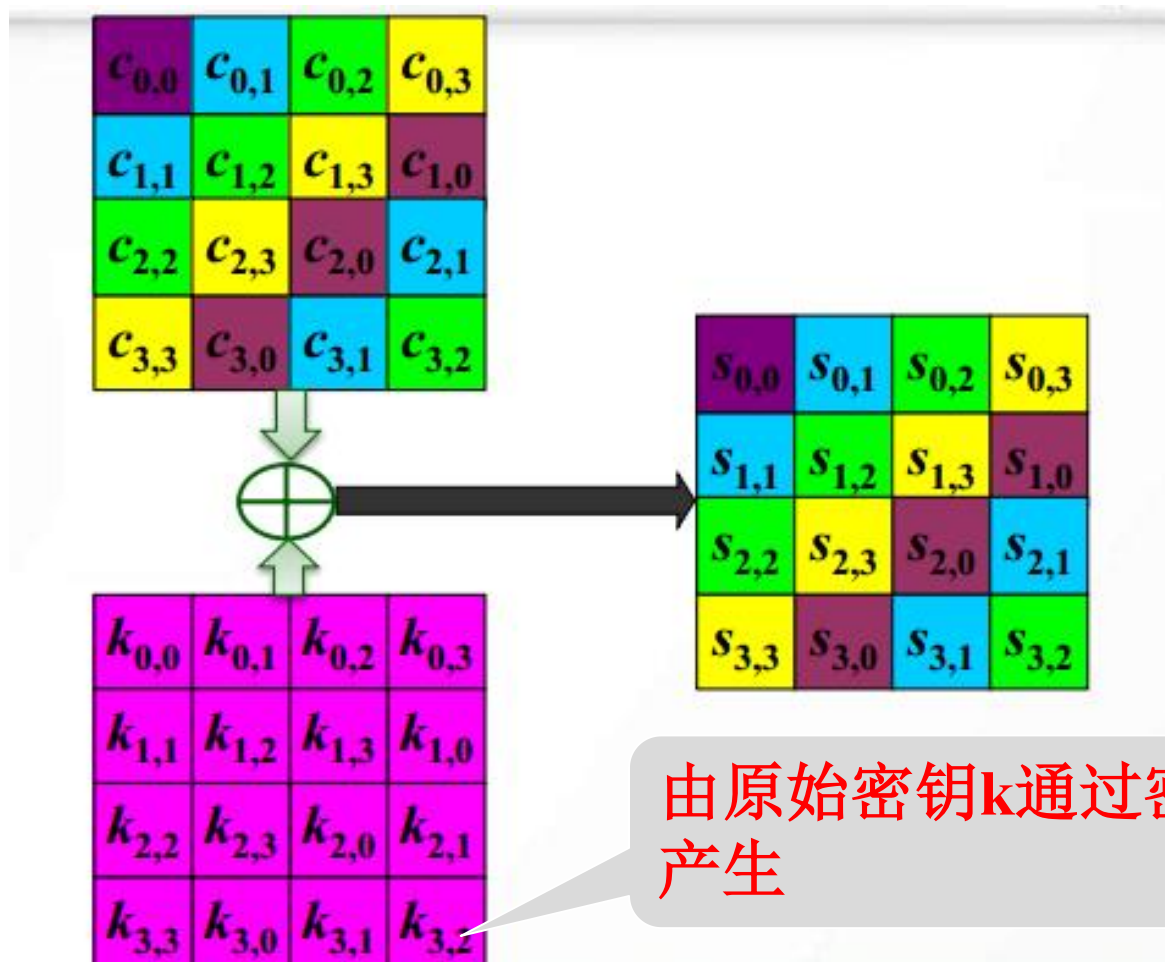
类似地，可以得到其他各项的结果：

### 列混合

输入				输出			
C9	E5	FD	2B	D4	E7	CD	66
7A	F2	78	6E	28	02	E5	BB
63	9C	26	67	BE	C6	54	BF
B0	A7	82	E5	22	0F	5D	A5

## 4.轮密钥加变换—AddRoundKey

轮密钥加就是一个轮密钥字与每一个状态列矩阵相加，即进行按位异或。



# AES的密钥生成

- ◆ AES首先将初始密钥输入到一个 $4 \times 4$ 矩阵中。这个 $4 \times 4$ 矩阵的每一列的4个字节组成一个字，矩阵4列的4个字依次命名为 $w[0]$ 、 $w[1]$ 、 $w[2]$ 、 $w[3]$ ，它们构成了一个以字为单位的数值 $w$ 。
- ◆  $w[0]$ 、 $w[1]$ 、 $w[2]$ 和 $w[3]$  作为扩展密钥的基础,然后添加 40 个新列来进行扩充,新列以递归方式产生。

## ◆ 扩展密钥:

- 对w数组中下标不为4的倍数的元素:

$$w[i] = w[i-1] \oplus w[i-4] \quad (i \text{ 不为 } 4 \text{ 的倍数})$$

- 对w数组中下标为4的倍数的元素:

$$w[i] = w[i-4] \oplus T(w[i-1]) \quad (i \text{ 为 } 4 \text{ 的倍数})$$

其中，T是一个复杂的函数，其产生过程为:

- 将一个字的四个字节循环左移一个字节，即  
 $[b_0, b_1, b_2, b_3] \rightarrow [b_1, b_2, b_3, b_0]$
- 基于S盒对输入字中的每个字节进行S代替;
- 将步骤1和步骤2的结果再与轮常量Rcon[i]相异或 (i表示轮数) .



# 前10个轮常数Rcon[i]

表 5.17  $R_{\text{con}}[i]$

$i$	1	2	3	4	5
$R_{\text{con}}[i]$	01000000	02000000	04000000	08000000	10000000
$i$	6	7	8	9	10
$R_{\text{con}}[i]$	20000000	40000000	80000000	1b000000	36000000

使用与轮相关的轮常量是为了防止不同轮中产生的轮密钥的对称性或相似性。

# 轮密钥加的例子

态

32	88	31	e0
43	5a	31	37
f6	30	98	07
a8	8d	a2	34

$\oplus$

轮密钥

2b	28	ab	09
7e	ae	f7	cf
15	d2	15	4f
16	a6	88	3c

$=$

新态

19	a0	9a	e9
3d	f4	c6	f8
e3	e2	8d	48
be	2b	2a	08

$$32 \oplus 2b = (00110010) \oplus (00101011) = 19_H$$

$$43 \oplus 7e = (01000011) \oplus (01111110) = 3d_H$$

$$f6 \oplus 15 = (11110110) \oplus (00010101) = e3_H$$

$$a8 \oplus 16 = (10101000) \oplus (00010110) = be_H$$

# 密钥扩展评价

- 创建每一轮的轮密钥，AES使用的是**密钥扩展程序**。
- 知道密钥或轮密钥的**部分位**不能计算出轮密钥的其它位，但知道扩展密钥中任何连续的 $N_k$ 个字能够重新产生整个扩展密钥（ $N_k$ 是不同版本AES的密钥长度）。
- 使用轮常量来排除**对称性**。
- 密钥的每一位**快速且均等**地影响到轮密钥的位。
- 足够的**非线性**以防止轮密钥的差异完全由密钥的差异所决定。

例2：如果初始的128位密钥为：

3C A1 0B 21 57 F0 19 16 90 2E 13 80 AC C1 07 BD

那么4个初始值为：

$w[0] = 3C\ A1\ 0B\ 21$

$w[1] = 57\ F0\ 19\ 16$

$w[2] = 90\ 2E\ 13\ 80$

$w[3] = AC\ C1\ 07\ BD$

求扩展的第1轮的子密钥 ( $w[4], w[5], w[6], w[7]$ ) .

解：由于4是4的倍数，所以 $w[4]=w[0]+T(w[3])$

$T(w[3])$ 的计算步骤如下：

(1) 循环地将 $w[3]$ 的元素移位：AC C1 07 BD变成了C1 07 BD AC；

(2) 将C1 07 BD AC作为S盒的输入，输出为78 C5 7A 91

(3) 将78 C5 7A 91与第1轮轮常量Rcon[1]进行异或运算，将得到 79 C5 7A 91.

因此， $T(w[3])=79\ C5\ 7A\ 91$ ，故

$$w[4]=79\ C5\ 7A91 \oplus 3C\ A1\ 0B\ 21 = 45\ 64\ 71\ B0$$

其余3个子密钥计算如下：

$$w[5] = w[1] + w[4]$$

$$= 57 \text{ } F0 \text{ } 19 \text{ } 16 \oplus 45 \text{ } 64 \text{ } 71 \text{ } B0 = 12 \text{ } 94 \text{ } 68 \text{ } A6$$

$$w[6] = w[2] + w[5]$$

$$= 90 \text{ } 2E \text{ } 13 \text{ } 80 \oplus 12 \text{ } 94 \text{ } 68 \text{ } A6 = 82 \text{ } BA \text{ } 7B \text{ } 26$$

$$w[7] = w[3] + w[6]$$

$$= AC \text{ } C1 \text{ } 07 \text{ } BD \oplus 82 \text{ } BA \text{ } 7B \text{ } 26 = 2E \text{ } 7B \text{ } 7C \text{ } 9B$$

于是，第1轮的密钥为：

45 64 71 B0 12 94 68 A6 82 BA 7B 26 2E 7B 7C 9B.

## 5.3.4 AES解密

基本运算中除轮密钥加AddRoundKey不变外（实际上按位异或操作的逆变换是其本身），其余的字节代换SubBytes、行移位ShiftRows、列混淆MixColumns都要进行求逆变换，即InvSubBytes、InvShiftRows、InvMixColumns。

# 1.逆字节变换—InvSubBytes

基于逆S盒实现。

87	f2	4d	97
ec	6e	4c	90
4a	c3	46	e7
f0	2d	ad	c5



ea	04	65	85
83	45	5d	96
5c	33	98	b0
a6	8c	d8	95



表 5.15 逆 S 盒(十六进制)

		y															
		0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
x	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

1.对字节作仿射变换的逆变换，变换如下：

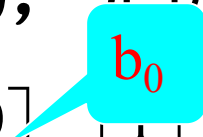
$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} b'_0 \\ b'_1 \\ b'_2 \\ b'_3 \\ b'_4 \\ b'_5 \\ b'_6 \\ b'_7 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

2.计算乘法逆。

## 例子

求2a的逆字节代换.

(1) 2a的二进制表示为00101010, 先进行仿射逆变换:

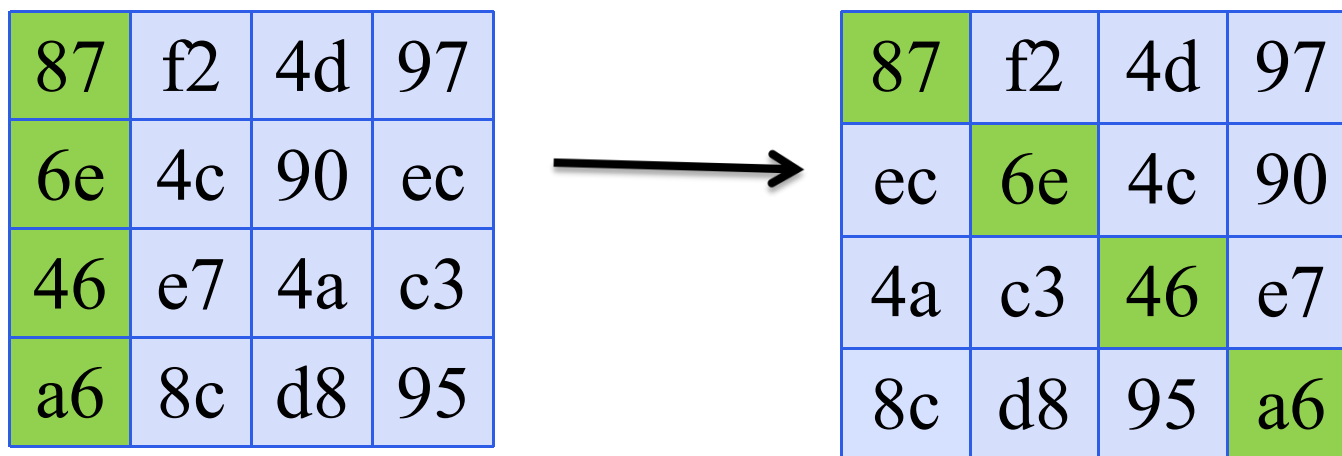
$$\begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \\ b_4 \\ b_5 \\ b_6 \\ b_7 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$


结果为“10001010”, 十六进制表示为“8a”

(2) 8a在GF(2<sup>8</sup>)的乘法逆为95.

## 2.逆行移位—InvShiftRows

与行移位相反，逆行移位将态State的后三行按相反的方向进行移位，即第1行保持不变，第2行循环向右移动1个字节，第3行循环向右移动2个字节，第4行循环向右移动3个字节。



### 3.逆列混合变换—InvMixColumns

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \end{bmatrix} = \begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

# 逆列混淆例子

常数矩阵

0e	0b	0d	09
09	0e	0b	0d
0d	09	0e	0b
0b	0d	09	0e

态

47	40	a3	4c
37	d4	70	9f
94	e4	3a	42
ed	a5	a6	bc

态

87	f2	4d	97
6e	4c	90	ec
46	e7	4a	c3
a6	8c	d8	95

$$0e \bullet 47 \oplus 0b \bullet 37 \oplus 0d \bullet 94 \oplus 09 \bullet ed = 87_H$$

## 也可查对数表和反对数表

$$\begin{aligned} 0e \bullet 47 &= \{03\}^{(df)} \bullet \{03\}^{(94)} = \{03\}^{(df)+(94)} \\ &= \{03\}^{(74)} = 87_H = (10000111) \end{aligned}$$

$$\begin{aligned} 0b \bullet 37 &= \{03\}^{(68)} \bullet \{03\}^{(24)} = \{03\}^{(68)+(24)} \\ &= \{03\}^{(8c)} = fa_H = (11111010) \end{aligned}$$

$$\begin{aligned} 0d \bullet 94 &= \{03\}^{(ee)} \bullet \{03\}^{(eb)} = \{03\}^{(ee)+(eb)} \\ &= \{03\}^{(da)} = 3e_H = (00111110) \end{aligned}$$

$$\begin{aligned} 09 \bullet ed &= \{03\}^{(c7)} \bullet \{03\}^{(99)} = \{03\}^{(c7)+(99)} \\ &= \{03\}^{(61)} = c4_H = (11000100) \end{aligned}$$

$$0e \bullet 47 \oplus 0b \bullet 37 \oplus 0d \bullet 94 \oplus 09 \bullet 4d$$

$$= (10000111) \oplus (11111010)$$

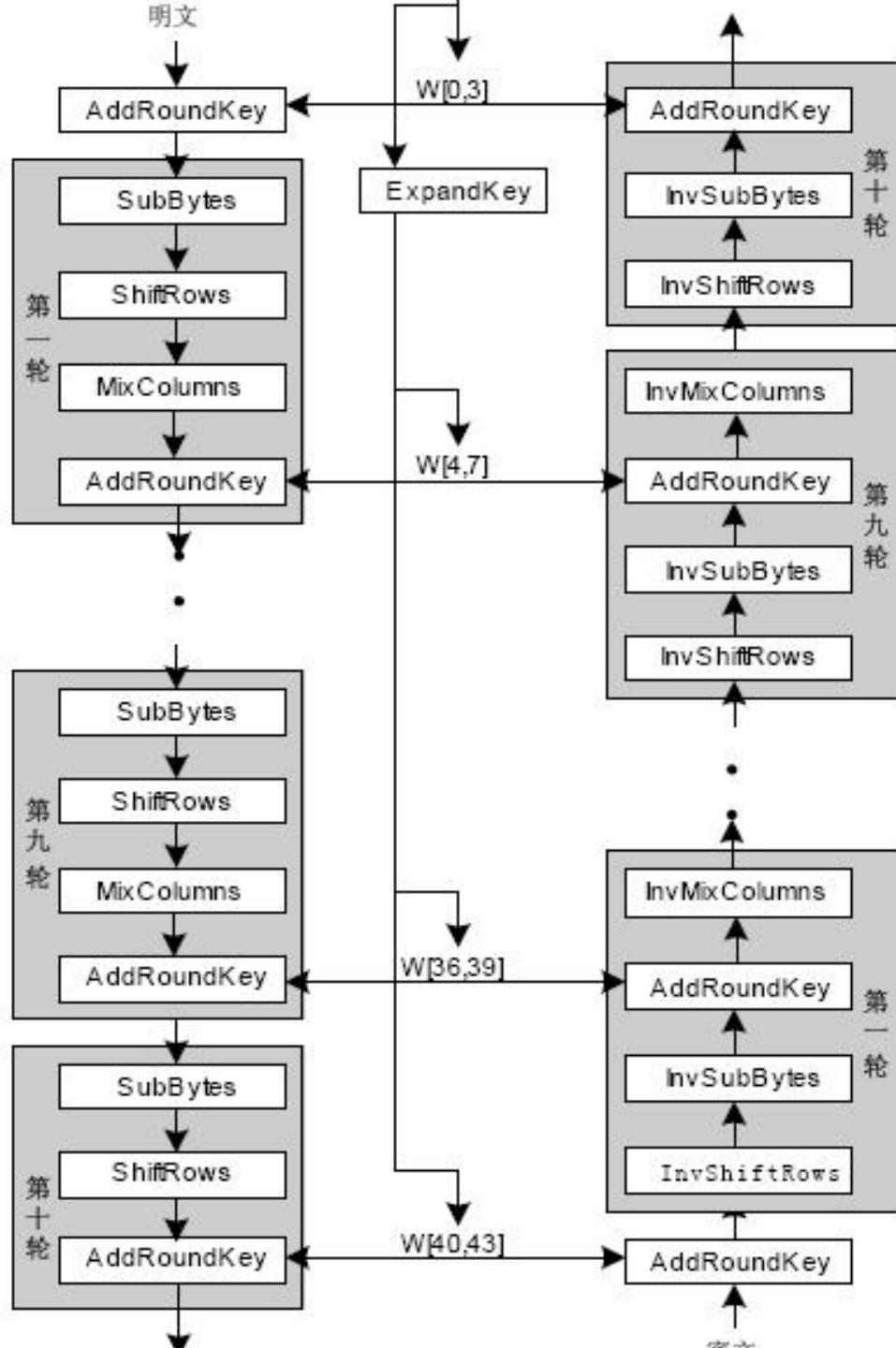
$$\oplus (00111110) \oplus (11000100)$$

$$= (10000111)$$

$$= 87_H$$



# AES加解密流程图



# AES的加密的伪代码

**Cipher(byte in[4\*Nb], byte out[4\*Nb], word w[Nb\*(Nr+1)])**

**begin**

**byte state[4,Nb]**

**state = in**

**AddRoundKey(state, w)**

**for round = 1 step 1 to Nr-1**

**SubBytes(state)**

**ShiftRows(state)**

**MixColumns(state)**

**AddRoundKey(state, w+round\*Nb)**

**end for**

**SubBytes(state)**

**ShiftRows(state)**

**AddRoundKey(state, w+Nr\*Nb)**

**out = state**

**end**

# AES的解密的伪代码

```
InvCipher(byte in[4*Nb], byte out[4*Nb], word w[Nb*(Nr+1)])  
begin  
    byte state[4,Nb]  
    state = in  
    AddRoundKey(state, w[Nr*Nb, (Nr+1)*Nb-1])  
    for round = Nr-1 step -1 downto 1  
        InvShiftRows(state)  
        InvSubBytes(state)  
        AddRoundKey(state, w[round*Nb, (round+1)*Nb-1])  
        InvMixColumns(state)  
    end for  
    InvShiftRows(state)  
    InvSubBytes(state)  
    AddRoundKey(state, w[0, Nb-1])  
    out = state  
end
```

# AES设计上的考虑

- ◆ 与DES相比，扩散的效果更快，即**两轮**可达到完全扩散。
- ◆ S盒使用**清晰而简单**的代数方法构造，避免任何对算法留有后门的怀疑。
- ◆ 密钥扩展方案实现对密钥位的非线性混合，既实现了**均等效应**，也实现了**非对称性**。
- ◆ 比穷举攻击更好的攻击进行到**6轮**，多出4轮可以提供足够多的安全性。**(注：针对AES-128)**。

在第一轮之前，用了一个初始密钥加层，其目的是在不知道密钥的情况下，对最后一个密钥加层以后的任一层（或者是当进行已知明文攻击时，对第一个密钥加层以前的任一层）可简单地剥去，因此初始密钥加层对密码的安全性无任何意义。许多密码的设计中都在轮变换之前和之后用了密钥加层，如IDEA、SAFER和Blowfish。

为了使加密算法和解密算法在结构上更加接近，最后一轮的线性迭代混合层与前面各轮的线性混合层不同，这类似于DES的最后一轮不做左右交换。可以证明这种设计不以任何方式提高或降低改密码的安全性。

# AES的安全性

## ◆ 弱密钥

AES在设计上不是对称的，其加密和解密过程不一致，这也避免弱密钥的存在。

## ◆ 差分分析和线性分析

由于在设计时考虑了这两种攻击的方法，因此AES具有较好的抗击其攻击的能力。

## ◆ 密钥穷举攻击

平均需要 $2^{127}$ 次AES运算，按目前的计算能力是不可能完成的。

# AES和DES相似之处

- 二者的轮函数都是由**四层**构成，非线性层、移位层、线性混合层、子密钥异或，只是顺序不同。
- AES的非线性运算是**字节代替**(ByteSub)，对应于DES中的非线性运算**S盒**。
- **行移位**运算保证了每一行的字节不仅仅影响其它行对应的字节，而且影响其它行所有的字节，这与DES中**置换P**相似。
- AES的**列混淆**运算的目的是让不同的字节相互影响，而DES中F函数的输出与左边一半数据相加也有类似的效果。
- AES的**子密钥异或**对应于DES中S盒之前的**子密钥异或**。



# AES和DES主要不同之处

- AES的密钥长度(128位、192位、256位)是可变的，而DES的密钥长度固定为56位。
- DES是面向比特的运算，AES是面向字节的运算。
- AES的加密运算和解密运算不一致，因而加密器不能同时用作解密器，而DES的加密器可用作解密器，只是子密钥的顺序不同。

# 本节主要内容

- ◆ AES的整体结构
- ◆ AES的轮函数（4个变换）
- ◆ AES的密钥编排算法
- ◆ AES的解密算法

# 作业

P132

5.11 5.13 5.14 5.15

谢谢！