附录 D　简化 AES

简化 AES（S-AES）是由圣塔·克拉拉大学的 Edward Schaefer 教授及其几名学生开发的，是一个面向教育的算法，但不是安全的加密算法[MUSA03]。它与 AES 的性质和结构类似，但使用的参数更少。读者跟随本附录动手完成一个例子后，应会有所收获。深入了解 S-AES 可以让读者更容易掌握 AES 的结构与操作。

D.1　概述

图 D.1 显示了 S-AES 的整体结构。加密算法使用一个 16 位明文分组和一个 16 位密钥作为输入，生成一个 16 位密文分组作为输出。S-AES 解密算法用一个 16 位密文分组和相同的密钥作为输入，生成原始的 16 位明文分组作为输出。

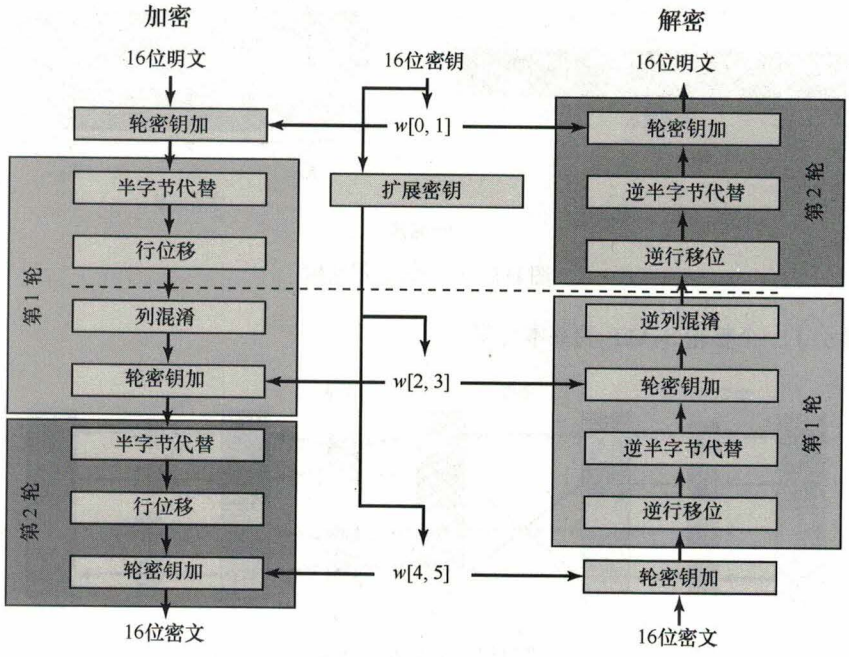


图 D.1　S-AES 加密和解密

加密算法使用 4 个不同的函数或变换：密钥加（A_K）、半字节代替（NS）、行移位（SR）和列混淆（MC）。下面介绍这些操作。

我们可以简单地将加密算法表示为一个复合函数[①]：

$$A_{K_2} \circ \mathrm{SR} \circ \mathrm{NS} \circ A_{K_1} \circ \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{NS} \circ A_{K_0}$$

① 定义：若 f 和 g 是两个函数，则称函数 $F(x) = g[f(x)]$ 是 f 和 g 的复合函数，表示为 $F = g \circ f$。

因此，首先应用 A_{K_0}。

加密算法被组织成三轮。第 0 轮是简单的密钥加轮；第 1 轮是包含 4 个函数的完整轮；第 2 轮仅包含 3 个函数。每轮都使用 16 位密钥的密钥加函数。初始的 16 位密钥被扩展到 48 位，以便每轮都可使用一个不同的轮密钥。

每个函数都可在一个被视为 2×2 半字节矩阵的 16 位状态上操作，其中半字节是 4 比特。状态矩阵的初值是 16 比特明文。加密过程中的每个后续函数修改状态，经最后一个函数处理后的结果是 16 比特密文。如图 D.2(a)所示，矩阵内的半字节是按列排序的。因此，加密密码的 16 比特明文的前 8 比特占据矩阵的第一列，第二个 8 比特占据第二列。16 比特密钥也采用类似的组织方式，但将密钥视为 2 字节而非 4 个半字节要方便一些［见图 D.2(b)］。扩展的 48 比特密钥可作为 3 个轮密钥处理，其比特标记如下：$K_0 = k_0 \cdots k_{15}$；$K_1 = k_{16} \cdots k_{31}$；$K_2 = k_{32} \cdots k_{47}$。

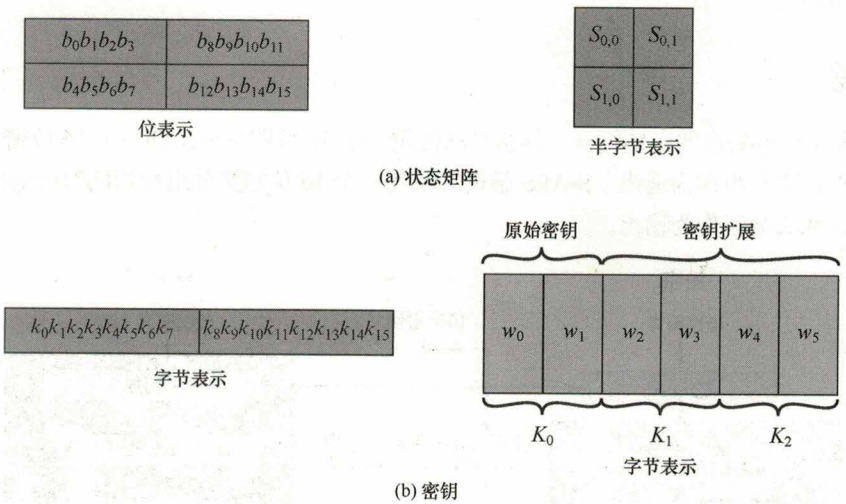


图 D.2 S-AES 数据结构

图 D.3 显示了一个整轮 S-AES 的基本元素。

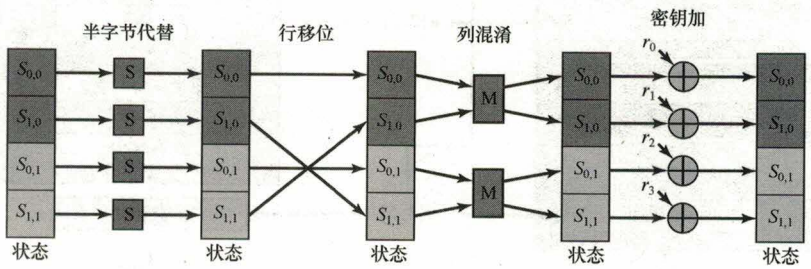


图 D.3 S-AES 加密轮

解密也显示在图 D.1 中，它本质上是加密的逆：

$$A_{K_0} \circ \text{INS} \circ \text{ISR} \circ \text{IMC} \circ A_{K_1} \circ \text{INS} \circ \text{ISR} \circ A_{K_2}$$

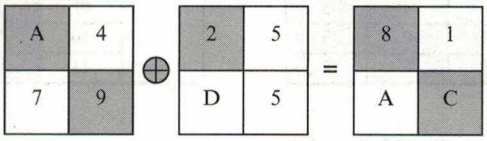
其中的三个函数都有一个对应的逆函数：逆半字节代替（INS）、逆行位移（ISR）和逆列混淆（IMC）。

D.2 S-AES 加密与解密

下面介绍加密算法中的每个函数。

D.2.1　密钥加

密钥加函数将 16 位状态矩阵与 16 位轮密钥逐位异或。图 D.4 将其描述为逐列操作，但它也可视为逐半字节操作或逐位操作。下面是一个例子。



由于异或运算是其本身的逆运算，因此密钥加函数的逆函数与密钥加函数相同。

D.2.2　半字节代替

半字节代替函数是简单的查表操作（见图 D.4）。AES 定义一个 4×4 的半字节值矩阵，称为 S 盒[见表 D.1(a)]，其中包含所有 4 位值的排列。状态中的每个半字节都按以下方式映射到一个新的半字节：半字节最左侧的 2 位用作行值，最右侧的 2 位用作列值。这些行和列的值用作 S 盒中选择唯一的 4 位输出值的索引。例如，十六进制值 A 代表 S 盒中第 2 行、第 2 列的值 0。因此，值 A 被映射为值 0。

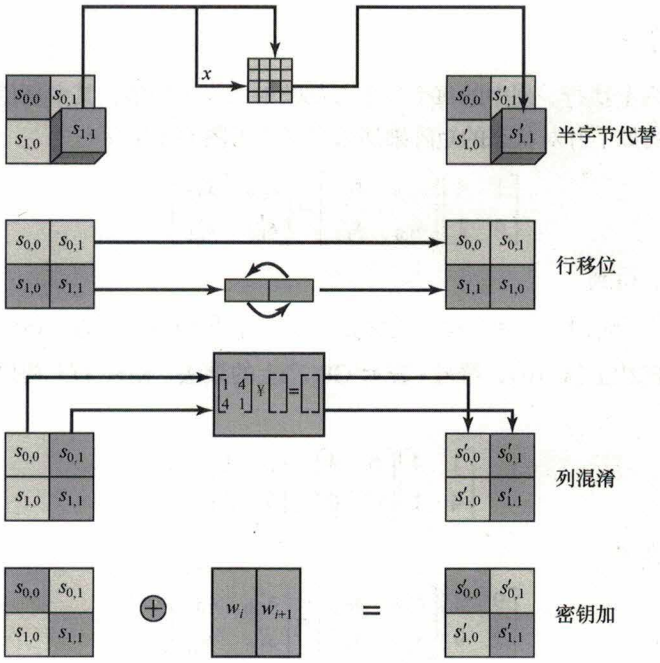
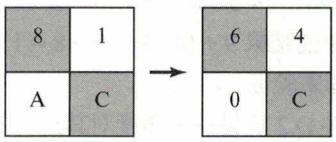


图 D.4　S-AES 变换

下面是一个半字节代替变换的例子。



半字节代替函数的逆函数使用表 D.1(b)中的逆 S 盒。例如，输入 0 生成输出 A，A 输入 S 盒后生成 0。

表 D.1 S-AES 的 S 盒

		j			
		00	01	10	11
i	00	9	4	A	B
	01	D	1	8	5
	10	6	2	0	3
	11	C	E	F	7

(a) S 盒

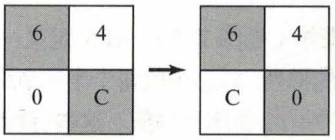
		j			
		00	01	10	11
i	00	A	5	9	B
	01	1	7	8	F
	10	6	0	2	3
	11	C	4	D	E

(b) 逆 S 盒

注意：阴影格中的是十六进制数，非阴影格中是二进制数。

D.2.3 行移位

行移位函数在状态的第二行执行一个半字节循环移位。第一行不变（见图 D.4）。下面举一个例子。



由于逆行移位函数将第二行移回原来的位置，因此逆行移位函数和行移位函数相同。

D.2.4 列混淆

列混淆函数在各列上执行。列中的每个半字节都映射为一个新值，其中新值是该列中两个半字节的函数。这个变换可由如下针对状态的矩阵乘法来定义（见图 D.4）：

$$\begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} \\ s_{1,0} & s_{1,1} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} \\ s'_{1,0} & s'_{1,1} \end{bmatrix}$$

执行矩阵乘法后，得到

$$s'_{0,0} = s_{0,0} \oplus (4 \bullet s_{1,0}),\ s'_{1,0} = (4 \bullet s_{0,0}) \oplus s_{1,0},\ s'_{0,1} = s_{0,1} \oplus (4 \bullet s_{1,1}),\ s'_{1,1} = (4 \bullet s_{0,1}) \oplus s_{1,1}$$

其中算术运算是在 GF(2^4)上执行的，符号 • 表示 GF(2^4)上的乘法。附件 D.1 提供了加法表和乘法表。下面是一个例子。

$$\begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} 6 & 4 \\ C & 0 \end{bmatrix} = \begin{bmatrix} 3 & 4 \\ 7 & 3 \end{bmatrix}$$

逆列混淆函数定义为

$$\begin{bmatrix} 9 & 2 \\ 2 & 9 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} \\ s_{1,0} & s_{1,1} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} \\ s'_{1,0} & s'_{1,1} \end{bmatrix}$$

下面证明我们确实定义了逆列混淆函数：

$$\begin{bmatrix} 9 & 2 \\ 2 & 9 \end{bmatrix} \begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} \\ s_{1,0} & s_{1,1} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} \\ s_{1,0} & s_{1,1} \end{bmatrix} = \begin{bmatrix} s_{0,0} & s_{0,1} \\ s_{1,0} & s_{1,1} \end{bmatrix}$$

上面的矩阵乘法使用了 GF(2^4)中的结果 $9 + (2 \bullet 4) = 9 + 8 = 1$ 和 $(9 \bullet 4) + 2 = 2 + 2 = 0$。这些运算可用附件 D.1 中的算术表或多项式运算来验证。

列混淆函数非常抽象，因此附件 D.2 从另一个角度对其进行了介绍。

D.3　密钥扩展

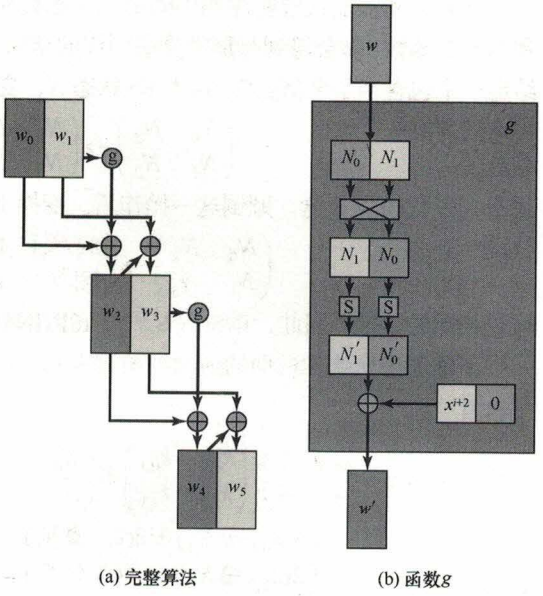
对于密钥扩展而言，16 位初始密钥被分成两个 8 位字。图 D.5 显示了扩展为 6 个字的过程，方法是由最初的 2 个字计算 4 个新字。算法如下：

$$w_2 = w_0 \oplus g(w_1) = w_0 \oplus \text{RCON}(1) \oplus \text{SubNib}(\text{RotNib}(w_1))$$
$$w_3 = w_2 \oplus w_1$$
$$w_4 = w_2 \oplus g(w_3) = w_2 \oplus \text{RCON}(2) \oplus \text{SubNib}(\text{RotNib}(w_3))$$
$$w_5 = w_4 \oplus w_3$$

RCON 是一个轮常数：RC[i] = x^{i+2}，有 RC[1] = x^3 = 100，RC[2] = $x^4 \bmod (x^4 + x + 1) = x + 1 = 0011$。RC[$i$]组成 1 字节的左侧半字节，右侧半字节都是 0。因此，RCON(1) = 10000000，RCON(2) = 00110000。

例如，假设密钥是 2D55 = 0010 1101 0101 0101 = w_0w_1，则有

w_2 = 00101101⊕10000000⊕SubNib(01010101)
= 00101101⊕10000000⊕00010001 = 10111100
w_3 = 10111100⊕01010101 = 11101001
w_4 = 10111100⊕00110000⊕SubNib(10011110)
= 10111100⊕00110000⊕00101111 = 10100011
w_5 = 10100011⊕11101001 = 01001010



(a) 完整算法　(b) 函数g

图 D.5　S-AES 密钥扩展

D.4　S 盒

S 盒的构建过程如下。

1. 使用逐行递增的半字节值序列初始化 S 盒。第一行包含十六进制值(0, 1, 2, 3)，第二行包含十六进制值(4, 5, 6, 7)，以此类推。因此，第 i 行、第 j 列的半字节值是 $4i + j$。
2. 将每个半字节视为模 $x^4 + x + 1$ 的有限域 GF(2^4)上的一个元素。每个半字节 $a_0a_1a_2a_3$ 代表一个阶为 3 的多项式。
3. 将 S 盒中的每个字节都映射为模 $x^4 + x + 1$ 的有限域 GF(2^4)上的乘法逆运算。值 0 映射为其本身。
4. 假设 S 盒中的每个字节都包含标为(b_0, b_1, b_2, b_3)的 4 位。对 S 盒中每个字节的每位应用如下变换。AES 标准以矩阵形式描述了这个变换：

$$\begin{bmatrix} b_0' \\ b_1' \\ b_2' \\ b_3' \end{bmatrix} = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix} \oplus \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

符号（'）表明变量使用右边的值更新。记住，加法和乘法以模 2 来计算。

表 D.1(a)中显示了得到的 S 盒。这是一个非线性的可逆矩阵。表 D.1(b)中显示了逆 S 盒。

D.5　S-AES 的结构

下面介绍 AES 结构的许多有趣的方面。第一个方面是，加密和解密算法都以密钥加函数开始和结束。首尾的其他函数在不知道密钥的情况下是可逆的，所以未增加安全性，而只是一种处理开销。因

此，第 0 轮只包含密钥加函数。

第二个方面是，第 2 轮不包含列混淆函数，原因与第三个方面有关：虽然通过图 D.1 可以清楚地看出解密算法是加密算法的逆运算，但是解密算法使用函数的顺序不同。因此，有

加密： $A_{K_2} \circ \mathrm{SR} \circ \mathrm{NS} \circ A_{K_1} \circ \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{NS} \circ A_{K_0}$

解密： $A_{K_0} \circ \mathrm{INS} \circ \mathrm{ISR} \circ \mathrm{IMC} \circ A_{K_1} \circ \mathrm{INS} \circ \mathrm{ISR} \circ A_{K_2}$

从实现角度看，我们希望解密与加密算法采用同一个函数序列，以便实现解密算法的方式与实现加密算法的方式相同，进而提高效率。

注意，若能交换解密序列中的第二个函数和第三个函数、第四个函数和第五个函数、第六个函数和第七个函数，则会得到与加密算法同样的结构。下面来看看这是否可行。首先，考虑交换 INS 和 ISR。给定一个包含半字节(N_0, N_1, N_2, N_3)的状态 N，变换 INS(ISR(N))的过程是

$$\begin{pmatrix} N_0 & N_2 \\ N_1 & N_3 \end{pmatrix} \rightarrow \begin{pmatrix} N_0 & N_2 \\ N_3 & N_1 \end{pmatrix} \rightarrow \begin{pmatrix} \mathrm{IS}[N_0] & \mathrm{IS}[N_2] \\ \mathrm{IS}[N_3] & \mathrm{IS}[N_1] \end{pmatrix}$$

式中，IS 代表逆 S 盒。颠倒这一操作后，变换 ISR(INS(N))的过程是

$$\begin{pmatrix} N_0 & N_2 \\ N_1 & N_3 \end{pmatrix} \rightarrow \begin{pmatrix} \mathrm{IS}[N_0] & \mathrm{IS}[N_2] \\ \mathrm{IS}[N_1] & \mathrm{IS}[N_3] \end{pmatrix} \rightarrow \begin{pmatrix} \mathrm{IS}[N_0] & \mathrm{IS}[N_2] \\ \mathrm{IS}[N_3] & \mathrm{IS}[N_1] \end{pmatrix}$$

这是相同的结果。因此，INS(ISR(N)) = ISR(INS(N))。

下面考虑密钥加后面的逆列混淆运算 $\mathrm{IMC}(A_{K_1}(N))$，其中轮密钥 K_1 包含半字节$(k_{0,0}, k_{1,0}, k_{0,1}\ k_{1,1})$。于是有

$$\begin{pmatrix} 9 & 2 \\ 2 & 9 \end{pmatrix}\left(\begin{pmatrix} k_{0,0} & k_{0,1} \\ k_{1,0} & k_{1,1} \end{pmatrix} \oplus \begin{pmatrix} N_0 & N_2 \\ N_1 & N_3 \end{pmatrix}\right) = \begin{pmatrix} 9 & 2 \\ 2 & 9 \end{pmatrix}\begin{pmatrix} k_{0,0} \oplus N_0 & k_{0,1} \oplus N_2 \\ k_{1,0} \oplus N_1 & k_{1,1} \oplus N_3 \end{pmatrix}$$

$$= \begin{pmatrix} 9(k_{0,0} \oplus N_0) \oplus 2(k_{1,0} \oplus N_1) & 9(k_{0,1} \oplus N_2) \oplus 2(k_{1,1} \oplus N_3) \\ 2(k_{0,0} \oplus N_0) \oplus 9(k_{1,0} \oplus N_1) & 2(k_{0,1} \oplus N_2) \oplus 9(k_{1,1} \oplus N_3) \end{pmatrix}$$

$$= \begin{pmatrix} (9k_{0,0} \oplus 2k_{1,0}) \oplus (9N_0 \oplus 2N_1) & (9k_{0,1} \oplus 2k_{1,1}) \oplus (9N_2 \oplus 2N_3) \\ (2k_{0,0} \oplus 9k_{1,0}) \oplus (2N_0 \oplus 9N_1) & (2k_{0,1} \oplus 9k_{1,1}) \oplus (2N_2 \oplus 9N_3) \end{pmatrix}$$

$$= \begin{pmatrix} (9k_{0,0} \oplus 2k_{1,0}) & (9k_{0,1} \oplus 2k_{1,1}) \\ (2k_{0,0} \oplus 9k_{1,0}) & (2k_{0,1} \oplus 9k_{1,1}) \end{pmatrix} \oplus \begin{pmatrix} (9N_0 \oplus 2N_1) & (9N_2 \oplus 2N_3) \\ (2N_0 \oplus 9N_1) & (2N_2 \oplus 9N_3) \end{pmatrix}$$

$$= \begin{pmatrix} 9 & 2 \\ 2 & 9 \end{pmatrix}\begin{pmatrix} k_{0,0} & k_{0,1} \\ k_{1,0} & k_{1,1} \end{pmatrix} \oplus \begin{pmatrix} 9 & 2 \\ 2 & 9 \end{pmatrix}\begin{pmatrix} N_0 & N_2 \\ N_1 & N_3 \end{pmatrix}$$

这些步骤都使用了有限域算术的性质。结果是 $\mathrm{IMC}(A_{K_1}(N)) = \mathrm{IMC}(K_1) \oplus \mathrm{IMC}(N)$。现将第 1 轮的逆轮密钥定义为 $\mathrm{IMC}(K_1)$，将逆密钥加运算 IA_{K_1} 定义为逆轮密钥与状态向量的逐位异或。于是有 $\mathrm{IMC}(A_{K_1}(N)) = \mathrm{IA}_{K_1}(\mathrm{IMC}(N))$。最后，可以写出

加密： $A_{K_2} \circ \mathrm{SR} \circ \mathrm{NS} \circ A_{K_1} \circ \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{NS} \circ A_{K_0}$

解密： $A_{K_0} \circ \mathrm{INS} \circ \mathrm{ISR} \circ \mathrm{IMC} \circ A_{K_1} \circ \mathrm{INS} \circ \mathrm{ISR} \circ A_{K_2}$

解密： $A_{K_0} \circ \mathrm{INS} \circ \mathrm{ISR} \circ A_{\mathrm{IMC}(K_1)} \circ \mathrm{IMC} \circ \mathrm{ISR} \circ \mathrm{INS} \circ A_{K_2}$

现在，加密和解密采用同一个序列。注意，若加密算法的第 2 轮包括 MC 函数，则该求导运算效率较低。因此，我们有

加密： $A_{K_2} \circ \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{NS} \circ A_{K_1} \circ \mathrm{MC} \circ \mathrm{SR} \circ \mathrm{NS} \circ A_{K_0}$

解密： $A_{K_0} \circ \mathrm{INS} \circ \mathrm{ISR} \circ \mathrm{IMC} \circ A_{K_1} \circ \mathrm{INS} \circ \mathrm{ISR} \circ \mathrm{IMC} \circ A_{K_2}$

不存在通过交换解密算法中的成对运算来得到与加密算法同样的结构的方法。

附件 D.1　GF(2^4)上的算术

表 D.2 中显示了模 x^4+x+1 的 GF(2^4)上的加法和乘法。例如，考虑积$(4 \cdot C)=(0100 \cdot 1100)$。若使用多项式运算描述，则它是积$[x^2 \times (x^3+x^2)] \bmod (x^4+x+1)=(x^5+x^4) \bmod (x^4+x+1)$。由于模运算符右边多项式的阶数大于等于模数的阶数，因此需要用除法来确定余数：

$$
\begin{array}{r}
x+1 \\
x^4+x+1 \overline{)\,x^5+x^4} \\
\underline{x^5+\quad\quad+x^2+x} \\
\quad x^4+\quad+x^2+x \\
\underline{x^4+\quad\quad+x+1} \\
\quad\quad\quad x^2+\quad 1
\end{array}
$$

余数用二进制数表示为 0101，用十六进制数表示为 5。因此$(4 \cdot C)=5$，这与表 D.2 的乘法表一致。

表 D.2　模 x^4+x+1 的 GF(2^4)上的算术

(a) 加法

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

(b) 乘法

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

附件 D.2 列混淆函数

列混淆函数对每列进行运算。一列中的每个半字节被映射为一个新值，这个新值是该列中两个半字节的函数。变换定义为如下关于状态的矩阵相乘（见图 D.4）：

$$\begin{bmatrix} 1 & 4 \\ 4 & 1 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} \\ s_{1,0} & s_{1,1} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} \\ s'_{1,0} & s'_{1,1} \end{bmatrix}$$

使用多项式可将上式描述如下。值 1 对应于多项式 1，值 4（二进制数 100）对应于多项式 x^2。因此有

$$\begin{bmatrix} 1 & x^2 \\ x^2 & 1 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} \\ s_{1,0} & s_{1,1} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} \\ s'_{1,0} & s'_{1,1} \end{bmatrix}$$

记住，乘法也是通过模 x^4+x+1 执行的。使用多项式公式可简单地说明算术运算。参考图 D.2(a) 中的状态矩阵表示，可将列混淆乘法表示为

$$\begin{bmatrix} 1 & x^2 \\ x^2 & 1 \end{bmatrix} \begin{bmatrix} b_0x^3+b_1x^2+b_2x+b_3 & b_8x^3+b_9x^2+b_{10}x+b_{11} \\ b_4x^3+b_5x^2+b_6x+b_7 & b_{12}x^3+b_{13}x^2+b_{14}x+b_{15} \end{bmatrix}$$

左侧矩阵的第一行和右侧矩阵的第一列相乘，得到目标矩阵的左上角元素，即多项式值 $s'_{0,0}$。因此有

$$\begin{aligned} s'_{0,0} &= (b_0x^3 + b_1x^2 + b_2x + b_3) + (x^2)\,(b_4x^3 + b_5x^2 + b_6x + b_7) \\ &= b_4x^5 + b_5x^4 + (b_0 \oplus b_6)x^3 + (b_1 \oplus b_7)x^2 + b_2x + b_3 \end{aligned}$$

容易证明

$$x^5 \bmod (x^4+x+1) = (x^2+x), \quad x^4 \bmod (x^4+x+1) = (x+1)$$

读者可以用多项式除法来验证这些公式。运用这些结果，有

$$\begin{aligned} s'_{0,0} &= b_4(x^2+x) + b_5(x+1) + (b_0 \oplus b_6)x^3 + (b_1 \oplus b_7)x^2 + b_2x + b_3 \\ &= (b_0 \oplus b_6)x^3 + (b_1 \oplus b_4 \oplus b_7)x^2 + (b_2 \oplus b_4 \oplus b_5)x + (b_3 \oplus b_5) \end{aligned}$$

用比特表示 $s'_{0,0}$ 时，$s'_{0,0}$ 的 4 个比特是

$$s'_{0,0} = [(b_0 \oplus b_6), (b_1 \oplus b_4 \oplus b_7), (b_2 \oplus b_4 \oplus b_5), (b_3 \oplus b_5)]$$

类似地，可得

$$\begin{aligned} s'_{1,0} &= [(b_2 \oplus b_4), (b_0 \oplus b_3 \oplus b_5), (b_0 \oplus b_1 \oplus b_6), (b_1 \oplus b_7)] \\ s'_{0,1} &= [(b_8 \oplus b_{14}), (b_9 \oplus b_{12} \oplus b_{15}), (b_{10} \oplus b_{12} \oplus b_{13}), (b_{11} \oplus b_{13})] \\ s'_{1,1} &= [(b_{10} \oplus b_{12}), (b_8 \oplus b_{11} \oplus b_{13}), (b_8 \oplus b_9 \oplus b_{14}), (b_9 \oplus b_{15})] \end{aligned}$$