AES 密码分析 -- Lightless

AES 的全称为 Advanced Encryption Standard，又叫 Rijndael 加密法（并不完全一样）。AES 其实可以算是在 DES 被攻破之后的替代品，当时 AES 的设计要求主要有四点，分别是要比 DES 更安全，在软件和硬件上都要高效，分组大小为 128bit，支持三种密码长度（128bit，192bit，256bit）。AES 完全是面向字节的设计，可以使得他在各个平台有良好的效果。与 DES 不同的是，AES 并没有采用 Feistel 架构，而是使用的 SPN（Substitution-Permutation Network，即代换-置换网络）架构。

AES 的数学基础

AES 是基于字节的设计，在加解密的过程中会涉及到一些数学的知识例如有限域（又叫伽罗瓦域）。不得不提及一些数学的东西，但是好像如果不太理解的话对 AES 的理解也没有太大的障碍。（假定读者已经掌握了群，域，素域的概念，并且知道单位元和逆元的概念）

因为 AES 是基于字节设计的，所以所有的运算都可以看作在有限域 GF(2)上进行的。GF(2)中的所有元素为所有系数在二元域 GF(2)中，并且次数小于 8 的多项式。AES 中有一些运算是按 4 个字节（32bits）定义的，一个 4 字节的元素可以看成系数在 $G(2^8)$中且次数小于 4 的多项式。

由 $b_7b_6b_5b_4b_3b_2b_1b_0$ 构成的一个字节可以被看作是多项式：

$$b_7x^7+b_6x^6+b_5x^5+b_4x^4+b_3x^3+b_2x^2+b_1x^1+b_0x^0$$

例如 0x57，其二进制为 01010111B，对应的多项式为：

$$x^6+x^4+x^2+x+1$$

下面来看一下在 $GF(2^8)$中的运算，首先是加法运算，两个元素相加，结果应该为一个多项

式，其系数是两个元素中对应系数的模 2 加。模 2 加其实可以看作是异或运算，下面举一个例子来说明。假设有以下多项式：

$$a(x) = DAX^3+FFX^2+03X+1B$$

$$b(x) = ABX^3+DFX^2+2DX+CE$$

下面来计算在 $mod(x^4+1)$情况下 $a(x)+b(x)$的结果。

$$a(x)+b(x) = (DA \text{ xor } AB)X^3+(FF \text{ xor } DF)X^2 + (2D \text{ xor } 03)X + (1B \text{ xor } CE)$$

xor 为异或，这样就可以计算出多项式相加的结果。相比多项式相加，在有限域中的乘法就比较蛋疼了。

有限域$GF(2^8)$中两个元素的乘法为模二元域$GF(2)$上的一个8次不可约多项式的多项式乘法，AES 选取的不可约多项式为 $m(x) = x^8 + x^4 + x^3 + x + 1$

其实乘法的资料网上很多，这里还是再说一遍吧。假设有以下两个多项式：

$$a(x) = a_3x^3 + a_2x^2 + a_1x^1 + a_0x^0$$

$$b(x) = b_3x^3 + b_2x^2 + b_1x^1 + b_0x^0$$

它们的乘积为：

$$c(x) = c_6x^6 + c_5x^5 + c_4x^4 + c_3x^3 + c_2x^2 + c_1x^1 + c_0x^0$$

其中系数分别为

$$c_0 = a_0 \cdot b_0$$

$$c_1 = a_1 \cdot b_0 \text{ xor } a_0 \cdot b_1$$

$$c_2 = a_2 \cdot b_0 \text{ xor } a_1 \cdot b_1 \text{ xor } a_0 \cdot b_2$$

$$c_3 = a_3 \cdot b_0 \text{ xor } a_2 \cdot b_1 \text{ xor } a_1 \cdot b_2 \text{ xor } a_0 \cdot b_3$$

$$c_4 = a_3 \cdot b_1 \text{ xor } a_2 \cdot b_2 \text{ xor } a_1 \cdot b_3$$

$$c_5 = a_3 \cdot b_2 \text{ xor } a_2 \cdot b_3$$

$c_6 = a_3 \cdot b_3$

如此一来，c(x)不再可以表示为一个次数小于 4 的多项式，但是可以通过模一个 4 次多项式可以得到一个次数小于 4 的多项式。AES 中选取的多项式为 $M(X) = X^4 + 1$。在这个条件下，AES 中两个 GF(2^8)上多项式的乘法为模 M(x)乘法。这里用 x 乘来表示（妈蛋我打不出这个符号）

$$d(x) = a(x) \times b(x) = d_3x^3 + d_2x^2 + d_1x^1 + d_0x^0$$

这里的四个系数用下面的方法来确定：

$$\begin{bmatrix} d_0 \\ d_1 \\ d_2 \\ d_3 \end{bmatrix} = \begin{bmatrix} a_0 & a_3 & a_2 & a_1 \\ a_1 & a_0 & a_3 & a_2 \\ a_2 & a_1 & a_0 & a_3 \\ a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} b_0 \\ b_1 \\ b_2 \\ b_3 \end{bmatrix}$$

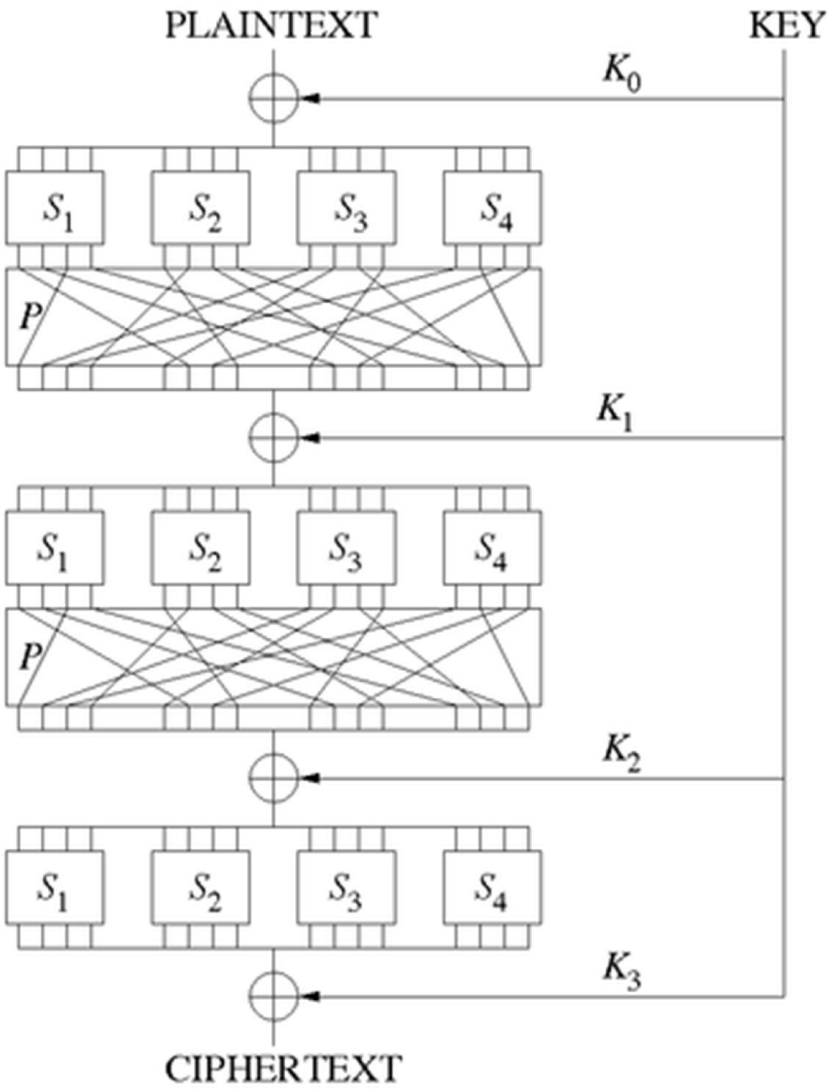
AES 选择了一个有逆元的固定多项式：

$$a(x) = \{03\}x^3 + \{01\}x^2 + \{01\}x + \{02\}$$

$$a^{-1}(x) = \{0b\}x^3 + \{0d\}x^2 + \{09\}x + \{0e\}$$

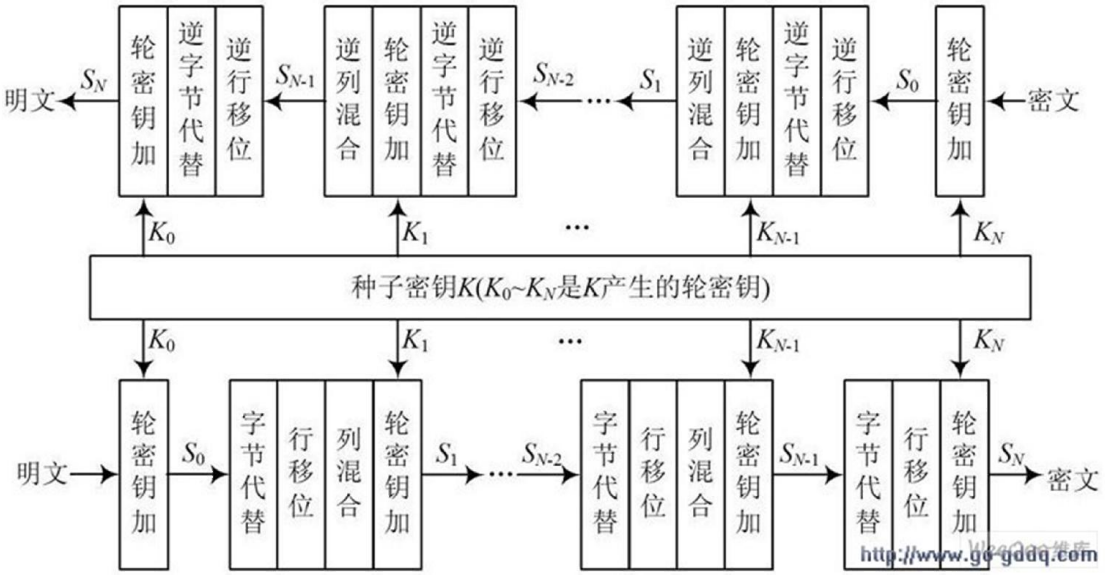
SPN 结构

前面提到了 AES 使用的 SPN 结构，那么 SPN 到底是什么呢？SPN 是一串分组密码中相关的数学运算，这样的一个加密网络使用明文块和密钥作为输入，并通过多轮代换操作和置换操作分别产生密文块。代换(Substitution)和置换(Permutation)分别被称为 S 盒和 P 盒，由于实施于硬件的高效性，所以 SPN 的应用很广泛。



如图为 SPN 的简化结构。主要分为三步：S 盒、P 盒、轮密钥混合（通常为异或）。其实这种思想的主要目的是通过混乱（confusion）和扩散（diffusion）来破坏对密码系统进行的各种统计分析。（很像 TWIST，都是进行破坏）

AES 加密过程



图片是 AES 加解密的主要流程（这图找的不好），下文所提到的子密钥加变换(AddRoundKey)就是图中的轮密钥加，这里说明一下。首先有几个概念要说明一下，AES 的分组长度为 128bit ,**加解密中的每一步的结果称为一个状态** ,当然每个状态也是 128bit。

假设有 $s = s_0 s_1 ... s_{127}$ 这一个状态，那么将其从左到右划分为 16 个字节，分别是：

$$s_{00} s_{10} s_{20} s_{30} s_{01} s_{11} s_{21} s_{31} s_{02} s_{12} s_{22} s_{32} s_{03} s_{13} s_{23} s_{33}$$

把这 16 个字节排成一个二维数组：

$$\begin{bmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{bmatrix}$$

这个数组称之为状态数组，AES 中的各种变换都是基于这个数组来进行的。

AES 的分组长度为 128bit ,密钥有三种长度 ,128bit ,192bit ,256bit ,分别称为 AES-128，AES-192，AES-256。如果密钥以 4 字节（32bits）为单位的话，密钥长度 N_k 可以记为 4,6 或 8。分组长度 N_b=4。迭代的轮数记为 N_r。

AES-128：密钥长度 N_k=4（*32bit） 轮数 N_r=10

AES-192：密钥长度 N_k=6 （*32bit）轮数 N_r=12

AES-256：密钥长度 N_k=8 （*32bit）轮数 N_r=14

好了，现在可以开始关注加密的算法了。对于整个加密过程，其实每次迭代进行了四个操作（最后一次操作除外），分别是字节代替变换（Subbytes），行移位变换（ShiftRows），列混合变换（MixColumns），子密钥加变换（AddRoundKey）

非线性层——Subbytes

通过一个非线性的替换函数，用查找表的方式把每个字节替换成对应的字节，这种方法可以很好的将混淆引入到数据中，即它可以保证对单个状态位的修改可以迅速的传播到整个数据路径中。

它的实质主要有两步，先在有限域 GF(2^8)上求乘法逆，00 映射到 00。然后在 GF(2)上进行一次下面的仿射变换。

$$\begin{bmatrix} y_0 \\ y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

但是这么算太麻烦了，人们制作了一张查找表，就是 S 盒，直接使用查找表来进行替换。

表 4-3 AES 的 S-盒：输入字节(*xy*)对应的十六进制表示的代换值

	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	Fl	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	lA	lB	6E	5A	A0	52	3B	D6	83	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	01	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
x 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	BE	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

上图即为 S 盒。在代码实现的时候通常直接使用 S 盒来进行代换即可。

扩散层——ShiftRows 和 MixColumns

ShiftRows 是行移位变换，目的是增加 AES 的扩散性，对每一个状态的每一行进行不同位移量的循环左移。第 0 行不移位，第 1 行循环左移 1 个字节，第 2 行循环左移 2 个字节，第 3 行循环左移 3 个字节。

$$\begin{bmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{10} & s_{11} & s_{12} & s_{13} \\ s_{20} & s_{21} & s_{22} & s_{23} \\ s_{30} & s_{31} & s_{32} & s_{33} \end{bmatrix} \xrightarrow{\text{ShiftRows}} \begin{bmatrix} s_{00} & s_{01} & s_{02} & s_{03} \\ s_{11} & s_{12} & s_{13} & s_{10} \\ s_{22} & s_{23} & s_{20} & s_{21} \\ s_{33} & s_{30} & s_{31} & s_{32} \end{bmatrix}$$

MixColumns 是一个线性变换，混淆了状态矩阵的每一列，由于每个输入字节都影响了 4 个输出字节，在 AES 中 MixColumns 是最主要的扩散元素。它将一个状态的每一列视为有限域 GF(2^8)上的一个多项式，计算过程稍显繁琐。假设 16 字节的输入状态为 S，经过列混合的输出为 C。S 为刚才经过行移位得到的矩阵。

MixColumns(S) = C

计算方式如下：

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} S_{00} \\ S_{11} \\ S_{22} \\ S_{33} \end{bmatrix}$$

正如前面所说，每一列视为一个多项式，与一个固定的 4x4 矩阵相乘，此矩阵包含常量的项。$C_4C_5C_6C_7$ 的第 2 列是通过四个输入字节 $S_{01}S_{12}S_{23}S_{30}$ 与常量矩阵相乘得到的，以此类推。

子密钥加变化——AddRoundKey

这一层有两个输入，分别是 16 字节的当前状态矩阵和长度为 16 字节的子密钥。这两个输入是通过按位异或操作组合在一起的。这里所说的子密钥需要通过密钥编排来得到，将在下面详细讲解。

密钥编排

密钥编排将输入的原始密钥改造成 AES 使用的子密钥，子密钥的个数等于迭代轮数加 1。在 AES 的输入和输出中都使用了子密钥的 XOR 加法，这个过程有时被称为密钥漂白。对于不同长度的原始密钥，编排方法也有所差异，首先来看 128bit 的原始密钥编排。

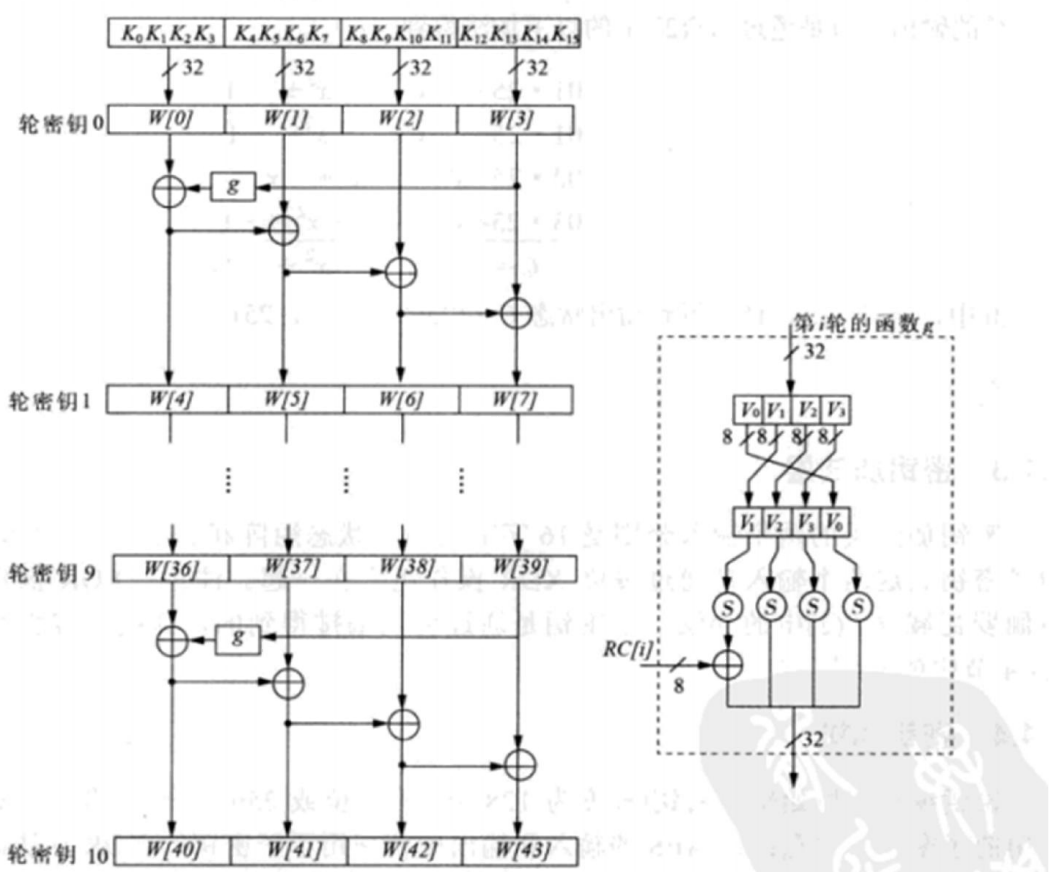


图 4-5　128 位密钥大小的 AES 密钥编排

原谅我又偷了一张图。。。。。

图中的 K_0~K_{15} 为原始密钥对应的字节。

图中的第一个子密钥 k_0 为 AES 的原始密钥。子密钥 W[4i]的计算方式为：

$$W[4i] = W[4(i-1)] + g(W[4i-1])$$

$$W[4i+j] = W[4i+j-1] + W[4(i-1)+j]$$

这里的 g 是一个非线性的函数，目的是增加密钥编排中的非线性和消除 AES 中的对称性。

g 函数首先将输入的 4 个字节翻转，并进行一次按字节的 S-盒替换，最后与轮系数 RC 相加。

轮系数每轮都会改变，且只与 g 最左边的字节相加。轮系数的变换规则如下：

$$RC[1] = x^0 = (0000\ 0001)_2,$$
$$RC[2] = x^1 = (0000\ 0010)_2,$$
$$RC[3] = x^2 = (0000\ 0100)_2,$$
$$\vdots$$
$$RC[10] = x^9 = (0011\ 0110)_2。$$

其他两种 AES-192 和 AES-256 的密钥编排方法与 AES-128 类似，读者可以自行推导。

AES 解密

因为 AES 并不是基于 Feistel 结构的，所以解密的时候要把所有的操作都颠倒过来。字节替代变成逆字节替代，逆向 ShiftRows，逆向 MixColumns，逆向密钥编排。解密的流程图前文已经给出了。具体的方法在此不继续讨论了，有兴趣的读者可以参考《Understanding Cryptography: A Textbook for Students and Practitioners》。AES 的源代码可以在维基百科上的“高级加密标准”词条中的外部连接获取。

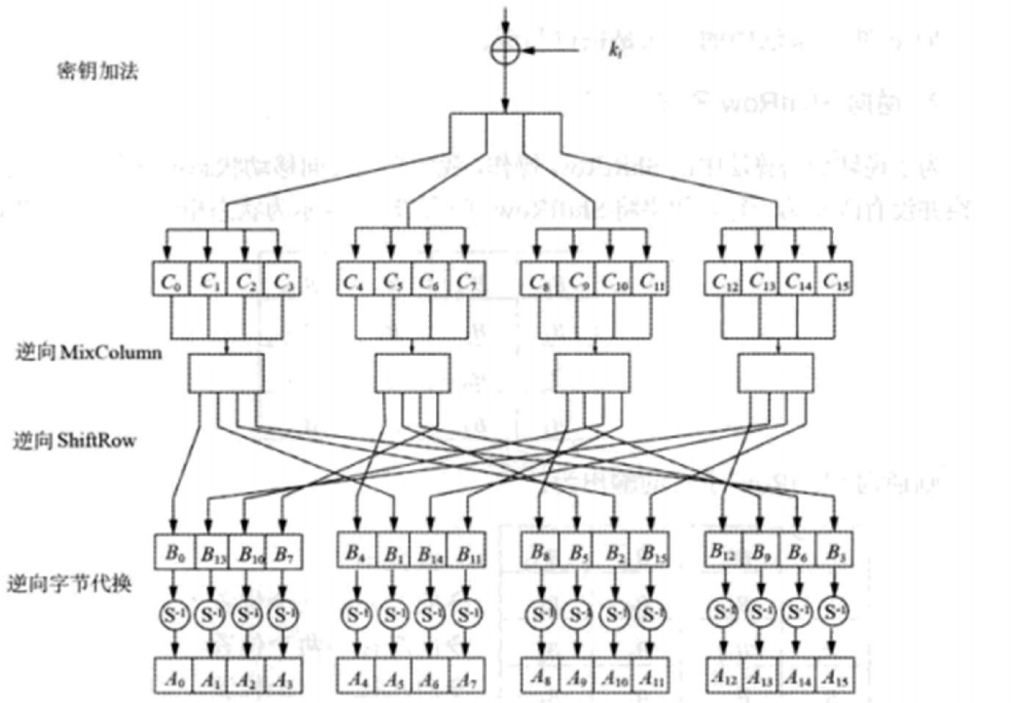


图 4-9　AES 解密轮函数 1,2，…，n_r-1

逆向 MixColumns 层

这里给出解密时需要用到的常量矩阵，使用方法和加密时一样

$$\begin{pmatrix} B_0 \\ B_1 \\ B_2 \\ B_3 \end{pmatrix} = \begin{pmatrix} 0E\ 0B\ 0D\ 09 \\ 09\ 0E\ 0B\ 0D \\ 0D\ 09\ 0E\ 0B \\ 0B\ 0D\ 09\ 0E \end{pmatrix} \begin{pmatrix} C_0 \\ C_1 \\ C_2 \\ C_3 \end{pmatrix}$$

逆向 ShiftRows 层

为了将之前的循环移位恢复回来，第 0 行不用动，因为之前并未对第 0 行进行修改，第 1 行向左移动 3 个位置，第 2 行向左移动 2 个位置，第 3 行向左移动 1 个位置。

逆向字节代换层

表 4-4　逆向 AES S-盒：输入字节(*xy*)对应的替换值的十六进制表示

	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	98	2F	FF	87	34	8E	43	44	C4	DE	E9	CB
2	54	78	94	32	A6	C2	23	3D	EE	4C	95	08	42	FA	C3	4E
3	08	2E	Al	66	28	D9	24	82	76	5B	A2	49	6D	8B	D1	25
4	72	F8	F6	64	86	68	98	16	D4	A4	5C	CC	5D	65	86	92
x 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	88	83	45	06
7	D0	2C	1E	8F	CA	3F	OF	02	Cl	AF	BD	03	01	13	8A	6B
8	3A	91	11	41	4F	67	DC	EA	97	F2	CF	CE	FO	B4	E6	73
9	96	AC	74	22	E7	AD	35	85	E2	F9	37	E8	Lc	75	DF	6E
A	47	Fl	lA	71	ID	29	C5	89	6F	B7	62	0E	AA	18	BE	lB
B	FC	56	3E	4B	C6	D2	79	20	9A	DB	C0	FE	78	CD	5A	F4
C	IF	DD	A8	33	88	07	C7	31	Bl	12	10	59	27	80	EC	5F
x D	60	51	7F	A9	19	85	4A	0D	2D	E5	7A	9F	93	C9	9C	EF
E	A0	E0	38	4D	AE	2A	F5	B0	C8	EB	BB	3C	83	53	99	61
F	17	28	04	7E	BA	77	D6	26	El	69	14	63	55	21	0C	7D

这里给出逆查找表。使用方法和之前一样，不再赘述。

参考文献：

《分组密码的设计与分析》(第二版) 吴文玲 冯登国 张文涛 编著
《信息安全与密码学》徐茂智 游林 编著
《Understanding Cryptography: A Textbook for Students and Practitioners》美 Christof Paar, Jan Pelzl 著
《分组密码的攻击方法与实例分析》李超 孙兵 李瑞林 编著
http://zh.wikipedia.org/wiki/代换-置换网络
http://zh.wikipedia.org/wiki/高级加密标准
http://www.gxu.edu.cn/college/hxhgxy/sec/COURSE/ch04/3_1.htm