三重DES

- 考虑到DES在暴力攻击下的潜在脆弱性，需要寻找替代方案更换DES。

- 一种方法是设计一种全新的算法，其中AES就是一个典型的例子。

- 另一种方法是使用带有DES和多个密钥的多重加密，这将保留现有的软件和设备。例如三重DES（3DES）。

双重DES的隐患

- 每个区块可以使用2个DES加密
  - C=E（K_2，E（K_1，P））
  - P=D（K_1，D（K_2，C））

- 预计破解加密的难度会成倍增加
  - 错误，一般来说，两次加密并不比一次好。[Merkle，Hellman，1981]
  - 只会加倍攻击者的工作量（由于Meet in the Middle Attack）

双重DES的隐患 - Meet in the Middle Attack

- 假定
  - 已获得明文和密文。
  - 对密钥空间2^{56}中的所有可能求明文的加密结果并存储至数组。
  - 检查密钥空间中所有可能密文的解密结果。
- 后果
  - 如果它们在中间相遇，这种攻击方法的复杂度仅略大于2^{57}。
  - 双重DES不安全!

三重DES

- 因此必须使用3重加密
  - 似乎需要3把不同的钥匙
- 但可以使用两个Key和E-D-E序列
  - $C = E_{K1}(D_{K2}(E_{K1}(P)))$
  - $P = D_{K1}(E_{K2}(D_{K1}(C)))$
  - nb加密与解密在安全性方面的等效性
  - 如果K1=K2，则使用单DES
- 使有效密钥长度加倍
  - 112位密钥非常强大
  - 即使是今天的电脑，所有可行的已知攻击都无法攻破
  - $O(2^{112})$ 穷举攻击 / 10^{52}差分密码分析
- 三重DES目前的应用
  - 金融

第6章 AES算法

浙江大學
ZHEJIANG UNIVERSITY

AES算法

- AES: Advanced Encryption Standard
- 1997年NIST(National Institute of Standards and Technology)公开征集新的加密标准以代替DES算法。
- 1998年, NIST从收到的21个算法中挑选出15算法;1999年又从15个算法中挑选了5个算法: MARS, RC6, Rijndael, Serpent, Twofish; 最终Rijndael算法获胜, 成为AES。
- Rijndael算法的作者是Joan Daemen和Vincent Rijmen。
- Rijndael算法与AES算法有一定区别, 前者允许明文块长度可变, 后者不可。
- AES算法的明文长度=128位(16字节), 密文长度=16字节;
- 密钥长度分成三种:
- 128位(16字节), 192位(24字节), 256位(32字节)

AES评估标准

- 初始标准
  - 安全性
  - 成本，计算效率
  - 算法与实现特点
- 最终标准
  - 通用安全性
  - 易于软件和硬件实现
  - 攻击实现
  - 灵活性

AES入围名单

- 经过测试和评估，1999年8月的入围名单如下：
  - MARS（IBM）-复杂、快速、高安全性
  - RC6（USA）-简单，快速，低安全性
  - Rijndael（Belgium）-简洁、快速、安全性较好
  - Serpent（Euro）-缓慢、简洁，高安全性
  - Twofish（USA）-复杂、快速、高安全性
- 进一步的分析和评价
  - 较少却复杂的回合（round） VS 较多却简单的回合
  - 改进现有密码 VS 新idea

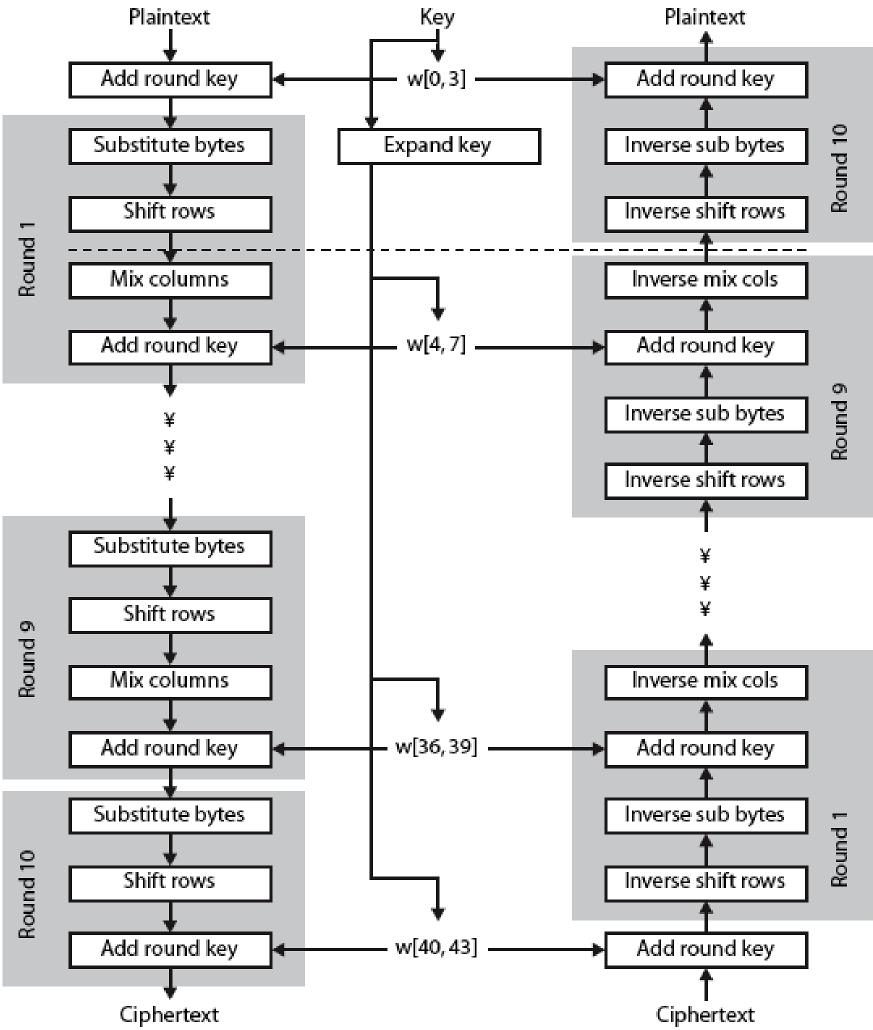
AES密码-Rijndael

- 由比利时的Rijmen Daemen设计，128/192/256位密钥和128位数据
- 一种迭代而非feistel密码
  - 将数据处理为4列4字节的块
  - 在每一轮中对整个数据块进行操作
- 旨在：
  - 抵抗已知的攻击
  - 在各种平台上的速度和代码紧凑性
  - 设计简单

AES算法流程

- AES加密过程是在一个4×4的字节矩阵上运作，这个矩阵又称为“state”，其初值就是一个明文区块（矩阵中一个元素大小就是明文区块中的一个Byte）。
- 加密时，各轮AES加密循环（除最后一轮外）均包含4个步骤：
- 1. AddRoundKey—矩阵中的每一个字节都与该次回合密钥（round key）做XOR运算；每个子密钥由密钥生成方案产生。
- 2. SubBytes—透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。
- 3. ShiftRows—将矩阵中的每个横列进行循环式移位。
- 4. MixColumns—为了充分混合矩阵中各个直行的操作。这个步骤使用线性转换来混合每内联的四个字节。最后一个加密循环中省略MixColumns步骤，而以另一个AddRoundKey取代。

AES算法流程



(a) Encryption
(b) Decryption

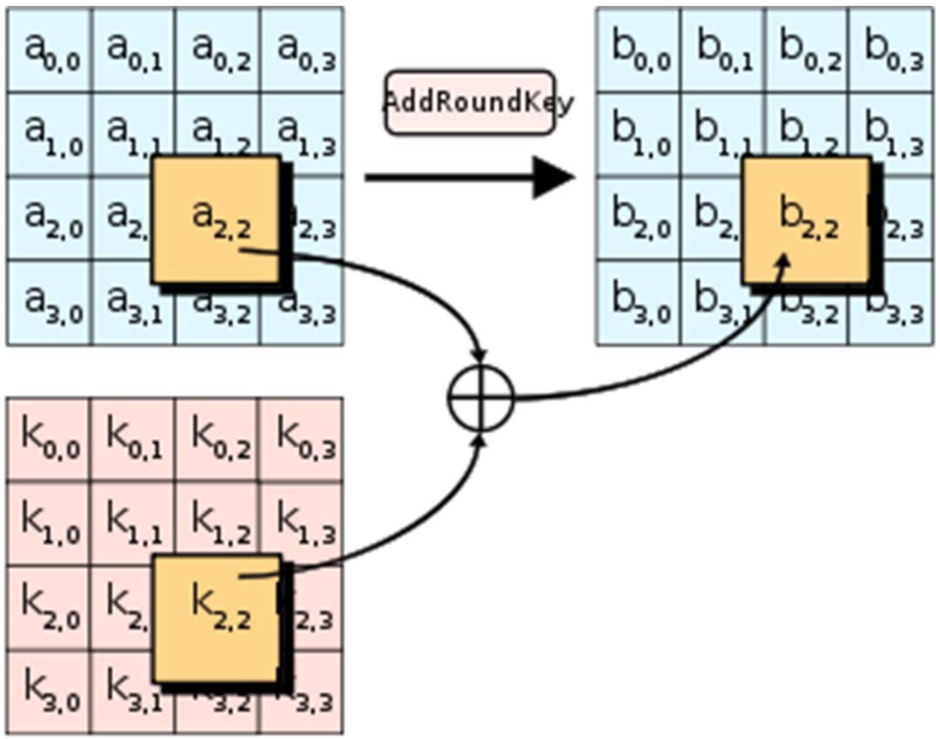
AES算法流程

- 以128位(16字节)密钥为例, 设p为明文, k为密钥, 则AES的加密过程如下:

```
unsigned char a[4] = {0x03, 0x01, 0x01, 0x02};
AddRoundKey(p, k);
for(i=1; i<=10; i++){
  ByteSub(p, 16); /* p[i] = sbox[p[i]]; */
  ShiftRow(p);
  if(i != 10)
    MixColumn(p, a, 1); /* do mul */
  else
    MixColumn(p, a, 0); /* don't mul */
  AddRoundKey(p, k+i*(4*4));
}
```

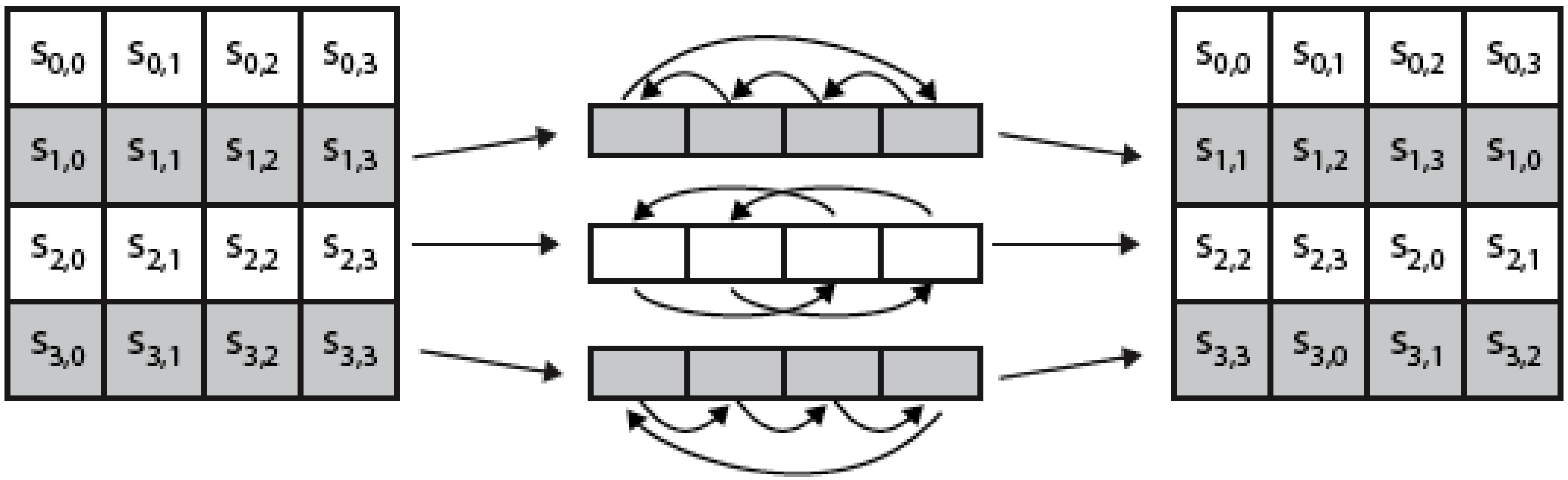
AES算法流程 - AddRoundKey

- 回合密钥将会与原矩阵合并。
- 在每次的加密循环中，都会由主密钥产生一把回合密钥（透过Rijndael密钥生成方案产生），这把密钥大小会跟原矩阵一样，以与原矩阵中每个对应的字节作异或（⊕）加法。



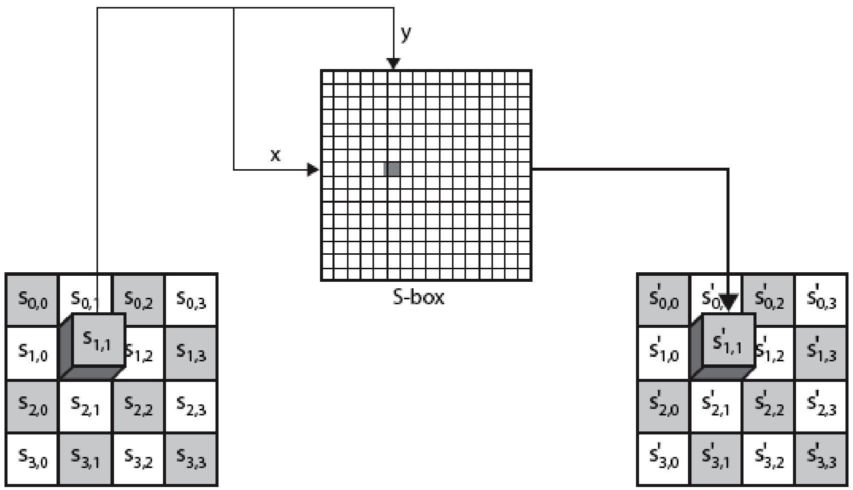
AES算法流程 - ShiftRow

- ShiftRow: 对明文16字节构成的4*4矩阵逐行做循环左移
- 0 1 2 3; 不移动 → 0 1 2 3
- 4 5 6 7; 循环左移1次 → 5 6 7 4
- 8 9 A B; 循环左移2次 → A B 8 9
- C D E F; 循环左移3次 → F C D E



AES算法流程 - ByteSub

- 逐字节的简单替换
- 使用一个16x16字节的表，其中包含所有256个8位值的排列
- 每个字节都由按行（左4位）和列（右4位）索引的字节替换
  - 例如，byte{95}被第9行第5列中的字节替换
- 使用GF（2^8）中定义的值转换构造的S盒
- 旨在抵抗已知的攻击



AES算法流程 – Sbox的生成

- sbox[a] =((a^{-1} * 0x1F) mod (X^8 + 1)) ^ 0x63;
- a^{-1}可以按照扩展欧几里德算法来得到
- a^{-1} * 0x1F mod 0x101 可以用农夫算法得到

AES算法流程 － Sbox的生成

- 例子:
- 查sbox表, 得到sbox[4]=F2, 此值的计算步骤如下:

4^{-1} = CB (AesDemo程序中输入x=4, 点x^{-1} mod 0x11B)

CB * 1F = 91 mod 0x101 (AesDemo中, x=CB, y=1F,点x*y mod 0x101)

91 ^ 63 = 1001 0001

0110 0011 ^)

= 1111 0010 = F2

MixColumn涉及的数学基础 – 有限域

- 域F有时记作{F,+,*}，是有两个运算的集合，这两个运算分别称为加法和乘法;
- 本质上说，域(field)就是一个集合，我们可以在其上做加减乘除运算而不脱离该集合;
- 有限域中元素的个数称为有限域的阶(order);
- 有限域的阶必为素数p的幂，即p^n，其中n为正整数;
- 对任意素数p和正整数n，存在p^n阶的有限域，记为$GF(p^n)$。当n=1时，有限域GF(p)也称素域;

MixColumn涉及的数学基础 – 有限域GF(p)

- 有限域GF(p) 即Galois（伽罗瓦） Field

- 给定一个素数p, 元素个数为p的有限域GF(p)定义为整数{0,1,2,...,p-1}的集合Z_p, 其运算为mod p的算术运算。

- 最简单的有限域是GF(2), 该有限域的元素的个数为2, 它们分别是0和1, 在GF(2)上的加法运算等价于异或运算, 乘法运算等价于二进制与运算。

MixColumn涉及的数学基础 – 有限域GF(2^8)

- (1)对于F[x]中的每个不可约多项式p(x), 可以构造一个域$F[x]_{p(x)}$;
- (2)设p(x)是F[x]中n次不可约多项式, 令$F[x]_{p(x)}$为F[x]中所有次数小于n的多项式集合, 即:
- $F[x]_{p(x)} = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1 x + a_0$
- 其中$a_i \in F$, 即在集合{0,1,2,···,p-1}上取值, 则我们可以定义$F[x]_{p(x)}$上的加法运算及乘法运算如下:
- a(x)+b(x) = (a(x)+b(x)) mod p(x)
- a(x)*b(x) = (a(x)*b(x)) mod p(x)

MixColumn涉及的数学基础 - 有限域GF(2^8)

- (3)在GF(2^n)中, $F[x]_{p(x)}$中所有次数小于n的多项式可以表示为:
- $f(x) = a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_1x + a_0$
- 其中系数a_i是二进制数, 该多项式可以由它的n个二进制系数唯一表示。因此GF(2^n)中的每个多项式都可以表示成一个n位的二进制数;
- (4)AES算法选择用多项式表示有限域GF(2^8)中的元素, 其中不可约多项式p(x)为$x^8+x^4+x^3+x+1$, 该多项式对应的二进制数为1 0001 1011, 对应的十六进制数为0x11B。AES算法的MixColumn步骤中对两个8位数做乘法运算时必须mod 0x11B。
- $x^6 \% (x^4+1) = ((x^4+1)x^2 - x^2) \% (x^4+1) = -x^2$
- $= x^2 \bmod (x^4+1)$

MixColumn涉及的数学基础 －有限域GF(2^8)

- (5)AES算法的MixColumn步骤中还涉及3次多项式的乘法, 多项式乘法运算必须mod (X^4+1), 目的是使得乘积仍旧为一个3次多项式。AES加密及解密分别使用以下两个多项式作为被乘数:
- ① $3x^3 + x^2 + x + 2$
- ② $0x0B*x^3 + 0x0D*x^2 + 0x09*x + 0x0E$

MixColumn中的8位数乘法运算 -用农夫算法计算8位数乘法 mod 0x11B

- 设x=1000 1000, y=0000 0101, p=0, 现要计算x*y mod 0x11B:

```
for(i=0; i<8; i++){
x = x << 1;
y = y >> 1;
若发现y右移后丢失一个1:   y=0000 0101  ->     y = 0000 0010
则做p = p ^ x; 相当于p=p+x, 其中x是左移前的值
若发现x左移后产生进位1:   x=1000 1000  ->     x = 1 0001 0000
则做x = x ^ 0x11B; 相当于x=x-0x11B
即x =          1 0001 0000    ; 被减数是x左移后的值
               1 0001 1011 ^) ; 异或相当于做减法
    =          0 0000 1011
}
```

- 上述过程重复8次, 即x左移8次, y右移8次后,
- p的最终值就是x*y = 0x9E mod 0x11B

MixColumn中的8位数乘法运算

- 8位数乘法实质上是多项式乘法 mod $(x^8+x^4+x^3+x+1)$
- 例如: 求1000 1000 * 0000 0101 mod 0x11B

- 可以把上述两数乘法转化成两个多项式相乘:
- $(x^7 + x^3) * (x^2 + 1) = x^9 + x^7 + x^5 + x^3$

$$= x^7 + x^4 + x^3 + x^2 + x \mod (x^8 + x^4 + x^3 + x + 1)$$

MixColumn中的8位数乘法运算

- 用手工除法求模:

$$
\begin{array}{r|l}
 & x \\
\cline{2-2}
x^8 + x^4 + x^3 + x + 1 & x^9 + x^7 + x^5 + x^3 \\
 & x^9 + x^5 + x^4 + x^2 + x \\
\cline{2-2}
 & x^7 + x^4 + x^3 + x^2 + x
\end{array}
$$

- 把余式转化成二进制就是:
- 1001 1110
- 结论:
- 0x88 * 0x05 = 0x9E mod 0x11B

MixColumn中的8位数乘法运算 -用农夫算法计算

- 用农夫算法分步计算0x05 * 0x43 mod 0x11B

a=0000 0101, b=0100 0011, p=0

a=0000 0101, b=0100 0010, p=0000 0101 （b右移）

a=0000 1010, b=0010 0001, p=0000 0101; ① （a左移）

a=0000 1010, b=0010 0000, p=0000 0101 ^ （b右移）

0000 1010

=0000 1111

MixColumn中的8位数乘法运算 -用农夫算法计算

a=0001 0100, b=0001 0000, p=0000 1111; ②

a=0010 1000, b=0000 1000, p=0000 1111; ③

a=0101 0000, b=0000 0100, p=0000 1111; ④

a=1010 0000, b=0000 0010, p=0000 1111; ⑤

=10100 0000, b=0000 0001, p=0000 1111; ⑥

^10001 1011

=0101 1011, b=0000 0001, p=0000 1111

MixColumn中的8位数乘法运算 -用农夫算法计算

a=0101 1011, b=0000 0001, p=0000 1111

a=0101 1011, b=0000 0000, p=0000 1111 ^

0101 1011

=0101 0100

a=1011 0110, b=0000 0000, p=0101 0100; ⑦

=10110 1100, b=0000 0000, p=0101 0100; ⑧

^10001 1011

=0111 0111

a=0111 0111, b=0000 0000, p=0101 0100

- 结论: 0x05 * 0x43 = 0x54 mod $(x^8 + x^4 + x^3 + x + 1)$

MixColumn中的8位数乘法运算 -用农夫算法计算

- 关于x左移进位后为什么要做x = x ^ 0x11B的进一步说明:
- x*y+p mod 0x11B
- 设x=1000 1000, 在x=x<<1后
- x=1 0001 0000
- 现把x分解成1 0001 1011+0000 1011之和(+其实是异或)
- (1 0001 1011 + 0000 1011)*y+p mod 0x11B
- =$\underline{\text{0x11B*y}}$ + 0x0B*y + p mod 0x11B
- =0x0B*y + p mod 0x11B
- 这里消去划线这一项, 其实就是做了x*y mod 0x11B的除法求余运算。

MixColumn中的3次多项式乘法运算

- char plain[16]={4,3,2,1,11,22,33,44,···}

$(3x^3 + x^2 + x + 2) * (x^3 + 2x^2 + 3x + 4) \bmod (x^4 + 1)$

= 低次系数

$$\begin{pmatrix} 2\ 3\ 1\ 1 \\ 1\ 2\ 3\ 1 \\ 1\ 1\ 2\ 3 \\ 3\ 1\ 1\ 2 \end{pmatrix} * \begin{pmatrix} 4 \\ 3 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 14 \\ 5 \\ 0 \\ 15 \end{pmatrix}$$

高次系数

$x^3(2*1 + 1*2 + 1*3 + 3*4)$

$x^2(2*2 + 1*3 + 1*4 + 3*1)$

$x^1(2*3 + 1*4 + 1*1 + 3*2)$

$x^0(2*4 + 1*1 + 1*2 + 3*3)$

MixColumn中的3次多项式乘法运算

- 注意3次多项式系数的乘法是指8位数乘法mod 0x11B，而加法是指异或，以x^0系数为例:
- 2*4 + 1*1 + 1*2 + 3*3 = 0x08 ^ 0x01 ^ 0x02 ^ 0x05
  = 0x09 ^ 0x02 ^ 0x05 = 0x0B ^ 0x05 = 0x0E
- mod x^4+1的作用是把大于等于4次的项转成小于等于3次:
- 例如: $x^6 \bmod x^4+1$
- $x^6 = (x^4+1)*x^2-x^2 = -x^2 = x^2 \bmod x^4+1$
- 加密时, MixColumn多项式乘法中的被乘数是{3,1,1,2}，而解密时的被乘数是{0x0B, 0x0D, 0x09, 0x0E}，因为:
- $0xB*x^3 + 0xD*x^2 + 9x + 0x0E$ 是 $3x^3 + x^2 + x + 2$的逆。

MixColumn的具体步骤

- void MixColumn(unsigned char *p, unsigned char a[4], int do_mul);
- (1) 对p指向的4*4矩阵m中的4列做乘法运算;
- (2) 这里的乘法是指有限域GF(2^8)多项式模(X^4+1)乘法;
- (3) aes加密时采用的被乘数a为多项式3*X^3 + X^2 + X + 2, 用数组表示为 unsigned char a[4]={0x03, 0x01, 0x01, 0x02};
- (4) aes解密时采用的被乘数a为加密所用多项式的逆, 即{0x0B, 0x0D, 0x09, 0x0E};

$$\begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,0} & s_{0,1} & s_{0,2} & s_{0,3} \\ s_{1,0} & s_{1,1} & s_{1,2} & s_{1,3} \\ s_{2,0} & s_{2,1} & s_{2,2} & s_{2,3} \\ s_{3,0} & s_{3,1} & s_{3,2} & s_{3,3} \end{bmatrix} = \begin{bmatrix} s'_{0,0} & s'_{0,1} & s'_{0,2} & s'_{0,3} \\ s'_{1,0} & s'_{1,1} & s'_{1,2} & s'_{1,3} \\ s'_{2,0} & s'_{2,1} & s'_{2,2} & s'_{2,3} \\ s'_{3,0} & s'_{3,1} & s'_{3,2} & s'_{3,3} \end{bmatrix}$$

MixColumn的具体步骤

(5) 矩阵m中的4列按以下顺序分别与a做乘法运算:

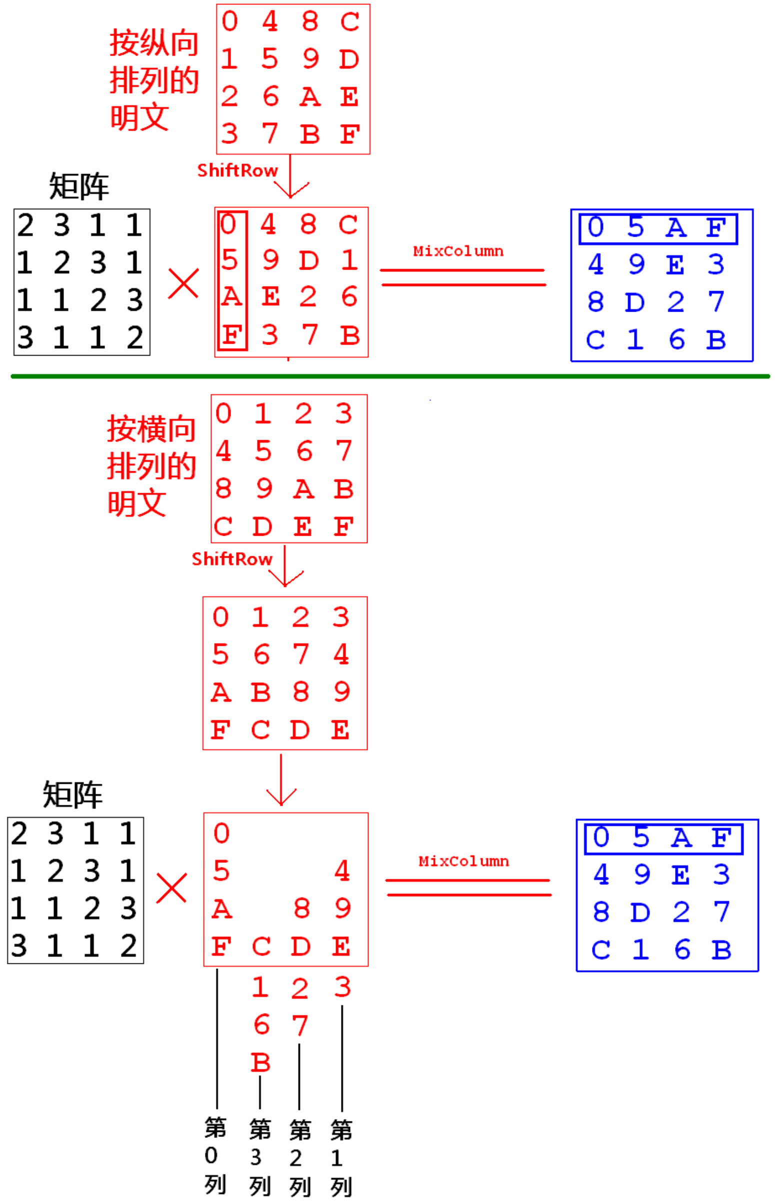
第0列: 由m[0][0], m[1][0], m[2][0], m[3][0]

第3列: 由m[1][3], m[2][3], m[3][3], m[0][3]

第2列: 由m[2][2], m[3][2], m[0][2], m[1][2]

第1列: 由m[3][1], m[0][1], m[1][1], m[2][1]

- 注:明文若按纵向排列, 则与a做乘法的各列顺序为第0、1、2、3列, 且每列不需要向上做循环移位。明文纵向排列与横向排列的对比见图:



MixColumn的具体步骤

- (6) 乘法所得4列转成4行，保存到p中，替换掉p中原有的矩阵；
- (7) do_mul用来控制是否要做乘法运算，加密最后一轮及解密第一轮do_mul=0;