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(28)中且次数小于4的多项式。

由b7b6b5b4b3b2b1b0构成的一个字节可以被看作是多项式：

b7x7+b6x6+b5x5+b4x4+b3x3+b2x2+b1x1+b0x0

例如0x57，其二进制为01010111B，对应的多项式为：

x6+x4+x2+x+1

下面来看一下在GF(28)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) = DAX3+FFX2+03X+1B

b(x) = ABX3+DFX2+2DX+CE

下面来计算在mod(x4+1)情况下a(x)+b(x)的结果。

a(x)+b(x) = (DA xor AB)X3+(FF xor DF)X2 + (2D xor 03)X + (1B 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(28)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) = x8 + x4 + x3 + 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) = a3x3 + a2x2 + a1x1 + a0x0

b(x) = b3x3 + b2x2 + b1x1 + b0x0

它们的乘积为：

c(x) = c6x6 +c5x5 + c4x4 +c3x3 +c2x2 + c1x1 + c0x0

其中系数分别为

c0 = a0 · b0

c1 = a1 · b0 xor a0 · b1

c2 = a2·b0 xor a1·b1 xor a0·b2

c3 = a3·b0 xor a2·b1 xor a1·b2 xor a0·b3

c4 = a3·b1 xor a2·b2 xor a1·b3

c5 = a3·b2 xor a2·b3

c6 = a3·b3

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) = X4 + 1。在这个条件下，AES中两个GF(28)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) x b(x) = d3x3 +d2x2 +d1x1 +d0x0

这里的四个系数用下面的方法来确定：

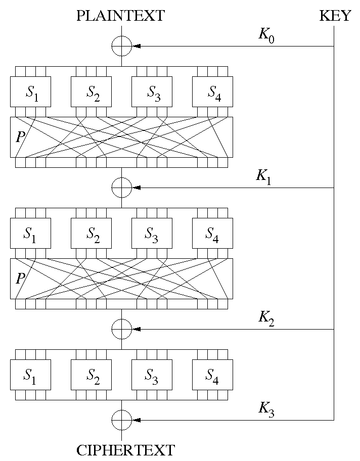
AES选择了一个有逆元的固定多项式：

a(x) = {03}x3 +{01}x2 +{01}x + {02}

a-1(x) = {0b}x3 + {0d}x2 + {09}x + {0e}

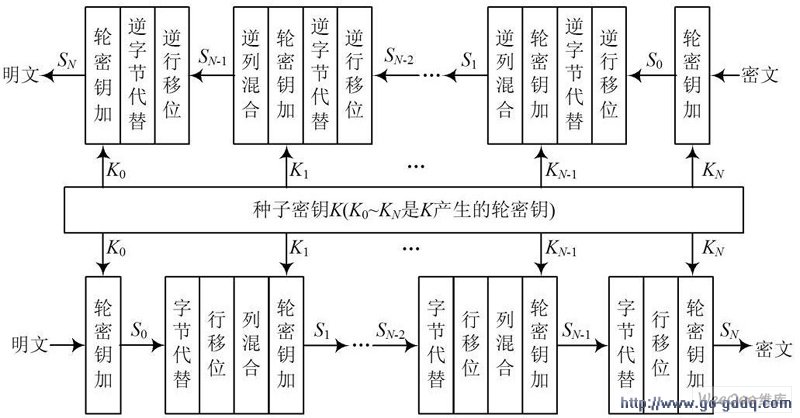
# SPN结构

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 = s0s1…s127这一个状态，那么将其从左到右划分为16个字节，分别是：

s00s10s20s30s01s11s21s31s02s12s22s32s03s13s23s33

把这16个字节排成一个二维数组：

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Nk可以记为4,6或8。分组长度N­b=4。迭代的轮数记为Nr。

AES-128：密钥长度Nk=4（\*32bit） 轮数Nr=10

AES-192：密钥长度Nk=6 （\*32bit）轮数Nr=12

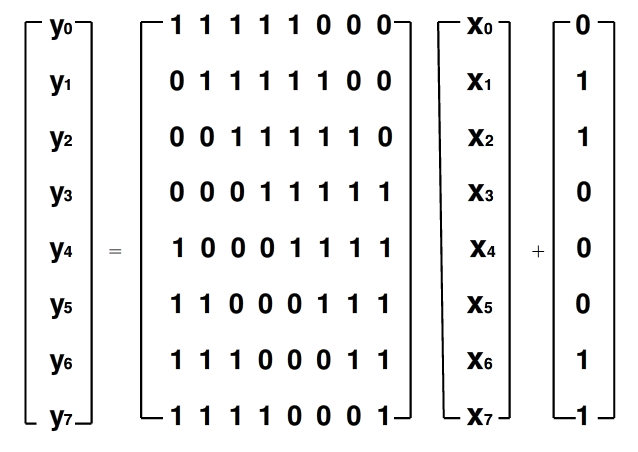
AES-256：密钥长度Nk=8 （\*32bit）轮数Nr=14

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）

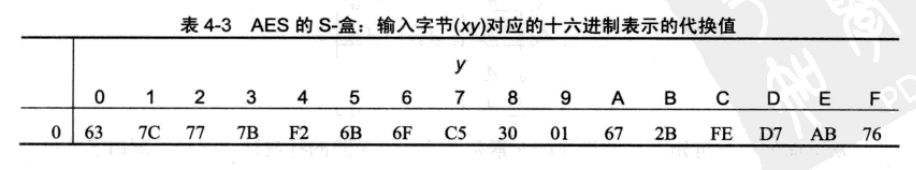
##### 非线性层­——Subbytes

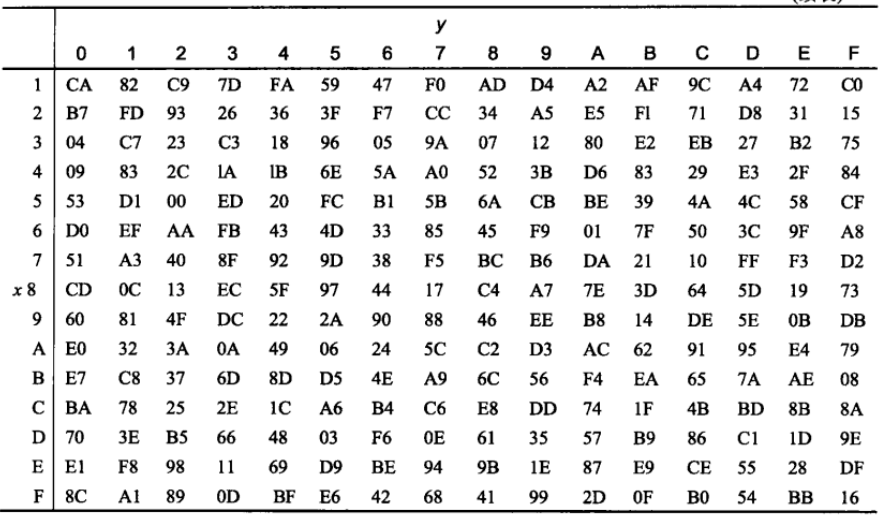
通过一个非线性的替换函数，用查找表的方式把每个字节替换成对应的字节，这种方法可以很好的将混淆引入到数据中，即它可以保证对单个状态位的修改可以迅速的传播到整个数据路径中。

它的实质主要有两步，先在有限域GF(28)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。



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。





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。

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(28)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：

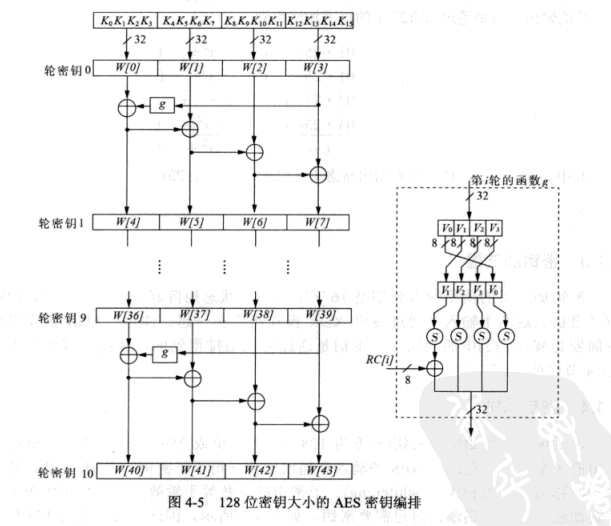
正如前面所说，每一列视为一个多项式，与一个固定的4x4矩阵相乘，此矩阵包含常量的项。C4C5C6C7的第2列是通过四个输入字节S01S12S23S30与常量矩阵相乘得到的，以此类推。

##### 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。



原谅我又偷了一张图。。。。。

图中的K0~K15为原始密钥对应的字节。

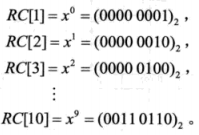
图中的第一个子密钥k0为AES的原始密钥。子密钥W[4i]的计算方式为：

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。

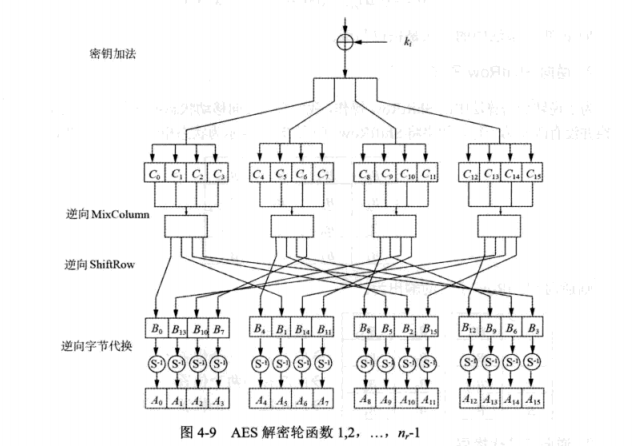
g函数首先将输入的4个字节翻转，并进行一次按字节的S-盒替换，最后与轮系数RC相加。轮系数每轮都会改变，且只与g最左边的字节相加。轮系数的变换规则如下：



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。

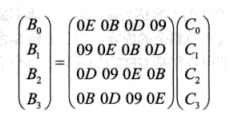
# AES解密

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。



##### 逆向MixColumns层

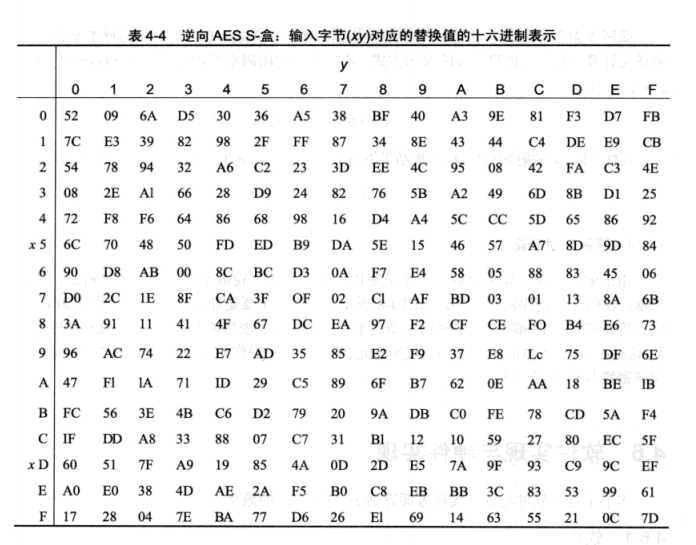
这里给出解密时需要用到的常量矩阵，使用方法和加密时一样



##### 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



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