Advanced Encryption Standard

在实际的工作中，客户端跟服务器交互一般都是字符串格式，所以一个比较好的加密流程是：

- 加密流程 ：明文通过密钥(有时也需要偏移量)，利用 AES 加密算法，然后通过 Base64 转码，最后生成加密后的字符串。
- 解密流程 ： 加密后的字符串通过密钥(有时也需要偏移量)，利用 AES 解密算法，然后通过 Base64 转码，最后生成解密后的字符串。

在 AES 加密中，特别也要注意到字符集的问题。一般用到的字符集是 utf-8 和 GBK，编码方式不一致可能会导致乱码。

AES 的区块长度固定为 128 比特，密钥长度可以是 128，192 或 256 比特，换算成字节长度，密钥必须是 16 个字节，24 个字节，32 个字节，密钥长度更长，破解难度就增大了，所以就更加安全。

AES	密钥长度(bit)	分块长度(bit)	加密轮数
AES-128	128	128	10
AES-192	192	128	12
AES-256	256	128	14

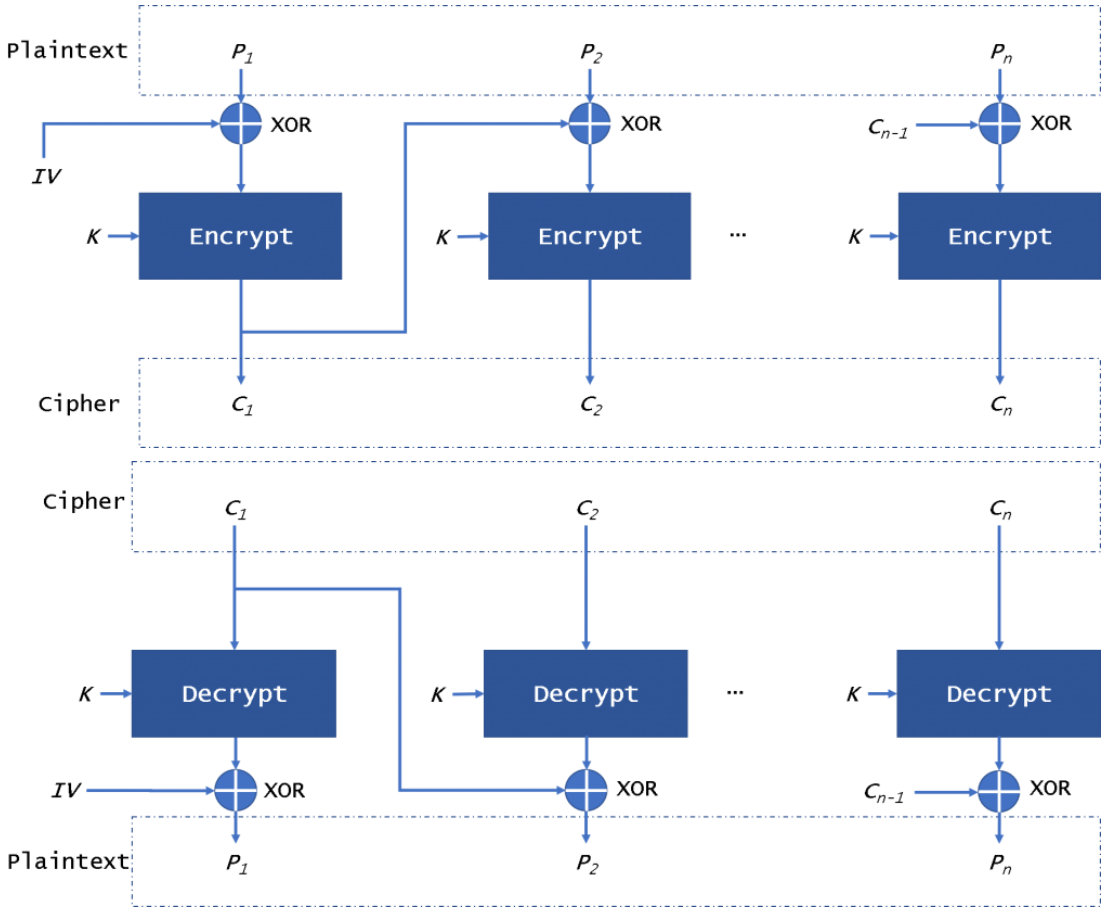
AES 加密需要：明文 + 密钥 + 偏移量（IV） + 密码模式(算法/模式/填充)

AES 解密需要：密文 + 密钥 + 偏移量（IV） + 密码模式(算法/模式/填充)

AES 的加密模式有以下几种：

- 电码本模式(ECB)
- 密码分组链接模式(CBC)
- 计算器模式(CTR)
- 密码反馈模式(CFB)
- 输出反馈模式(OFB)

密码分组链接模式(CBC)：将整段明文切成若干小段，然后每一小段与初始块或者上一段的密文段进行异或运算后，再与密钥进行加密。下图是 CBC 模式加密解密大致流程：



不同加密模式加密解密过程：https://www.highgo.ca/2019/08/08/the-difference-in-five-modes-in-the-aes-encryption-algorithm/

Mode	Formulas	Ciphertext
Electronic Codebook(ECB)	Y_i=F(PlainText$_i$, Key)	Y_i
Cipher Block Chaining(CBC)	Y_i=PlainText$_i$ XOR Ciphertext$_{i-1}$	F(Y , Key);Ciphertext$_0$=IV
Propagating CBC(PCBC)	Y_i=PlainText$_i$ XOR (Ciphertext$_{i-1}$ XOR Text$_{i-1}$)	F(Y , Key);Ciphertext$_0$=IV
Cipher Feedback(CFB)	Y_i=Ciphertext$_{i-1}$	Plaintext XOR F(Y , Key);Ciphertext$_0$=IV
Output Feedback(OFB)	Y_i=F(Y_{i-1} , Key)	Plaintext XOR Y_i
Counter(CTR)	Y_i=F(IV+g(i) , Key);IV=token()	Plaintext XOR Y_i

```
from Crypto import Random
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad


def encrypt(value, key):
    value = value.encode('ascii')
    iv = Random.new().read(AES.block_size)
    cipher = AES.new(key, AES.MODE_CBC, iv)
    ciphertext = cipher.encrypt(pad(bytes(value, encoding='utf8'),
size))
    return iv + ciphertext
```

这是我在国外的网站找到的一个关于 IV 的示例，百度能找到的资料太少了，不过博主好像将 IV 理解错了。

IV 一开始应该是和 block 进行异或运算，然后后面再进行正常的加密操作。

下面来详细介绍 IV：

IV: initialization vector 初始向量，又叫偏移量

是一个固定长度的输入值。一般的使用上会要求它是随机数或伪随机数（pseudorandom）。使用随机数产生的初始向量才能达到语义安全（消息验证码也可能用到初始向量），并让攻击者难以对原文一致且使用同一把密钥生成的密文进行破解。在区块加密中，使用了初始向量的加密模式被称为区块加密模式。

有些密码运算只要求初始向量不要重复，并只要求它用是内部求出的随机数值（这类随机数实际上不够乱）。在这类应用下，初始向量通常被称为 nonce（临时使用的数值），是可控制的（stateful）而不是随机数。这种作法是因为初始向量不会被寄送到密文的接收方，而是收发两方透过事前约定的机制自行计算出对应的初始向量（不过，实现上还是经常会把 nonce 送过去以便检查消息的遗漏）。计数器模式中使用序列的方式来做为初始向量，它就是一种可控制之初始向量的加密模式。

应该是在TLS握手阶段约定IV

CBC 模式的作法是对第一块明文投入随机的 IV，IV 通常是随机数而不是 nonce，然后将明文与向量运算的结果加密，加密的结果再作为下一块明文的向量。这种做法的最终目的是要达到语义上的安全，让攻击者无法从密文中获取能助其破译的相关线索，避免遭受选择明文攻击法。

按照我的理解，如果用同一个密钥而不使用 IV，一个 message 加密多次得到的结果都是一样的，这时候就有问题了，如果这个加密序列在密文中出现很多次，那么攻击者就可以利用一些概率算法，根据对应词汇的出现频率，直接猜出内容，相当于有了一个 leverage，直接跨过密码破解出来。如果我们每次都是用不同的 IV 去加密同一段信息，那我们每次都能得到不一样的秘文，攻击者就更难破解。

为什么需要补齐?

数据按照块的大小 128 比特进行分组，如果不能恰好完全分完，最后分剩下那一块不够 128 比特，就需要进行补齐。类似地，网络中也会对长度不够的分组进行补齐。

算法、模式、填充	16 位字节加密后数据长度	不满 16 字节加密后长度
AES/CBC/NoPadding	16	不支持
AES/CBC/PKCS5Padding	32	16
AES/CBC/ISO10126Padding	32	16
AES/CFB/NoPadding	16	原始数据长度
AES/CFB/PKCS5Padding	32	16
AES/CFB/ISO10126Padding	32	16
AES/ECB/NoPadding	16	不支持
AES/ECB/PKCS5Padding	32	16
AES/ECB/ISO10126Padding	32	16
AES/OFB/NoPadding	16	不支持
AES/OFB/PKCS5Padding	32	16
AES/OFB/ISO10126Padding	32	16
AES/PCBC/NoPadding	16	不支持
AES/PCBC/PKCS5Padding	32	16
AES/PCBC/ISO10126Padding	32	16

各轮 AES 加密均包含 4 个步骤：

A. SubBytes 字节代换

字节代替的主要功能是通过 S 盒完成一个字节到另外一个字节的映射。

AES 定义了一个 S 盒和一个逆 S 盒，用于提供密码算法的混淆性。

S 盒

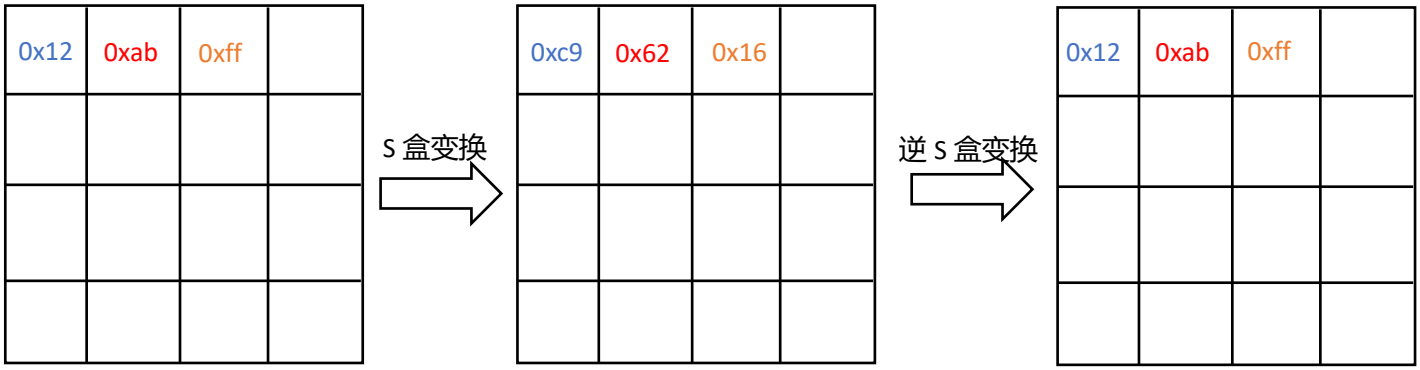
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0x63	0x7c	0x77	0x7b	0xf2	0x6b	0x6f	0xc5	0x30	0x01	0x67	0x2b	0xfe	0xd7	0xab	0x76
1	0xca	0x82	0xc9	0x7d	0xfa	0x59	0x47	0xf0	0xad	0xd4	0xa2	0xaf	0x9c	0xa4	0x72	0xc0
2	0xb7	0xfd	0x93	0x26	0x36	0x3f	0xf7	0xcc	0x34	0xa5	0xe5	0xf1	0x71	0xd8	0x31	0x15
3	0x04	0xc7	0x23	0xc3	0x18	0x96	0x05	0x9a	0x07	0x12	0x80	0xe2	0xeb	0x27	0xb2	0x75
4	0x09	0x83	0x2c	0x1a	0x1b	0x6e	0x5a	0xa0	0x52	0x3b	0xd6	0xb3	0x29	0xe3	0x2f	0x84
5	0x53	0xd1	0x00	0xed	0x20	0xfc	0xb1	0x5b	0x6a	0xcb	0xbe	0x39	0x4a	0x4c	0x58	0xcf
6	0xd0	0xef	0xaa	0xfb	0x43	0x4d	0x33	0x85	0x45	0xf9	0x02	0x7f	0x50	0x3c	0x9f	0xa8
7	0x51	0xa3	0x40	0x8f	0x92	0x9d	0x38	0xf5	0xbc	0xb6	0xda	0x21	0x10	0xff	0xf3	0xd2
8	0xcd	0x0c	0x13	0xec	0x5f	0x97	0x44	0x17	0xc4	0xa7	0x7e	0x3d	0x64	0x5d	0x19	0x73
9	0x60	0x81	0x4f	0xdc	0x22	0x2a	0x90	0x88	0x46	0xee	0xb8	0x14	0xde	0x5e	0x0b	0xdb
A	0xe0	0x32	0x3a	0x0a	0x49	0x06	0x24	0x5c	0xc2	0xd3	0xac	0x62	0x91	0x95	0xe4	0x79
B	0xe7	0xc8	0x37	0x6d	0x8d	0xd5	0x4e	0xa9	0x6c	0x56	0xf4	0xea	0x65	0x7a	0xae	0x08
C	0xba	0x78	0x25	0x2e	0x1c	0xa6	0xb4	0xc6	0xe8	0xdd	0x74	0x1f	0x4b	0xbd	0x8b	0x8a
D	0x70	0x3e	0xb5	0x66	0x48	0x03	0xf6	0x0e	0x61	0x35	0x57	0xb9	0x86	0xc1	0x1d	0x9e
E	0xe1	0xf8	0x98	0x11	0x69	0xd9	0x8e	0x94	0x9b	0x1e	0x87	0xe9	0xce	0x55	0x28	0xdf
F	0x8c	0xa1	0x89	0x0d	0xbf	0xe6	0x42	0x68	0x41	0x99	0x2d	0x0f	0xb0	0x54	0xbb	0x16

逆 S 盒

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0x52	0x09	0x6a	0xd5	0x30	0x36	0xa5	0x38	0xbf	0x40	0xa3	0x9e	0x81	0xf3	0xd7	0xfb
1	0x7c	0xe3	0x39	0x82	0x9b	0x2f	0xff	0x87	0x34	0x8e	0x43	0x44	0xc4	0xde	0xe9	0xcb
2	0x54	0x7b	0x94	0x32	0xa6	0xc2	0x23	0x3d	0xee	0x4c	0x95	0x0b	0x42	0xfa	0xc3	0x4e
3	0x08	0x2e	0xa1	0x66	0x28	0xd9	0x24	0xb2	0x76	0x5b	0xa2	0x49	0x6d	0x8b	0xd1	0x25
4	0x72	0xf8	0xf6	0x64	0x86	0x68	0x98	0x16	0xd4	0xa4	0x5c	0xcc	0x5d	0x65	0xb6	0x92
5	0x6c	0x70	0x48	0x50	0xfd	0xed	0xb9	0xda	0x5e	0x15	0x46	0x57	0xa7	0x8d	0x9d	0x84
6	0x90	0xd8	0xab	0x00	0x8c	0xbc	0xd3	0x0a	0xf7	0xe4	0x58	0x05	0xb8	0xb3	0x45	0x06
7	0xd0	0x2c	0x1e	0x8f	0xca	0x3f	0x0f	0x02	0xc1	0xaf	0xbd	0x03	0x01	0x13	0x8a	0x6b
8	0x3a	0x91	0x11	0x41	0x4f	0x67	0xdc	0xea	0x97	0xf2	0xcf	0xce	0xf0	0xb4	0xe6	0x73
9	0x96	0xac	0x74	0x22	0xe7	0xad	0x35	0x85	0xe2	0xf9	0x37	0xe8	0x1c	0x75	0xdf	0x6e
A	0x47	0xf1	0x1a	0x71	0x1d	0x29	0xc5	0x89	0x6f	0xb7	0x62	0x0e	0xaa	0x18	0xbe	0x1b
B	0xfc	0x56	0x3e	0x4b	0xc6	0xd2	0x79	0x20	0x9a	0xdb	0xc0	0xfe	0x78	0xcd	0x5a	0xf4
C	0x1f	0xdd	0xa8	0x33	0x88	0x07	0xc7	0x31	0xb1	0x12	0x10	0x59	0x27	0x80	0xec	0x5f
D	0x60	0x51	0x7f	0xa9	0x19	0xb5	0x4a	0x0d	0x2d	0xe5	0x7a	0x9f	0x93	0xc9	0x9c	0xef
E	0xa0	0xe0	0x3b	0x4d	0xae	0x2a	0xf5	0xb0	0xc8	0xeb	0xbb	0x3c	0x83	0x53	0x99	0x61
F	0x17	0x2b	0x04	0x7e	0xba	0x77	0xd6	0x26	0xe1	0x69	0x14	0x63	0x55	0x21	0x0c	0x7d

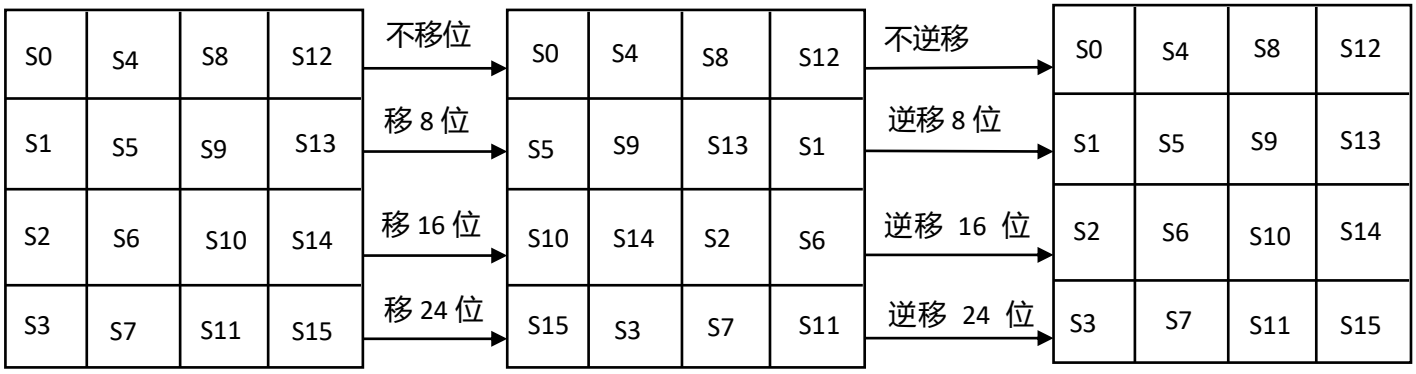
下面来简单演示一下字节变换及逆变换过程

每个字节有 8 位，前四位和后四位分开，4 位能表示的数 0~15，十六进制即为 0~f，相当于一个二维的坐标，代换规则就是取 S 盒和逆 S 盒矩阵中对应位置的字节来替换

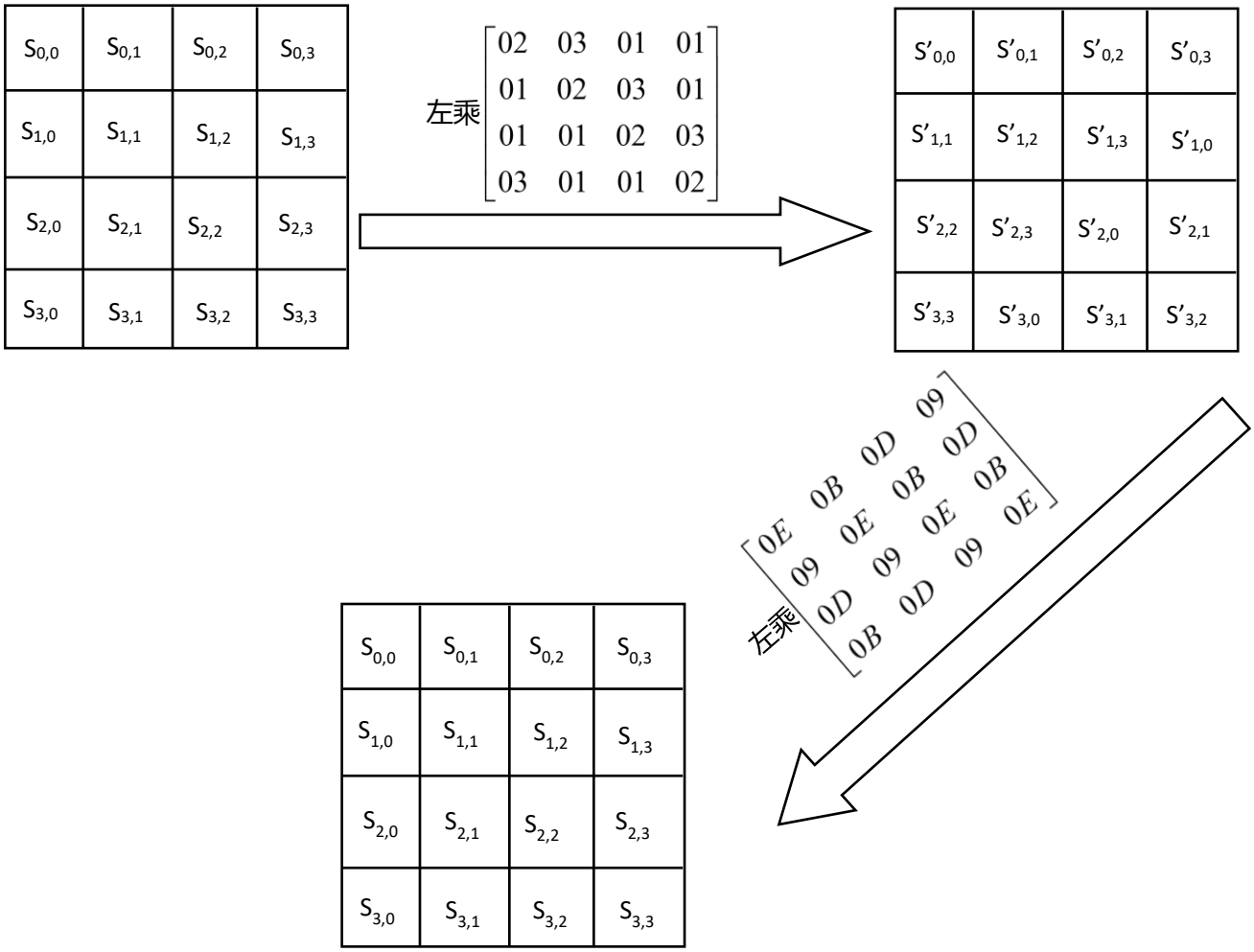


对应颜色在 S 盒和逆 S 盒中有标注

B. ShiftRows 行移位，注意是向左移

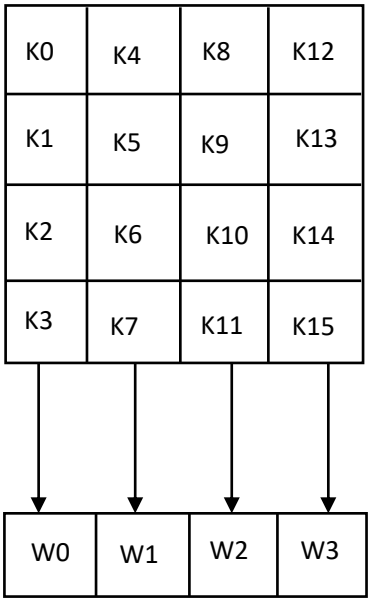


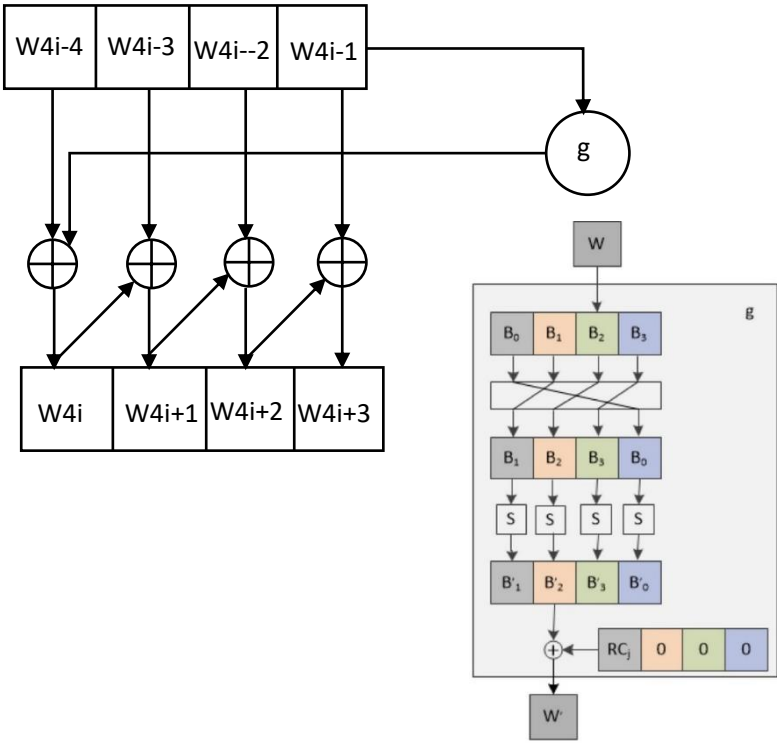
C. MixColumns 列混淆



$$\begin{bmatrix} 0E & 0B & 0D & 09 \\ 09 & 0E & 0B & 0D \\ 0D & 09 & 0E & 0B \\ 0B & 0D & 09 & 0E \end{bmatrix} \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} = \begin{bmatrix} 1 & & & \\ & 1 & & \\ & & 1 & \\ & & & 1 \end{bmatrix}$$

D. AddRoundKey 轮密钥加

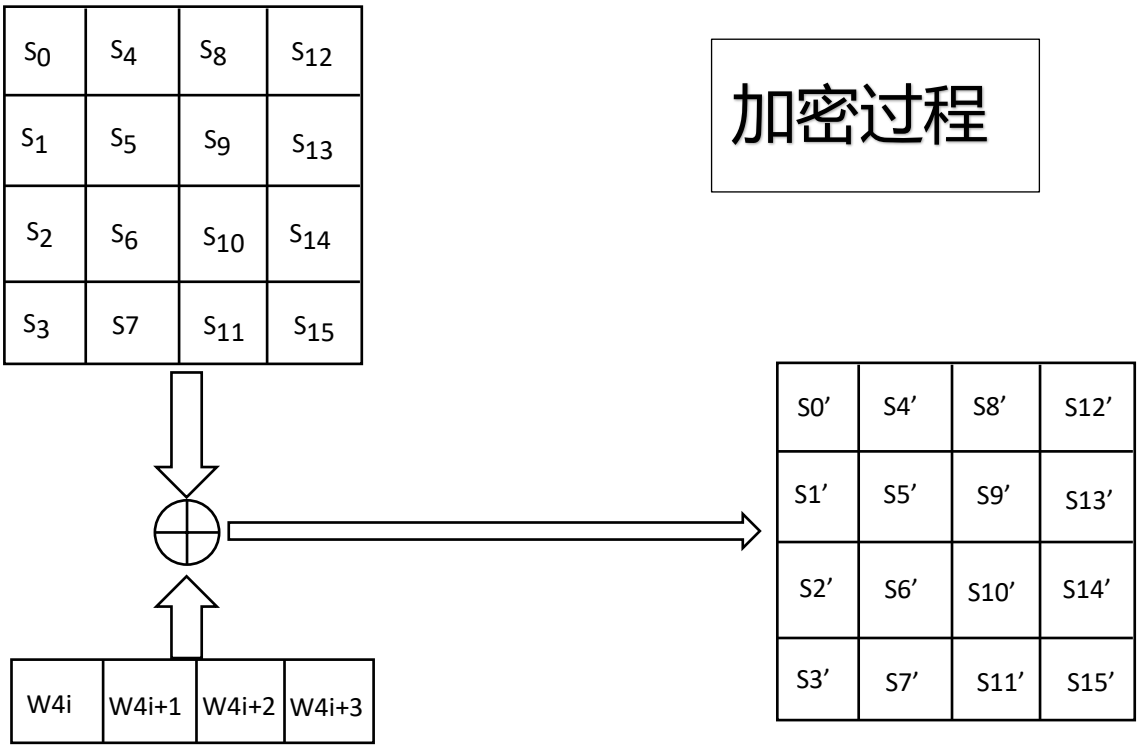




W0 有 32 位，K0+K1+K2+K3

j	1	2	3	4	5
Rcon[j]	01000000	02000000	04000000	08000000	10000000

j	6	7	8	9	10
Rcon[j]	20000000	40000000	80000000	1B000000	36000000



加密过程