实验报告（AES）

【实验目的】

1、 通过本次实验，熟练掌握 AES-128 加解密流程；
2、 了解 AES-192 与 AES-256 加解密流程；
3、 了解 S 盒的生成原理；
4、 通过尝试 AES 的攻击，了解常用的攻击方法；
5、 必做部分需要给出 AES 算法的流程图和伪代码。

【实验环境】

- Python 3.9.7 64-bit

【实验内容】

一、AES 算法的 S 盒生成

1、算法流程

（1）S 盒变换

- 按逐行上升排列的方式初始化 S 盒，第x行y列的字节值为xy；
- 将每个字节映射为有限域$GF(2^8)$上的乘法逆元（注意$0x00$映射到自己）；
- 字节变换（$c=0x63$）：

$$b_i' = b_i \oplus b_{(i+4)\bmod 8} \oplus b_{(i+5)\bmod 8} \oplus b_{(i+6)\bmod 8} \oplus b_{(i+7)\bmod 8} \oplus c_i$$

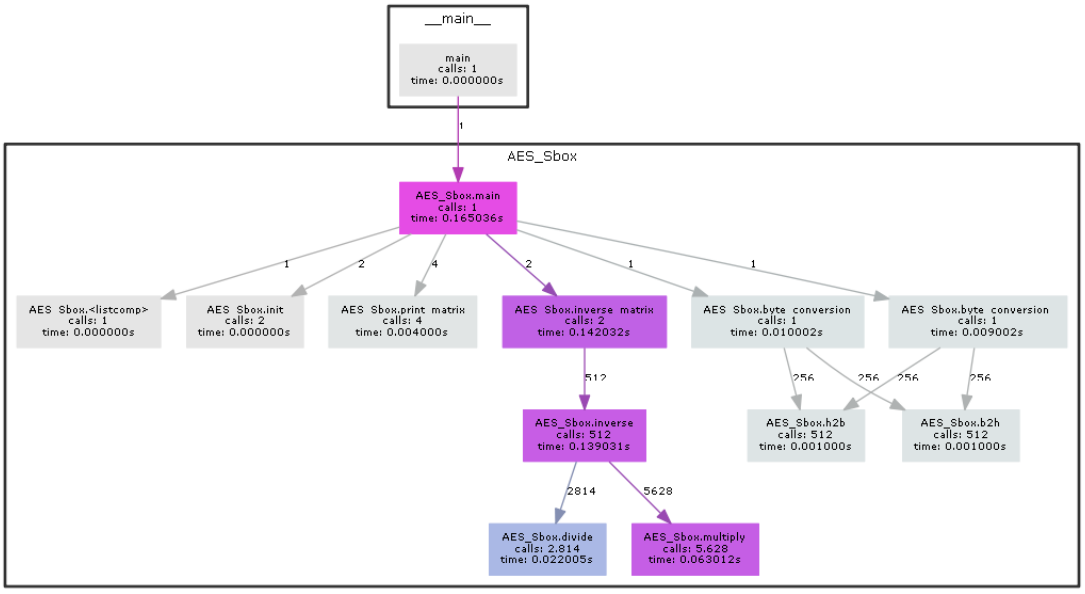
（2）S 盒逆变换

- 按逐行上升排列的方式初始化 S 盒，第x行y列的字节值为xy；
- 字节逆变换（$c=0x05$）：

$$b_i' = b_i \oplus b_{(i+2)\bmod 8} \oplus b_{(i+5)\bmod 8} \oplus b_{(i+7)\bmod 8} \oplus c_i$$

- 将每个字节映射为有限域$GF(2^8)$上的乘法逆元。

函数调用关系图如下：



函数名	函数功能
h2b(),b2h()	十六进制转二进制、二进制转十六进制
multiply(),devide()	$GF(2^8)$上乘、除
inverse()	$GF(2^8)$求逆
init()	Sbox 初始化
byte_conversion()	Sbox 字节变换
byte_conversion_()	Sbox 逆字节变换
inverse_matrix()	Sbox 矩阵逐字节求逆
print_matrix()	Sbox 矩阵打印

2、测试样例及结果截图（S 盒变换 & S 盒逆变换）

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

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

3、讨论与思考

- `h2b(), b2h(), inverse()`调用之前写的，渐成体系。
- 一些细节（如字节逆变换需要仔细想一想）。
- Python 储存是由高位到地位，写索引需要注意。

二、AES 算法

1、算法流程

伪代码：

Algorithm 1： AES Encrypt

Input： plaintext $m[4\times4]$, extend key $w[4\times(Nr+1)]$
Output： ciphertext $c[4\times4]$

1. **function** "AES_enc$(m,\ w)$"
2. **set** $state[4,\ 4]$
3. $state \leftarrow m$
4. AddRoundKey$(state,\ w[0,\ 3])$
5. **for** $round \leftarrow 1$ to $Nr-1$ **do**
6. SubBytes$(state)$
7. ShiftRows$(state)$
8. MixColumns$(state)$
9. AddRoundKey$(state,\ w[4\times round,\ 4\times(round+1)-1])$
10. **end for**
11. SubBytes$(state)$
12. ShiftRows$(state)$
13. AddRoundKey$(state,\ w[4\times Nr,\ 4\times(Nr+1)-1])$

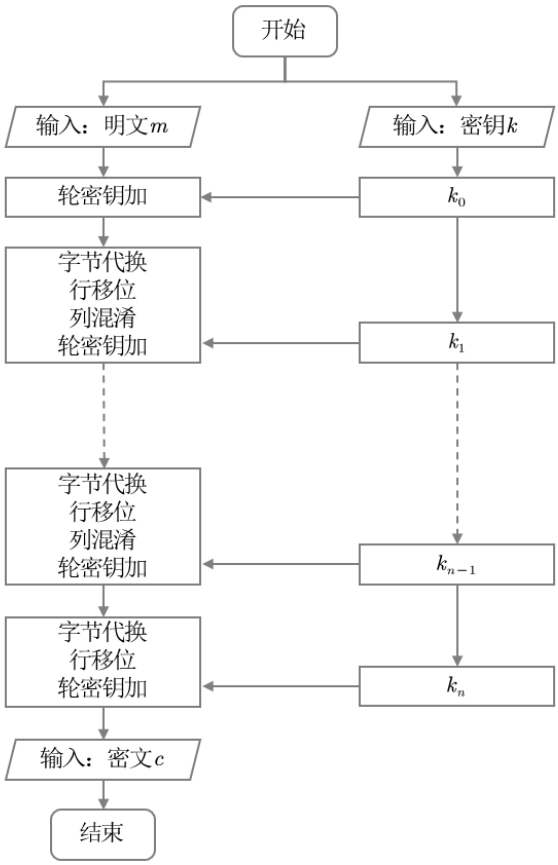
14. $c \leftarrow state$
15. **return** c
16. **end function**

Algorithm 2： AES Decrypt

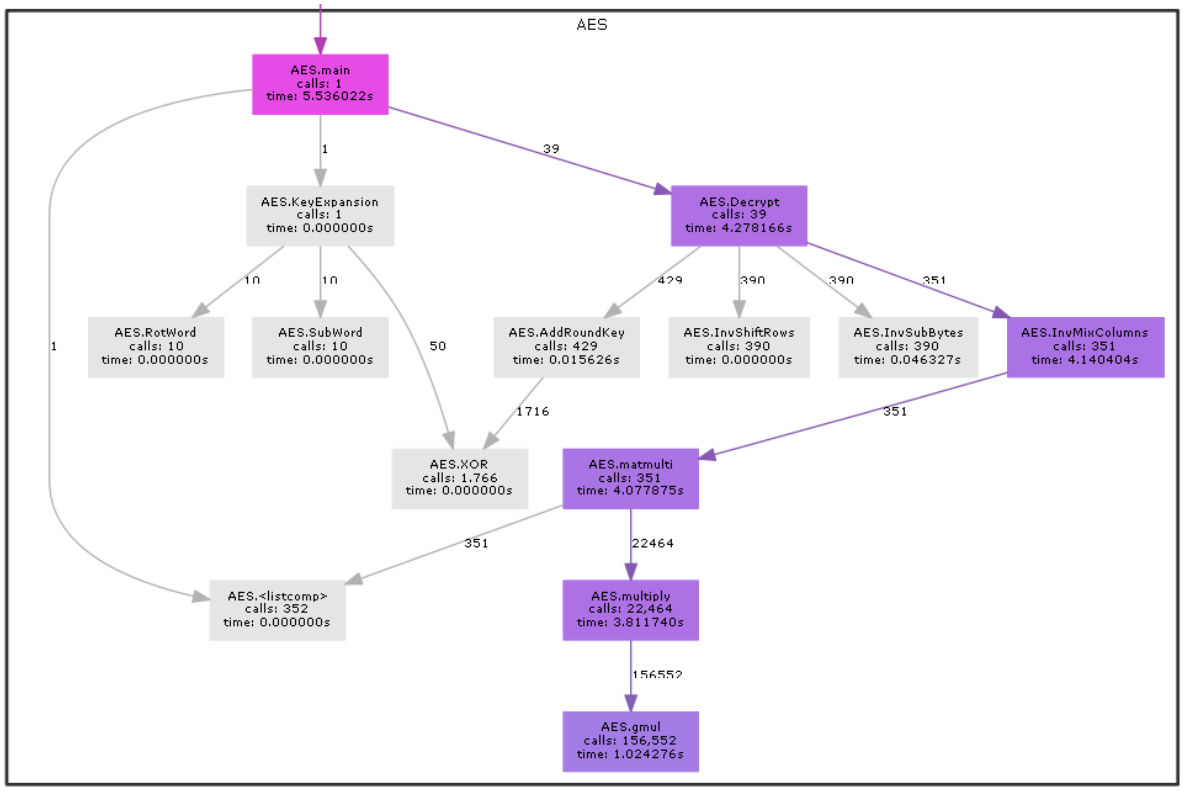
Input： ciphertext $c[4\times4]$, extend key $w[4\times(Nr+1)]$
Output： plaintext $m[4\times4]$

1. **function** "AES_enc$(m,\ w)$"
2. **set** $state[4,\ 4]$
3. $state \leftarrow c$
4. AddRoundKey$(state,\ w[4\times Nr,\ 4\times(Nr+1)-1])$
5. **for** $round \leftarrow Nr-1$ downto 1 **do**
6. InvShiftRows$(state)$
7. InvSubBytes$(state)$
8. AddRoundKey$(state,\ w[4\times round,\ 4\times(round+1)-1])$
9. InvMixColumns$(state)$
10. **end for**
11. InvShiftRows$(state)$
12. InvSubBytes$(state)$
13. AddRoundKey$(state,\ w[0,\ 3])$
14. $c \leftarrow state$
15. **return** m
16. **end function**

流程图：



函数调用关系图如下（以解密为例）：



函数名	函数功能
KeyExpansion()	密钥扩展函数
RotWord()	密钥扩展中的移位操作
SubWord()	字节代替
AddRoundKey()	轮密钥加
InvShiftRows()	逆向行移位
InvSubBytes()	逆向字节代替（对状态矩阵）
InvMixColumns()	逆向列混淆
matmulti()	$GF(2^8)$上的矩阵乘法

2、测试样例及结果截图

```
39
0x0ee863a353b646b8a8d105517804b12b
0xf7dad038fb3dc0ab58bb9987ce4aa0f3
0
0xad4b2698b827a6ffd5115686bf562134

Process finished with exit code 0
```

```
1
0x12dd11bb363e4f010b8f4da1d0e2ad18
0x905cd980666fee1bc97df9c195933cfd
1
0x9b99cb1c2493d9ebd3a3c3270c2b28c4

Process finished with exit code 0
```

```
1
0x1b0403571be0f4e7163bbcfa7f138360
0xea039e44af1b69b93bf85391fa7f221b
0
0x77b15d8dd8b071976372631c60f2c21b

Process finished with exit code 0
```

```
39
0x540ff554865c35abd37411eec81c5c68
0x996c94ae9556e87bd28eb3919a2a89d7
1
0xee5d7e7b9c9432de6628432b54c7907a

Process finished with exit code 0
```

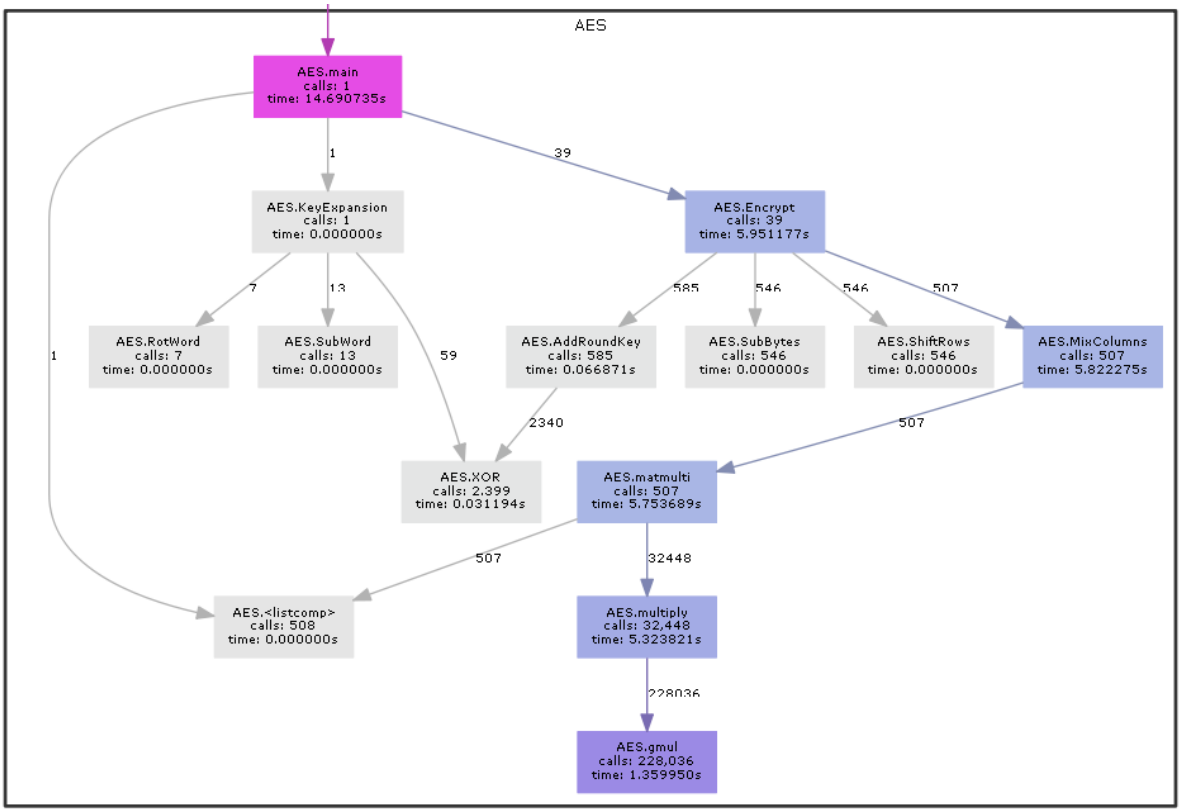
3、讨论与思考

- 该 AES 必做部分代码根据 AES 标准文档中的伪代码进行编写，因此同时实现了 AES-192 和 AES-256，具体内容将在下节讨论。

三、AES-192 与 AES-256（选做一）

1、算法流程

函数调用关系图如下（以 AES-256 加密为例）：



函数名	函数功能
ShiftRows()	行移位
SubBytes()	字节代替（对状态矩阵）
InvMixColumns()	逆向列混淆

2、测试样例及结果截图

```
192
39
0xc142a1e4acb27192f0e36869d2b5b9f8
0xebf353bd5a61d35084c0664197daf0a555d3c9990681171d
0
0xb0620b4b0b35fa6f4aaf1c2e7b52be72

Process finished with exit code 0
```

```
192
1
0x9781be5aa814088ca649afe586bd6c62
0x21d7d2a82a95fa5a0c690cc04a1d7a0c604a2fd7c9fc1231
1
0x476d4bde27714471f7502fa4b16ab470

Process finished with exit code 0
```

```
256
1
0x9d05e548aea37bca11784985f41fbd41
0xf90ecec88de2a2684d8f80affa698080e1d34a9d34bf95489a240906eed38ddf
1
0x6973d1e3929a7c4082a2f558515b850e

Process finished with exit code 0
```

```
256
39
0x2f59099dce0e691e4cc2f6c053eddc28
0x92850bebc6fd62c37d0263db5c24b2bcb6fb8a5351f606acf875bdf1cb145346
0
0xf4e18fb2ea29a1cfc9b2a70ccb1fb6c7

Process finished with exit code 0
```

3、讨论与思考

- AES-192 与 AES-128 除加密轮数不同外其他完全相同；AES-256 在密钥扩展中与 AES-128 不同：当第i个秘钥是 4 的整倍数时需要额外处理一次。
- AES 相比同类对称加密算法速度算是非常快，比如在有 AES-NI 的 x86 服务器至少能达到几百 M/s 的速度。安全性在可预见的未来是基本等同的，因为即使是 128 位也足够复杂无法被暴力破解。现在 112 位密码还在商业应用，而 128 位是 112 位的几万倍，所以在实务中用 128 位比较划算（节约资源）。
- AES-256 比 AES-128 大概需要多花 40%的时间，用于多出的 4 轮 round key 生成以及对应的 SPN 操作；另外，产生 256 bit 的密钥可能也需要比 128 位密钥多些开销。

【收获与建议】

1、收获

- ~~加深了对 Rijndael 加密结构及 AES 算法的理解。~~
- ~~提高了对 Python 的熟练度。~~
- ~~巩固了排版能力。~~
- 收获了劳累与繁忙。

2、建议

- 无。

【思考题】

1、比较 AES 与 DES 的异同，AES 相比于 DES 有哪些改进。

相同：均为对称分组密码，重复使用轮函数；且每一轮由替换、置换和密钥叠加组成。

不同：

AES	DES
以状态矩阵的形式处理数据	分为左右两半处理数据
基于替换置换网络	基于 Feistel 网络
密钥长度可为 128 bit, 192 bit, 256 bit	密钥长度为 56 bit
对应加密轮数可为 10 轮, 12 轮, 14 轮	加密轮数为 16 轮
plaintext 为 128 bit	plaintext 为 64 bit
字节代替、行移位、列混淆、轮密钥加	扩展置换、S 盒、P 盒、异或、交换
整个函数可逆，每轮函数需要可逆	整个函数可逆，每轮函数不需要可逆

改进：

- AES 基于了$GF(2^8)$的强数学结构，从运算的角度增强了安全性。
- AES 提供 128、192、256 三种参数，其密钥长度和加密轮数不同，增强了可选择性。
- AES 每轮处理整段而不是一半的数据，且密钥大小更大，提高了算法的安全性。
- 作为分组密码，DES 加密单位仅为 64 bit，每个分组仅含 8 个字符，这对于数据传输来说太小，且其中某些位还要用于奇偶校验或其他通讯开销，处理速度慢、加密耗时；AES 对内存的需求非常低，运算速度快，在有反馈模式、无反馈模式的软硬件中，基于 Rijndael 结构的 AES 算法提高了系统的性能。

2、（选做）破解不包含 S 盒的 AES（即不做字节代替），并使用伪代码或者代码给出破解流程。

除 S 盒外的运算：行移位、列混淆、轮密钥加均为仿射变换。

行移位和列混淆可用矩阵乘的方式表示(这里的矩阵不是想象中的4×4矩阵这么简单，行移位需要用 16×16 矩阵表示，即将状态矩阵转化为 16×1 的列向量)：

$$A_1=\begin{bmatrix}
01 & & & & & & & & & & & & & & & \\
 & 01 & & & & & & & & & & & & & & \\
 & & 01 & & & & & & & & & & & & & \\
 & & & 01 & & & & & & & & & & & & \\
 & & & & 01 & & & & & & & & & & & \\
 & & & & & 01 & & & & & & & & & & \\
 & & & & & & 01 & & & & & & & & & \\
 & & & & & & & 01 & & & & & & & & \\
 & & & & & & & & 01 & & & & & & & \\
 & & & & & & & & & 01 & & & & & & \\
 & & & & & & & & & & 01 & & & & & \\
 & & & & & & & & & & & 01 & & & & \\
 & & & & & & & & & & & & 01 & & & \\
 & & & & & & & & & & & & & 01 & & \\
 & & & & & & & & & & & & & & 01 & \\
 & & & & & & & & & & & & & & & 01
\end{bmatrix}$$

行移位可以看作一个置换，如 6 变换到 2 的位置，则$A_1[2,6]=$‘01’。

$$\begin{bmatrix} 1 & 5 & 9 & 13 \\ 2 & 6 & 10 & 14 \\ 3 & 7 & 11 & 15 \\ 4 & 8 & 12 & 16 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 5 & 9 & 13 \\ 6 & 10 & 14 & 2 \\ 11 & 15 & 3 & 7 \\ 16 & 4 & 8 & 12 \end{bmatrix}$$

列混淆就直接是一个准单位阵：

$$A_2 = \begin{bmatrix} 02 & 03 & 01 & 01 & & & & & & & & & & & & \\ 01 & 02 & 03 & 01 & & & & & & & & & & & & \\ 01 & 01 & 02 & 03 & & & & & & & & & & & & \\ 03 & 01 & 01 & 02 & & & & & & & & & & & & \\ & & & & 02 & 03 & 01 & 01 & & & & & & & & \\ & & & & 01 & 02 & 03 & 01 & & & & & & & & \\ & & & & 01 & 01 & 02 & 03 & & & & & & & & \\ & & & & 03 & 01 & 01 & 02 & & & & & & & & \\ & & & & & & & & 02 & 03 & 01 & 01 & & & & \\ & & & & & & & & 01 & 02 & 03 & 01 & & & & \\ & & & & & & & & 01 & 01 & 02 & 03 & & & & \\ & & & & & & & & 03 & 01 & 01 & 02 & & & & \\ & & & & & & & & & & & & 02 & 03 & 01 & 01 \\ & & & & & & & & & & & & 01 & 02 & 03 & 01 \\ & & & & & & & & & & & & 01 & 01 & 02 & 03 \\ & & & & & & & & & & & & 03 & 01 & 01 & 02 \end{bmatrix}$$

若把密钥也看作一个 16×1 矩阵B，则加密过程可看作仿射变换f：

$$f(X_{16\times1}) = Y = A_{16\times16} \otimes X_{16\times1} \oplus B_{16\times1},\ A = A_1 \otimes A_2$$

其中$\oplus$, $\otimes$为$GF(2^8)$上的矩阵加法和乘法
这样一来，只需要一组明密文对（X, Y）即可解出密钥矩阵B

$$B_{16\times1} = Y_{16\times1} \oplus A_{16\times16} \otimes X_{16\times1}$$

事实上，这里的B基本上跟 AES 里面的密钥没有联系了，也不用考虑密钥拓展的问题，因为不含字节代替的 AES 在密钥拓展时也没有字节代替，所有的操作都是线性的，我们在解密时只需将矩阵变为逆矩阵即可。

Algorithm 3：AES without Sbox

Input：plaintext & ciphertext pair $(X_{16\times1},\ Y_{16\times1})$
Output：key matrix $B_{16\times1}$

```
function "Fake_AES(X, Y)"
    /* 按上述原理生成A1, A2 */
    set A1[16×16], A2[16×16]
    B = Y ⊕ A ⊗ X
    return B
end function
```

笔者认为这道选做题的意义不大，但也不完全没有意义，至少让大家能够更深刻地理解 AES 算法的细节，也可以感受到 S 盒字节代替在 AES 算法中的重要地位；同时，也让笔者认识到矩阵与线性变换的一些微妙的关系。
因此对于不含 S 盒的 AES，已经不能叫 AES 了，它连最基本的 Rijndael 结构都不满足了，下面是网上仅存的一些对这个“拍脑袋”想出来问题的讨论：

AES with linear operations: AES with no s-box?

Asked 5 years, 5 months ago Modified 5 years, 5 months ago Viewed 615 times

0

I am looking to find an AES algorithm/implementation that is linear. I know that s-box is the only part in the AES method that has non-linear operations. So is there any algorithm/implementation that does not use S-box or any other way to make it linear? I am aware that this would be less secure, but that is okay for this purpose.

algorithm encryption aes

Share Improve this question Follow

asked Oct 27, 2016 at 19:39
smjee
67 ● 2 ● 5

1 This is off-topic for Stack Overflow. The right forum is crypto.stackexchange.com. That said, if you removed the S-box, it wouldn't be Rijndael any more. It'd be something else, and so not AES. Do you really mean "can I make a cipher that does not require an S-box?" Or do you really mean AES specifically? – Rob Napier Oct 27, 2016 at 19:44

Okay, so if I understand correctly.. removing the s-box affects the NIST rules? If so, I would like to have a cipher without s-box. – smjee Oct 27, 2016 at 19:50

1 I don't know why you would call this "NIST rules." AES is a specific algorithm (a specific set of configurations of Rijndael). If you change the algorithm, then it's not AES. But sure, there are plenty of ciphers without an S-Box: en.wikipedia.org/wiki/Caesar_cipher (that's a joke, please don't use the Caesar Cipher). You should expand your question to the actual problem you're trying to solve (why linear? what security guarantees do you require?) and post it to crypto.stackexchange.com. – Rob Napier Oct 27, 2016 at 20:07

AES with linear S-Box

Asked 3 years, 8 months ago Modified 3 years, 8 months ago Viewed 617 times

2

If the AES S-Box is replaced with a linear or affine transformation, for instance the identity mapping $\sigma(x) = x$, does the cipher become entirely affine and hence trivially weak?

aes s-boxes affine-cipher

Share Improve this question Follow

asked Aug 10, 2018 at 11:47
conchild
685 ● 5 ● 17

Add a comment

1 Answer Sorted by: Highest score (default)

6

> does the cipher become entirely affine and hence trivially weak?

Yes: the AES sbox is the only source of nonlinearity (the ShiftRows and MixColumns are linear, and AddRoundKeys for a fixed key is affine), and so if you replace it with a linear/affine one, the entire cipher becomes affine.

Share Improve this answer Follow

answered Aug 10, 2018 at 14:54
poncho
130k ● 10 ● 208 ● 334