实验报告

【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】

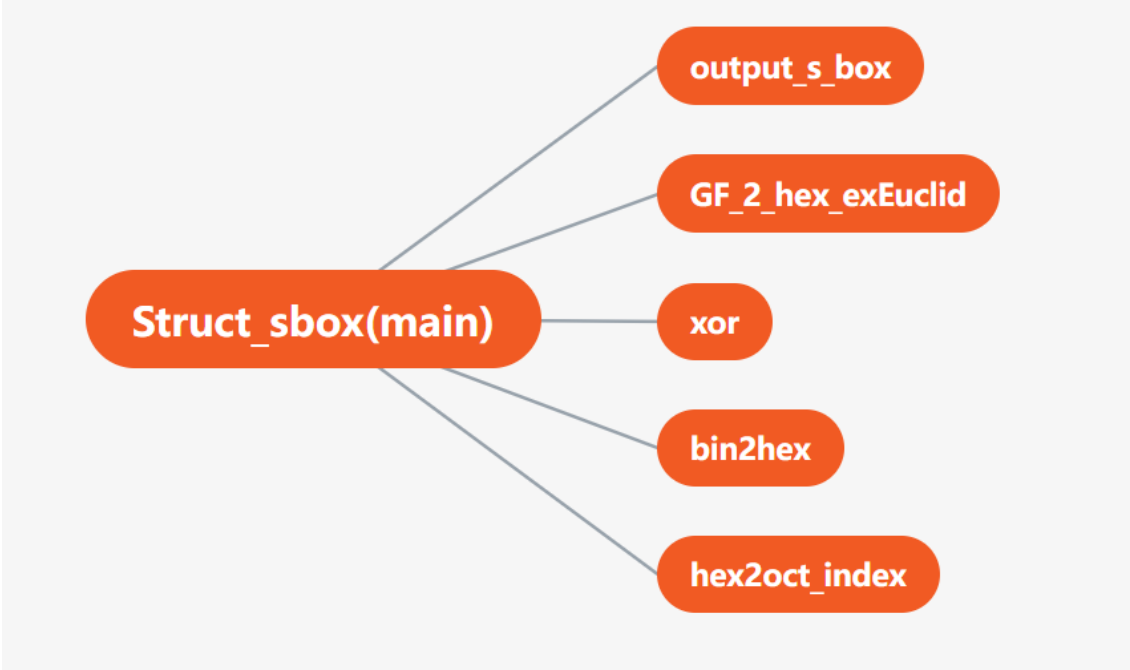
1. 语言：C
2. 平台：clion 2021.2 版本

【实验内容】

一、AES算法的S盒生成

1. **算法流程**

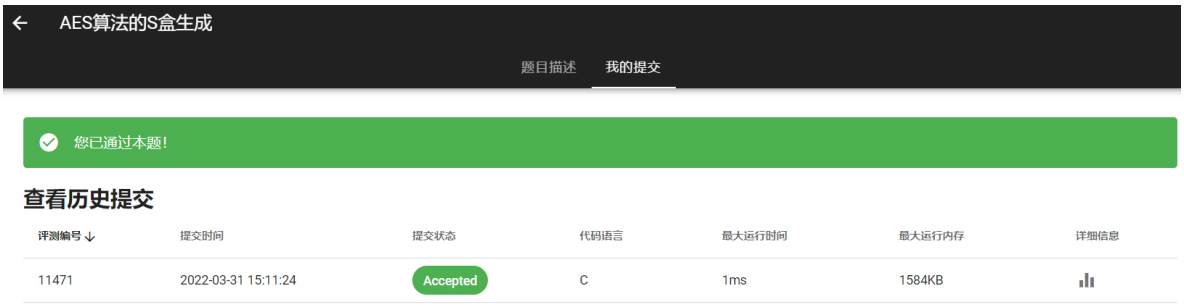
   - **函数调用图如下：**



2. **测试样例及结果截图：**

   本地测试样例的运行结果如下所示：

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



3. **讨论与思考：**

   这道题其实不难，按着题干中给的生成流程做就可以了。

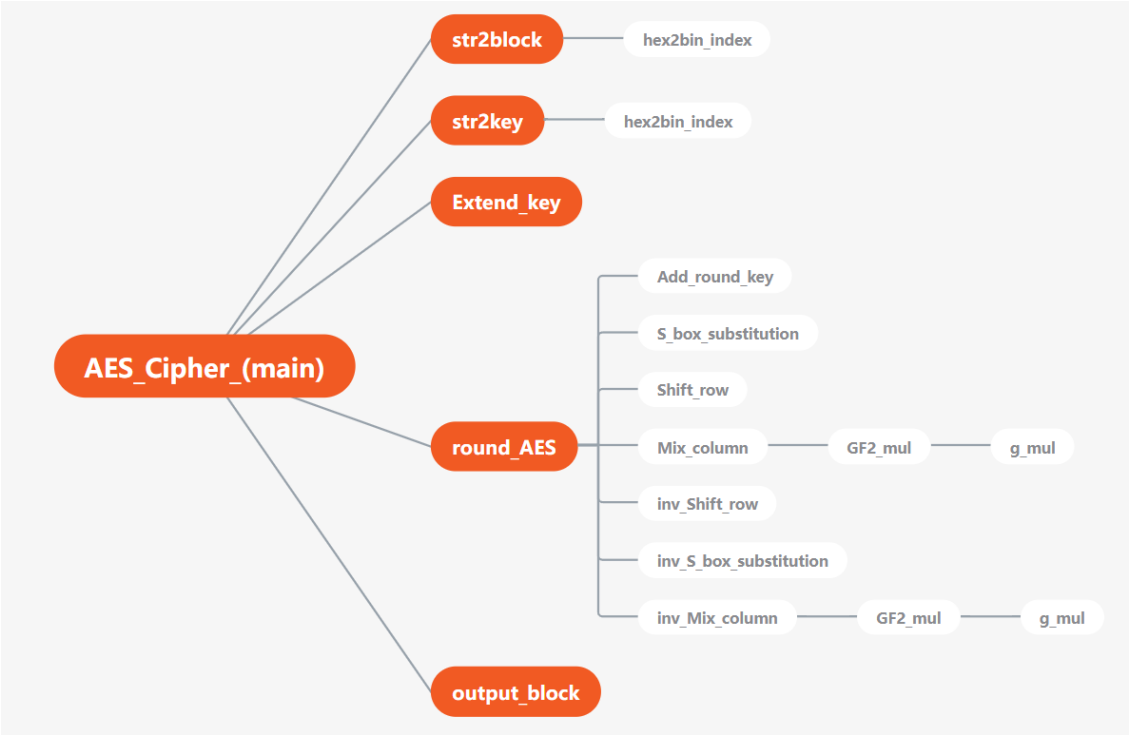
二、AES算法

三、AES-192与AES-256（选做一）

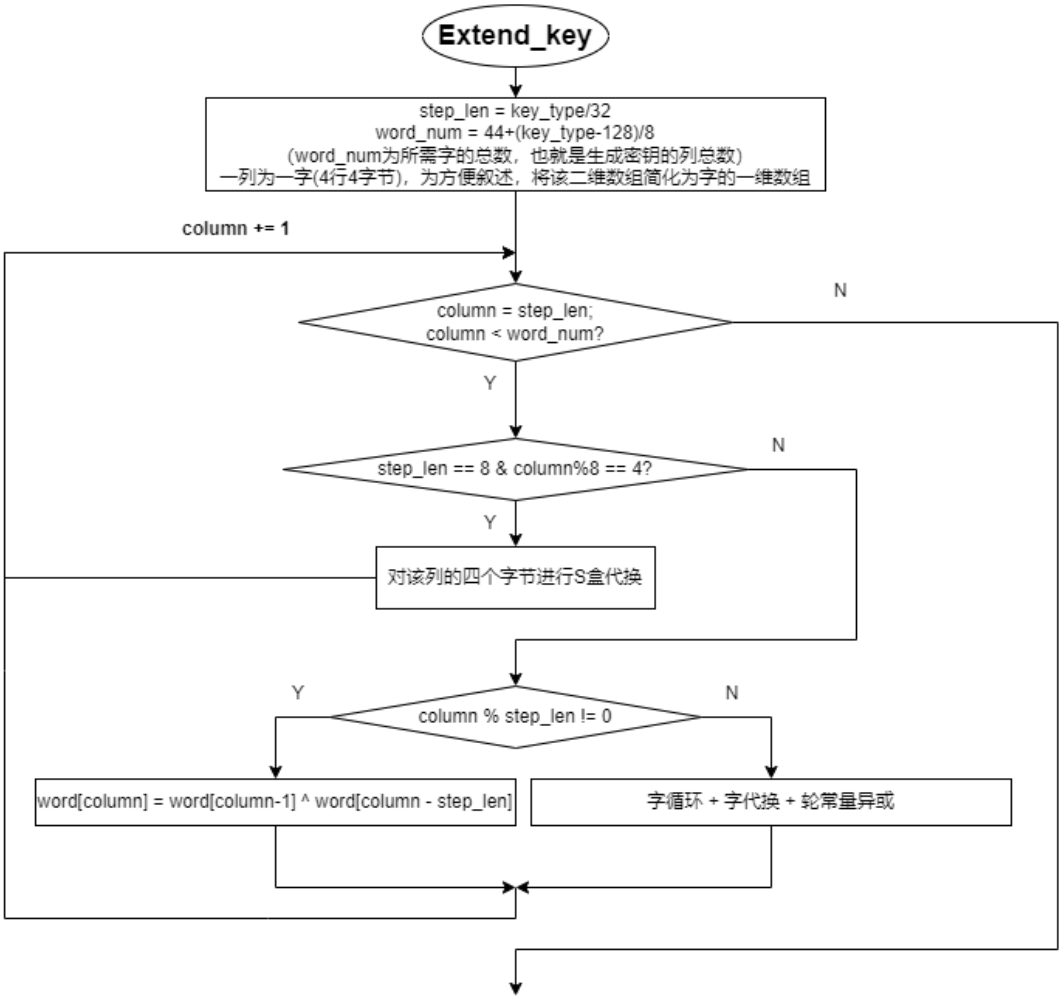
注：本题我在实现的过程中结合了*AES-192*及*AES-256*实现了拓展，为不重复叙述，将选作*1*一并叙述：

1. **算法流程：**

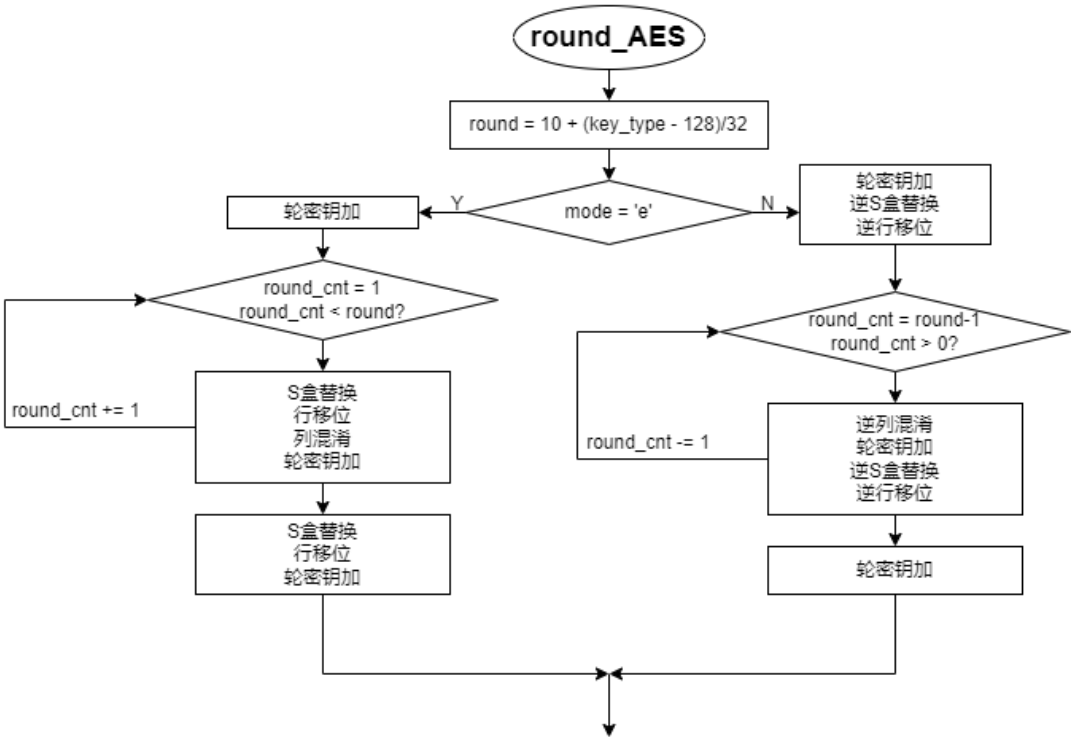
   - **函数调用图：**



- **函数流程图如下：**
  - **生成密钥算法流程图：**



  - **AES一轮加解密的流程图：**

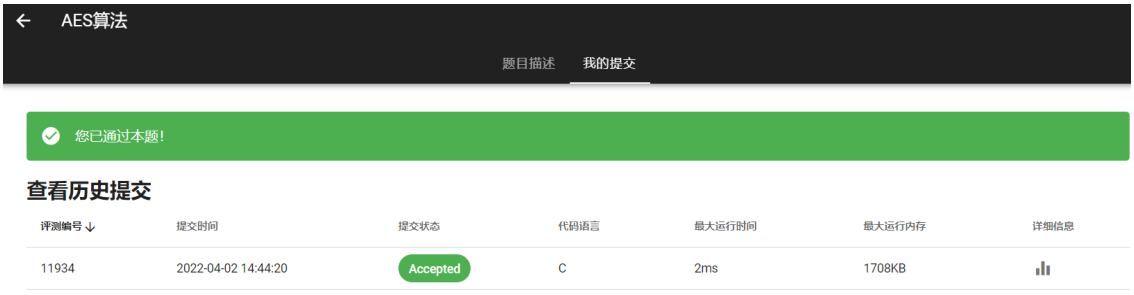


2. **测试样例及结果截图：**

1. AES算法测试样例及结果截图：

```
"E:\E_drive\clion\Project List\cryptography\Crypto_Experiment\cmake-build-debug\AES_Cipher.exe"
39
0x0ee863a353b646b8a8d105517804b12b
0xf7dad038fb3dc0ab58bb9987ce4aa0f3
0
0xad4b2698b827a6ffd5115686bf562134

Process finished with exit code 0
```

```
"E:\E_drive\clion\Project List\cryptography\Crypto_Experiment\cmake-build-debug\AES_Cipher.exe"
39
0x540ff554865c35abd37411eec81c5c68
0x996c94ae9556e87bd28eb3919a2a89d7
1
0xee5d7e7b9c9432de6628432b54c7907a

Process finished with exit code 0
```



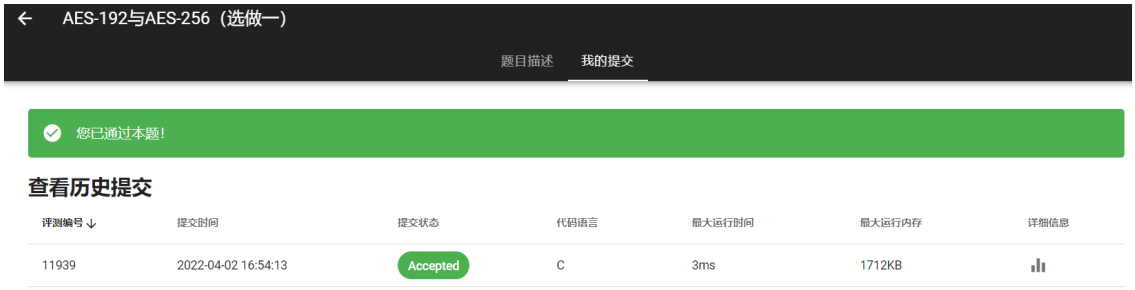
2. AES-192与AES-256（选做一）测试样例及结果截图：

```
"E:\E_drive\clion\Project List\cryptography\Crypto_Experiment\cmake-build-debug\AES_Cipher.exe"
192
39
0xc142a1e4acb27192f0e36869d2b5b9f8
0xebf353bd5a61d35084c0664197daf0a555d3c9990681171d
0
0xb0620b4b0b35fa6f4aaf1c2e7b52be72

Process finished with exit code 0
```

```
"E:\E_drive\clion\Project List\cryptography\Crypto_Experiment\cmake-build-debug\AES_Cipher.exe"
192
39
0x0774a2be3d31f2859d9d6ebb9f2198f1
0xc1188f350e36b2486e6f0bc86b3372335a459e9665240cd8
1
0xd04f98d501efb911254819ce8520741f

Process finished with exit code 0
```

```
"E:\E_drive\clion\Project List\cryptography\Crypto_Experiment\cmake-build-debug\AES_Cipher.exe"
256
39
0xda56a6cdfa205074e305a6a71b8107d7
0x69c0167a73c13af6da7d849f9349521a03f84f3c8535fb27df6a888db151c5d7
1
0x98209dc962820519dc1c7f2c07694f88

Process finished with exit code 0
```

```
"E:\E_drive\clion\Project List\cryptography\Crypto_Experiment\cmake-build-debug\AES_Cipher.exe"
256
39
0x2f59099dce0e691e4cc2f6c053eddc28
0x92850bebc6fd62c37d0263db5c24b2bcb6fb8a5351f606acf875bdf1cb145346
0
0xf4e18fb2ea29a1cfc9b2a70ccb1fb6c7

Process finished with exit code 0
```



3. **讨论与思考：**

    1. **比较 AES 与 DES 的异同，AES 相比于 DES 有哪些改进?**

        - DES:Data Encryption Standard

          对称加密算法，密钥长度64bit（有效密钥56,8位校验位），主要分三步

            1. 初始置换
            2. 乘积变换
            3. 逆初始置换

        - AES:Advanced Encryption Standard

          对称加密算法的一种标准，Rijndael是其中的一种具体实现；aes密钥长度必须是128bit（32*4），192bit（32*6），256bit（32*8）；而Rijndael的密钥长度为32*n,且总长度在[128,258]之间，主要分五步：

            1. 字节替换
            2. 行移位
            3. 列混淆
            4. 轮密钥加
            5. 密钥拓展

        - 优缺点对比：

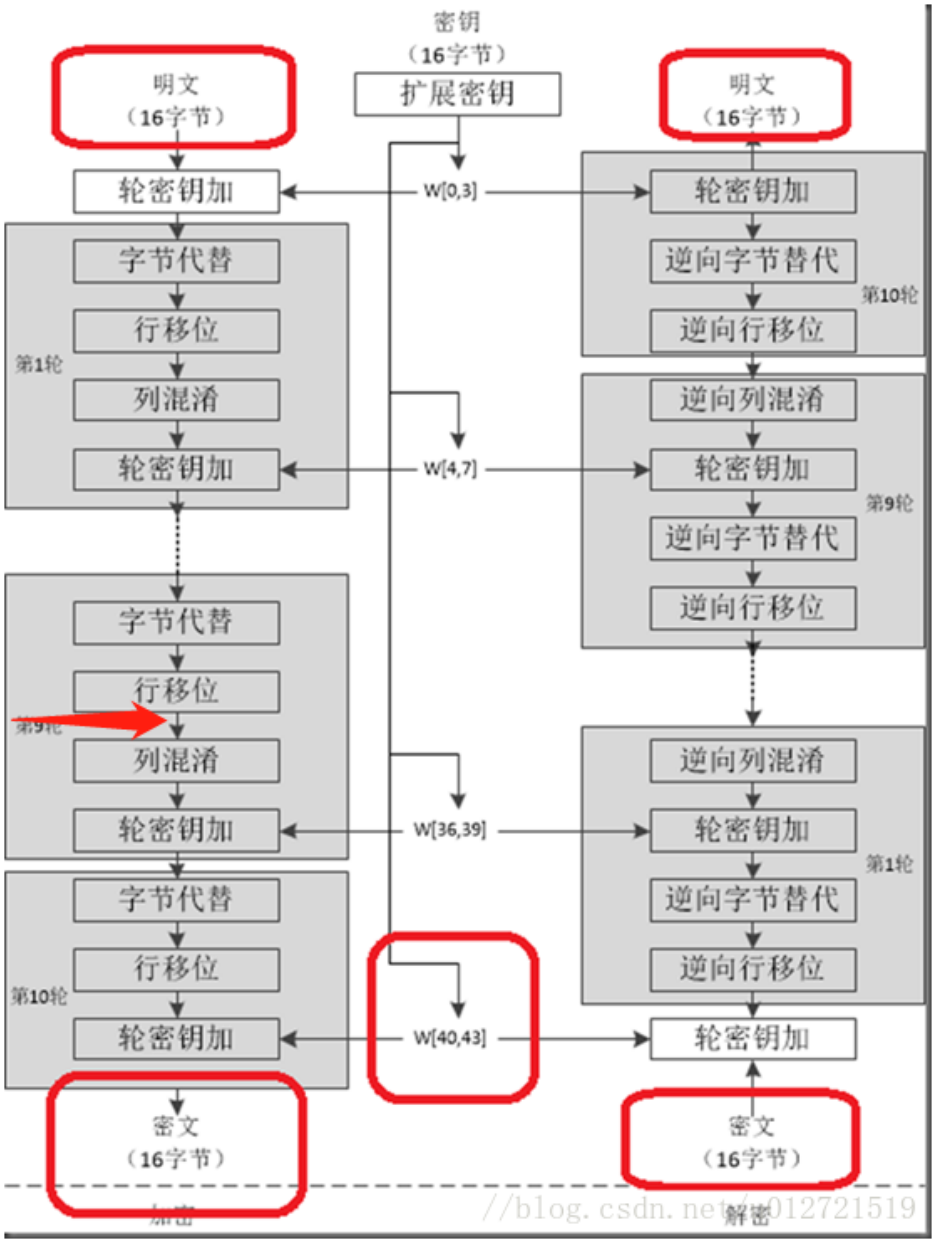
          总体来说aes是更高级的对称加密算法，比des及3des更安全高效。体现在：

            1. aes加密速度快
            2. aes安全性不差于3des
            3. aes密钥长度更长且可变

四、AESの差分攻击（选做二）

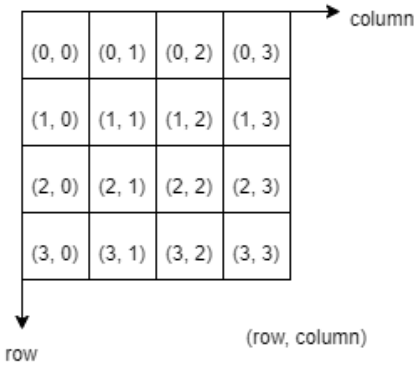
1. **攻击原理：**

    1. 由相关的文献我们可知：故障发生在第九轮行移位之后：



此时我们模拟在进行故障注入之后的加密情形：

对于一个加密块(4*4的方阵，每个位置存储一个字节)，如下图所示将块按行列进行编号：



假设$(row, column)$位置出现了故障注入，由加密流程的实施方式，可得：

- 进行**第九轮列混淆**后，错误扩散到了如下位置：

$(0, column)$
$(1, column)$
$(2, column)$
$(3, column)$

- 进行**第九轮轮密钥加**后，错误转移到了如下位置：
  $(0, column)$
  $(1, column)$
  $(2, column)$
  $(3, column)$
- 进行**第十轮S盒替换**后，错误转移到了如下位置：
  $(0, column)$
  $(1, column)$
  $(2, column)$
  $(3, column)$
- 进行**第十轮行移位**后，错误转移到了如下位置：
  $(0, (column + 4)\%4)$
  $(1, (column + 3)\%4)$
  $(2, (column + 2)\%4)$
  $(3, (column + 1)\%4)$
- 进行**第十轮轮密钥加**后，错误转移到了如下位置：
  $(0, (column + 4)\%4)$
  $(1, (column + 3)\%4)$
  $(2, (column + 2)\%4)$
  $(3, (column + 1)\%4)$

2. 为方便表示，我们只考虑这些错误位置的运算流程：

   进一步的，设在(r, c)位置注入的故障值为：ϵ，则有：

   注：为表示方便，以$C(i,j)_{\text{阶段下标}}$表示该阶段加密的正确值

   - $S(r, c) = correct_{after\_sr} \oplus \epsilon$
   - 进行**第九轮列混淆**时，实际上就是状态块($State\_block$)做如下矩阵乘法：

     $$State\_block' = \begin{bmatrix} 2 & 3 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 1 & 1 & 2 & 3 \\ 3 & 1 & 1 & 2 \end{bmatrix} * State\_block$$

     这里为表示方便，同时省去矩阵运算，我们设数组：

     $$co[4] = \{2, 1, 1, 3\}$$

     来存放该左乘矩阵的第一列的列向量；

     则经过该列混淆后，**错误位置的值相应变为**：

     $$\begin{aligned}
     S[0,c]_{after_mc9} &= (C[0,c]_{after_sr9} \oplus \epsilon) \otimes co[(0+4-r)\%4] \\
     &= (C[0,c]_{after_sr9} \otimes co[(0+4-r)\%4]) \oplus (\epsilon \otimes co[(0+4-r)\%4]) \\
     &= C[0,c]_{after_mc9} \oplus (\epsilon \otimes co[(0+4-r)\%4]) \\
     S[1,c]_{after_mc9} &= (C[1,c]_{after_sr9} \oplus \epsilon) \otimes co[(1+4-r)\%4] \\
     &= (C[1,c]_{after_sr9} \otimes co[(1+4-r)\%4]) \oplus (\epsilon \otimes co[(1+4-r)\%4]) \\
     &= C[1,c]_{after_mc9} \oplus (\epsilon \otimes co[(1+4-r)\%4]) \\
     S[2,c]_{after_mc9} &= (C[2,c]_{after_sr9} \oplus \epsilon) \otimes co[(2+4-r)\%4] \\
     &= (C[2,c]_{after_sr9} \otimes co[(2+4-r)\%4]) \oplus (\epsilon \otimes co[(2+4-r)\%4]) \\
     &= C[2,c]_{after_mc9} \oplus (\epsilon \otimes co[(2+4-r)\%4]) \\
     S[3,c]_{after_mc9} &= (C[3,c]_{after_sr9} \oplus \epsilon) \otimes co[(3+4-r)\%4] \\
     &= (C[3,c]_{after_sr9} \otimes co[(3+4-r)\%4]) \oplus (\epsilon \otimes co[(3+4-r)\%4]) \\
     &= C[3,c]_{after_mc9} \oplus (\epsilon \otimes co[(3+4-r)\%4])
     \end{aligned}$$

   - 进行**第九轮轮密钥加**后，错误位置的值相应为：

$$
\begin{aligned}
S[0,c]_{after\_ark9} &= C[0,c]_{after\_mc9} \oplus (\epsilon \otimes co[(0+4-r)\%4]) \oplus key_9[0,c] \\
&= C[0,c]_{after\_ark9} \oplus (\epsilon \otimes co[(0+4-r)\%4]) \\
S[1,c]_{after\_ark9} &= C[1,c]_{after\_mc9} \oplus (\epsilon \otimes co[(1+4-r)\%4]) \oplus key_9[1,c] \\
&= C[1,c]_{after\_ark9} \oplus (\epsilon \otimes co[(1+4-r)\%4]) \\
S[2,c]_{after\_ark9} &= C[2,c]_{after\_mc9} \oplus (\epsilon \otimes co[(2+4-r)\%4]) \oplus key_9[2,c] \\
&= C[2,c]_{after\_ark9} \oplus (\epsilon \otimes co[(2+4-r)\%4]) \\
S[3,c]_{after\_ark9} &= C[3,c]_{after\_mc9} \oplus (\epsilon \otimes co[(3+4-r)\%4]) \oplus key_9[3,c] \\
&= C[3,c]_{after\_ark9} \oplus (\epsilon \otimes co[(3+4-r)\%4])
\end{aligned}
$$

- 进行**第十轮S盒替换**后，错误位置的值相应为：

$$
\begin{aligned}
S[0,c]_{after\_S_{box}10} &= S_{box}(C[0,c]_{after\_ark9} \oplus (\epsilon \otimes co[(0+4-r)\%4])) \\
S[1,c]_{after\_S_{box}10} &= S_{box}(C[1,c]_{after\_ark9} \oplus (\epsilon \otimes co[(1+4-r)\%4])) \\
S[2,c]_{after\_S_{box}10} &= S_{box}(C[2,c]_{after\_ark9} \oplus (\epsilon \otimes co[(2+4-r)\%4])) \\
S[3,c]_{after\_S_{box}10} &= S_{box}(C[3,c]_{after\_ark9} \oplus (\epsilon \otimes co[(3+4-r)\%4]))
\end{aligned}
$$

- 进行**第十轮行移位**后，错误位置的值相应为：

$$
\begin{aligned}
S[0,(c+4)\%4]_{after\_sr10} &= S_{box}(C[0,c]_{after\_ark9} \oplus (\epsilon \otimes co[(0+4-r)\%4]) \\
S[1,(c+3)\%4]_{after\_sr10} &= S_{box}(C[1,c]_{after\_ark9} \oplus (\epsilon \otimes co[(1+4-r)\%4]) \\
S[2,(c+2)\%4]_{after\_sr10} &= S_{box}(C[2,c]_{after\_ark9} \oplus (\epsilon \otimes co[(2+4-r)\%4]) \\
S[3,(c+1)\%4]_{after\_sr10} &= S_{box}(C[3,c]_{after\_ark9} \oplus (\epsilon \otimes co[(3+4-r)\%4])
\end{aligned}
$$

- 进行**第十轮轮密钥加**后，错误位置的值相应为：

$$
\begin{aligned}
S[0,(c+4)\%4]_{after\_ark10} &= S_{box}(C[0,c]_{after\_ark9} \oplus (\epsilon \otimes co[(0+4-r)\%4]) \oplus key_{10}[0,(c+4)\%4] \\
&= wrong\_byte[0,(c+4)\%4] \\
S[1,(c+3)\%4]_{after\_ark10} &= S_{box}(C[1,c]_{after\_ark9} \oplus (\epsilon \otimes co[(1+4-r)\%4]) \oplus key_{10}[1,(c+3)\%4] \\
&= wrong\_byte[1,(c+3)\%4] \\
S[2,(c+2)\%4]_{after\_ark10} &= S_{box}(C[2,c]_{after\_ark9} \oplus (\epsilon \otimes co[(2+4-r)\%4]) \oplus key_{10}[2,(c+2)\%4] \\
&= wrong\_byte[2,(c+2)\%4] \\
S[3,(c+1)\%4]_{after\_ark10} &= S_{box}(C[3,c]_{after\_ark9} \oplus (\epsilon \otimes co[(3+4-r)\%4]) \oplus key_{10}[3,(c+1)\%4] \\
&= wrong\_byte[3,(c+1)\%4]
\end{aligned}
$$

- 正确的密文应为：

$$
\begin{aligned}
correct\_byte[0,(c+4)\%4] &= S_{box}(C[0,c]_{after\_ark9}) \oplus key_{10}[0,(c+4)\%4] \\
correct\_byte[1,(c+3)\%4] &= S_{box}(C[1,c]_{after\_ark9}) \oplus key_{10}[1,(c+3)\%4] \\
correct\_byte[2,(c+2)\%4] &= S_{box}(C[2,c]_{after\_ark9}) \oplus key_{10}[2,(c+2)\%4] \\
correct\_byte[3,(c+1)\%4] &= S_{box}(C[3,c]_{after\_ark9}) \oplus key_{10}[3,(c+1)\%4]
\end{aligned}
$$

由此可得：在**第十轮S盒替换**后，可得：

$$
\begin{aligned}
S[0,c]_{after\_S_{box}10} &= S_{box}(C[0,c]_{after\_ark9} \oplus (\epsilon \otimes co[(0+4-r)\%4])) \\
&= wrong\_byte[0,(c+4)\%4] \oplus correct\_byte[0,(c+4)\%4] \oplus S_{box}(C[0,c]_{after\_ark9}) \\
S[1,c]_{after\_S_{box}10} &= S_{box}(C[1,c]_{after\_ark9} \oplus (\epsilon \otimes co[(1+4-r)\%4])) \\
&= wrong\_byte[1,(c+3)\%4] \oplus correct\_byte[1,(c+3)\%4] \oplus S_{box}(C[1,c]_{after\_ark9}) \\
S[2,c]_{after\_S_{box}10} &= S_{box}(C[2,c]_{after\_ark9} \oplus (\epsilon \otimes co[(2+4-r)\%4])) \\
&= wrong\_byte[2,(c+2)\%4] \oplus correct\_byte[2,(c+2)\%4] \oplus S_{box}(C[2,c]_{after\_ark9}) \\
S[3,c]_{after\_S_{box}10} &= S_{box}(C[3,c]_{after\_ark9} \oplus (\epsilon \otimes co[(3+4-r)\%4])) \\
&= wrong\_byte[3,(c+1)\%4] \oplus correct\_byte[3,(c+1)\%4] \oplus S_{box}(C[3,c]_{after\_ark9})
\end{aligned}
$$

由此可知：设$C[i,c]_{after_ark9} = X_i$，其中$i = 0,1,2,3$，有：

$$
S_{box}(X_i \oplus (\epsilon \otimes co[(i+4-r)\%4])) = wrong\_byte[i,(c+4-i)\%4] \oplus correct\_byte[i,(c+4-i)\%4] \oplus S_{box}(X_i)
$$

在上式中，$X_i(i=0,1,2,3)$和ϵ作为未知数，我们可以通过遍历字节验证上式找到所有的可能值。由于对于这4个X_i，均是由错误注入ϵ得到的，所以我们以遍历ϵ作为大循环，遍历这四个X_i的值作为小循环即可找到所有可能的X_i值。

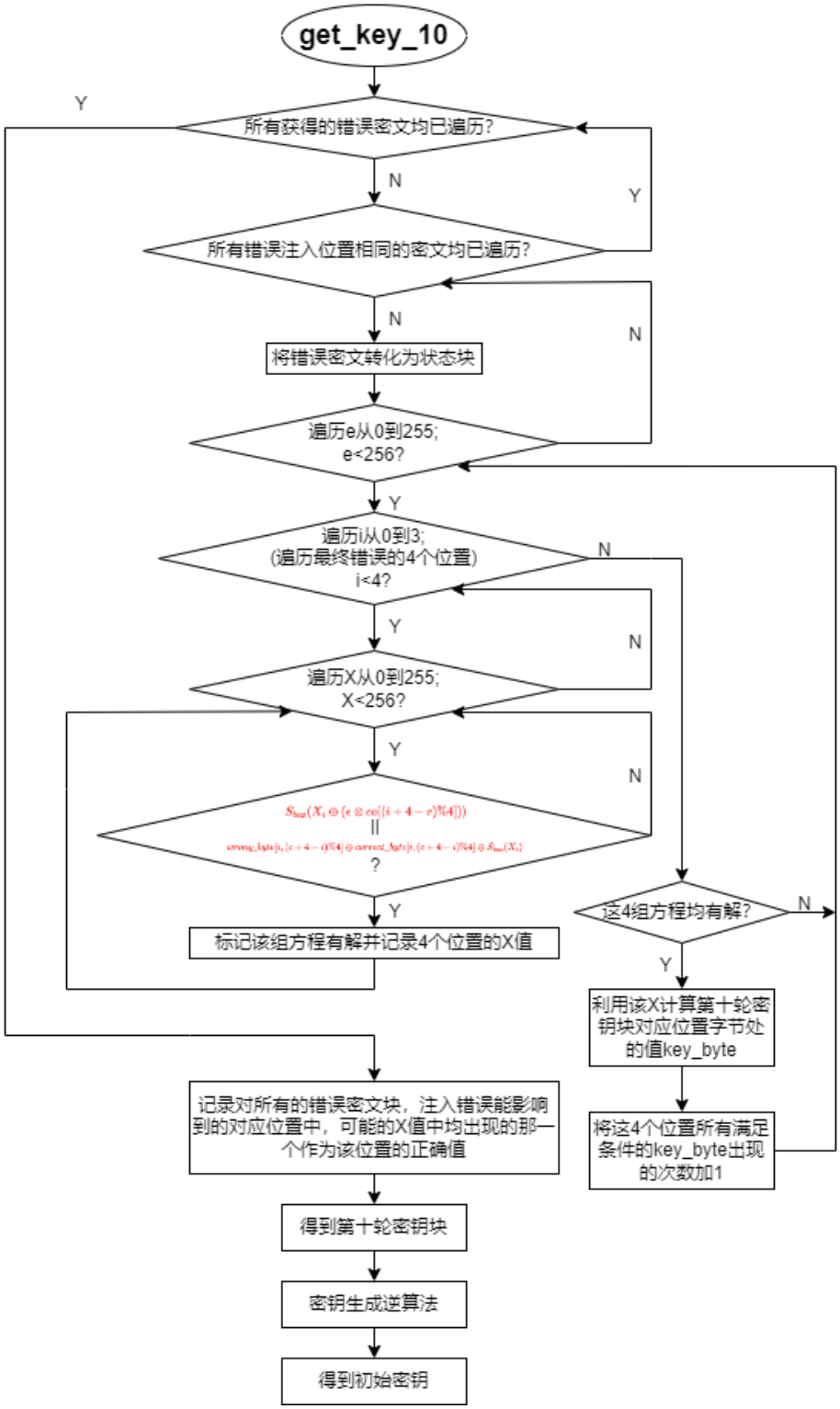
(注意：对于每个确定的ϵ，满足上述方程的四个X_i的值都存在，那么这样的取值集合：$(\epsilon, X_0, X_1, X_2, X_3)$才是有效的)

→如果我们通过对状态块里的每个字节实现多次注入，就可能通过筛选得到C_{after_ark9}，进而通过运算得到正确的第十轮密钥：

$$
\begin{aligned}
S_{box}(X_0) \oplus correct\_byte[0,(c+4)\%4] &= key_{10}[0,(c+4)\%4] \\
S_{box}(X_1) \oplus correct\_byte[1,(c+3)\%4] &= key_{10}[1,(c+3)\%4] \\
S_{box}(X_2) \oplus correct\_byte[2,(c+2)\%4] &= key_{10}[2,(c+2)\%4] \\
S_{box}(X_3) \oplus correct\_byte[3,(c+1)\%4] &= key_{10}[3,(c+1)\%4]
\end{aligned}
$$

2. **攻击流程图：**

基于上面的攻击原理我们可以得到相应的攻击流程图：



3. **伪代码如下：**

$$
\begin{aligned}
&Algorithm\quad Attack\_AES\\
&Input: correct\_cipher\_block, wrong\_cipher\_block, co[4], S_{box}, round\_check[4]\\
&output: round_{10}\_key\_block\\
&1: for\ all\ the\ wrong\_cipher\_block\\
&2:\quad for\ \epsilon \leftarrow 0\ to\ 255\\
&3:\quad\quad init\ related\ var\\
&4:\quad\quad\quad for\ round \leftarrow 0\ to\ 3\\
&5:\quad\quad\quad\quad for\ X \leftarrow 0\ to\ 255\\
&6:\quad\quad\quad\quad\quad if\ S_{box}(X_i \oplus (\epsilon \otimes co[(i+4-r)\%4])) == wrong\_byte[i,(c+4-i)\%4] \oplus\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad correct\_byte[i,(c+4-i)\%4] \oplus S_{box}(X_i)\\
&7:\quad\quad\quad\quad\quad\quad round\_check[round] = true\\
&8:\quad\quad\quad\quad\quad\quad calculate\ round_{10}\_related\_bytes\\
&9:\quad\quad\quad\quad\quad end\ if\\
&10:\quad\quad\quad\quad end\ for\\
&11:\quad\quad\quad end\ for\\
&12:\quad\quad if\ round\_check[0,1,2,3] == true\\
&13:\quad\quad\quad store\ round_{10}\_related\_bytes\\
&14:\quad\quad end\ if\\
&15:\quad end\ for\\
&16: end\ for\\
&17: for\ pos \leftarrow every\_byte\_of\_round_{10}\_key\_block\\
&18:\quad if\ round_{10}\_related\_bytes\ appears\ as\ affected\\
&19:\quad\quad round_{10}\_key\_block[pos] = round_{10}\_related\_bytes\\
&20:\quad end\ if\\
&21: end\ for\\
&22: return\ round_{10}\_key\_block
\end{aligned}
$$

4. **函数调用图：**

5. **测试样例及结果截图：**

```
0x1f29a639b7751260e0a3fc155851c04b
0x1f291f39b753126086a3fc155851c0e3
0x1f290239b77f12609da3fc155851c0a9
0x1f29d439b7e31260baa3fc155851c0fc
0x1f294439b72e126084a3fc155851c0f8
0x1f29eb72b7719a600b00fc150951c009
0x1f29eb6fb7712e600b07fc156351c009
0x1f29eb2bb771ff600b89fc157f51c009
0x1f29eb46b771a3600bfbfc156151c009
0x1f29ebe1b7713e600b96fc150351c009
0x1f29eb5fb77158600bbbfc154051c009
0x1f29eb2fb77179600b76fc153351c009
0x1f29eb95b77149600b0dfc151a51c009
0x1f29eb98b771d1600b60fc15cb51c009
0x1f29eb8ab7719f600b94fc159151c009
0xaeb6573144cf754b3aab40770a79f859
Process finished with exit code 0
```

AESの差分攻击（选做二）

题目描述　我的提交

您已通过本题!

查看历史提交

评测编号	提交时间	提交状态	代码语言	最大运行时间	最大运行内存	详细信息
12615	2022-04-06 20:46:38	Accepted	C	204ms	1736KB	

Rows per page: 10　1-1 of 1