

密码学实验实验报告五

18374480-黄翔

2021 年 4 月 21 日

1 实验目的

1. 通过本次实验, 熟练掌握 AES128 的加解密流程。
2. 了解 AES192 与 AES256 加解密流程。
3. 了解 S 盒生成原理。
4. 通过尝试 AES 攻击, 了解常用的攻击方法。

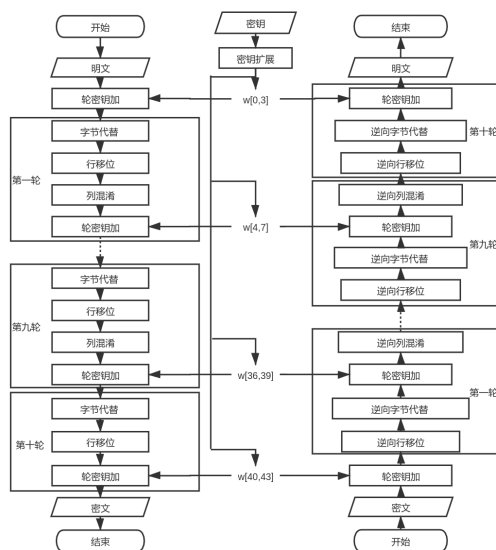
2 实验环境

python 3.9.1+

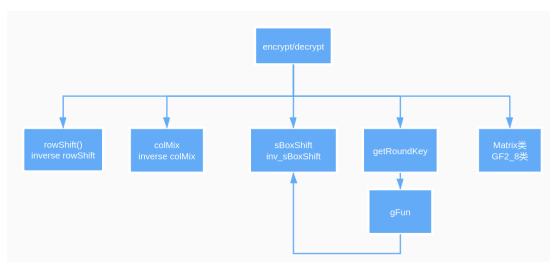
3 实验内容

3.1 AES128 加解密算法

3.1.1 算法流程图



3.1.2 函数调用关系



3.1.3 算法伪代码

Algorithm 1 AES 加密

Input: 明文 P (128 bits), 密钥 K (128 bits)

Output: 密文 C (128 bits)

- 1: **function** $Encrypt(P, K)$
- 2: $w[0, 44] \leftarrow K$ 密钥扩展

```

3:    $temp \leftarrow P \oplus w[0, 3]$ 
4:   for  $i \leftarrow 1$  to 10 do
5:        $temp \leftarrow$  字节代替  $subKey(temp)$ 
6:        $temp \leftarrow$  行移位  $rowShift(temp)$ 
7:       if  $i \neq 10$  then
8:            $temp \leftarrow$  列混淆  $colMix(temp)$ 
9:       end if
10:       $temp \leftarrow temp \oplus w[4 \times i, 4 \times i + 3]$ 
11:  end for
12:  return  $C \leftarrow temp$ 
13: end function

```

Algorithm 2 AES 解密

Input: 密文 C (128 bits), 密钥 K (128 bits)

Output: 明文 P (128 bits)

```

1: function  $Decrypt(C, K)$ 
2:    $w[0, 44] \leftarrow K$  密钥扩展
3:    $temp \leftarrow C \oplus w[40, 43]$ 
4:   for  $i \leftarrow 1$  to 10 do
5:        $temp \leftarrow$  逆向字节代替  $subKey(temp, inverse = True)$ 
6:        $temp \leftarrow$  逆向行移位  $rowShift(temp, inverse = True)$ 
7:       if  $i \neq 10$  then
8:            $temp \leftarrow$  逆向列混淆  $colMix(temp, inverse = True)$ 
9:       end if
10:       $temp \leftarrow temp \oplus w[4 \times (10 - i), 4 \times (10 - i) + 3]$ 
11:  end for
12:  return  $P \leftarrow temp$ 
13: end function

```

3.1.4 测试样例及结果截图

```

cypher: ff0b844a0853bf7c6934ab4364148fb9
plaintext: 0123456789abcdeffedcba9876543210

cypher: f3855216ddf401d4d42c8002e686c6e7
plaintext: 41b267bc5905f0a3cd691b3ddaee149d

```

3.1.5 总结

编程相关 通过使用自定义类 *Matrix* 以及 *GF2*, 使得代码更加模块化, 且编写 *AES* 类时无需关注运算细节, 简化了编程的复杂度。

明文	密文	相差位
0x0123456789abcdeffedcba9876543210	0xff0b844a0853bf7c6934ab4364148fb9	58 bits
0x0023456789abcdeffedcba9876543210	0x612b89398d0600cde116227ce72433f0	

表 1: 明文雪崩效应

密钥	密文	相差位
0x0f1571c947d9e8590cb7add6af7f6798	0xff0b844a0853bf7c6934ab4364148fb9	53 bits
0x0e1571c947d9e8590cb7add6af7f6798	0xfc8923ee501a7d207ab670686839996b	

表 2: 密钥雪崩效应

AES 雪崩效应

AES 结构设计

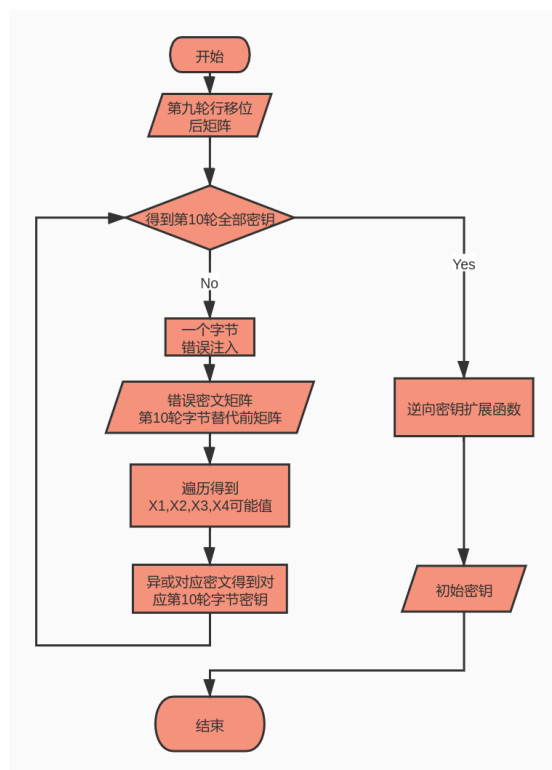
1. AES 以轮密钥加作为起始以及结束, 因为仅在轮密钥加使用密钥, 以其他不需要密钥的运算作为起始结束不能增加算法安全性。
2. 轮密钥实质是一种 Vernam 密码形式

扩散与混淆 AES 充分体现了密码设计中的扩散与混淆理念。

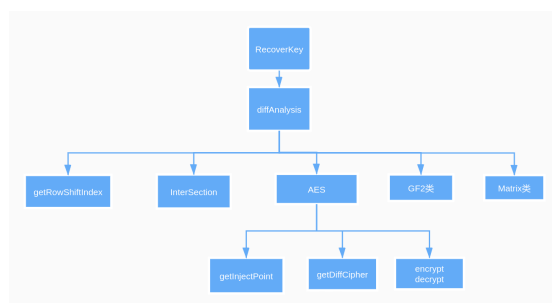
1. 扩散
 - 大扩散: 行移位变换 (字的扩散)
 - 中扩散: 列混淆变换 (字节的扩散)
 - 小扩散: S 盒构造时对 S 盒每个字节的变换 (位的扩散)
2. 混淆: 轮密钥加后 S 盒字节替代

3.2 AES 第九轮差分错误攻击

3.2.1 算法流程图



3.2.2 函数调用关系图



3.2.3 算法伪代码

Algorithm 3 AES 第九轮差分错误攻击

Input: 明文 P (128 bits), 密文 C (128 bits), 注入矩阵 M (128 bits)

Output: 密钥 K (128 bits)

```
1: 第十轮密钥矩阵  $roundKey_{10} \leftarrow 4 \times 4$  零阵
2:  $times[4] = (0 \times 02, 0 \times 01, 0 \times 01, 0 \times 03)$ 
3: for  $i \leftarrow 0$  to 4 do
4:    $M' : M[0, i] \leftarrow M[0, i] + \varepsilon$ 
5:    $C' \leftarrow$  由  $M'$  得到密文矩阵
6:   for  $j \leftarrow 0$  to 4 do
7:      $r, c \leftarrow$  列混淆后  $M'[j, i]$  再行移位最终影响的字节位置
8:      $\varepsilon_j \leftarrow C[r, c] \oplus C'[r, c]$ 
9:      $X_j \leftarrow$  遍历  $x, \varepsilon$  得到满足  $S(x + times[j] \times \varepsilon) = S(x) + \varepsilon_j$  的集合
10:     $X[j]$  与上一次注入得到集合取交
11:    if  $\forall t \in \{0, 1, 2, 3\}$   $len(X[t])$  is 1 then
12:      for  $t \leftarrow 0$  to 3 do
13:         $roundKey_{10}[r, c] \leftarrow X[t]$ 
14:      end for
15:    else
16:      goto line 4
17:    end if
18:  end for
19: end for
20:  $K \leftarrow$  对  $roundKey_{10}$  逆向轮密钥扩展
21: return  $K$ 
```

Algorithm 4 逆向轮密钥扩展

Input: $roundKey_{10}$

Output: Key

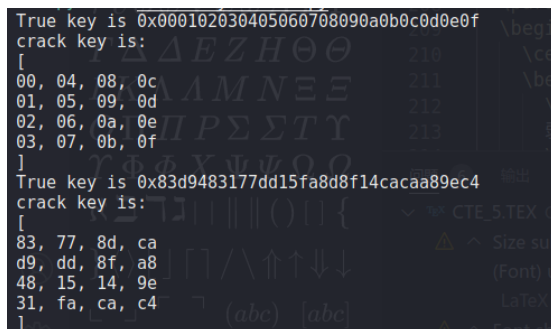
```
1:  $w[44]$ 
2: for  $i \leftarrow 0$  to 3 do
3:    $w[i] \leftarrow roundKey_{10}$  的第  $i$  列
4: end for
5: for  $i \leftarrow 0$  to 9 do
6:    $RC \leftarrow [2^{9-i}, 0, 0, 0]$  //  $GF(2^8)$  上运算
7:    $temp[4] = (0, 0, 0, 0)$ 
```

```

8:   for  $j \leftarrow 1$  to 3 do
9:        $temp[j] \leftarrow w[4 \times i + j] \oplus w[4 \times i + j - 1]$ 
10:   end for
11:    $temp[0] \leftarrow w[4 \times i] \oplus g(temp[3], RC)$ 
12:    $w$  append  $temp$ 
13: end for
14: return  $Key \leftarrow [w[40], w[41], w[42], w[43]]$ 

```

3.2.4 测试样例及结果截图



```

True key is 0x000102030405060708090a0b0c0d0e0f
crack key is:
[
00, 04, 08, 0c
01, 05, 09, 0d
02, 06, 0a, 0e
03, 07, 0b, 0f
]
True key is 0x83d9483177dd15fa8d8f14caca89ec4
crack key is:
[
83, 77, 8d, ca
d9, dd, 8f, a8
48, 15, 14, 9e
31, fa, ca, c4
]

```

3.2.5 总结

代码设计 由于设计 AES 类时对如 S 盒替换等普适性函数归为了对象函数，在进行此选做代码编写时带来诸多不便，不得不令生成错误注入和错误分析放在了同一个函数内，代码可读性有所降低（但是确实没有直接用到注入的错误!!）。

攻击原理 故障注入攻击的成功是由于线性变换的分配律，导致能够很容易的得到 $\varepsilon_0, \varepsilon_1, \varepsilon_2, \varepsilon_3$ ，错误只经过第九轮列混合到结束的处理，没有充分的混淆和扩散，很容易被遍历出来。

4 总结

通过本次实验，我熟练掌握了 AES128 的加解密流程，并了解了 AES192 与 AES256 加解密流程，S 盒生成原理。最后尝试了 AES 故障注入攻击，了解了常用的攻击方法。

5 思考题

1. 在 DES 中，数据块分为两半；在 AES 中，整个数据块都作为单个矩阵进行处理。
2. DES 属于 Feistel 密码结构；AES 致力于替代和置换原则。
3. DES 密钥一般为 64 位；AES 密钥可以是 128,192 或 256 位。
4. DES 为 16 轮；AES128 为 10 轮，AES192 为 12 轮，AES256 为 14 轮。
5. DES 的密钥空间较小，安全性较低；AES 相对而言具有较大的密钥，因此更加安全。