05Reverse基础（六）

原创

beloved_fjq 于 2020-06-06 21:25:08 发布 187 收藏

05Reverse基础（六）

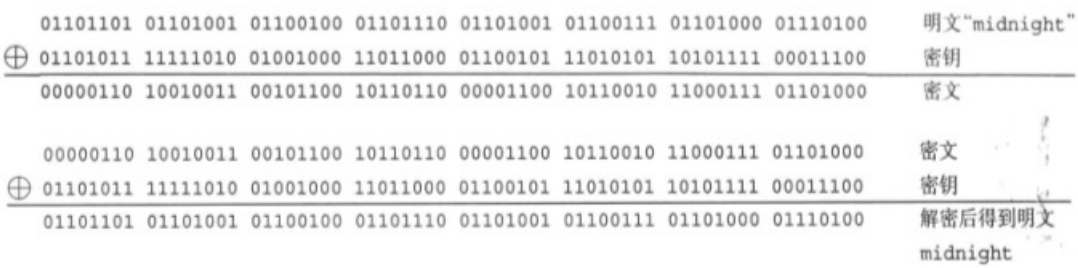
任务内容

1.完成攻防世界 REVERSE 新手区 7-12道题目并编写WriteUp。（网上有其他人写的WP，可以参考着学习，做完这些题再回过头做之前的四题）
2.学习密码学知识，推荐阅读书籍《图解密码学》，阅读第三章、第四章
3.阅读书籍《程序员的自我修养》

学习笔记

《图解密码学》

第三章 对称密码（共享密钥密码）——对相同的密钥进行加密和解密
3.1炒鸡蛋和对称密码
为了使明文无法被推测，打乱密文（打乱的是比特序列），只要将数据转换成比特序列，就能够加密
3.3从文学密码到比特序列密码
XOR运算（异或）
规律：两个相同的数进行异或的结果一定为0
计算与加解密规律很相似：
（1）明文A用密钥B加密，得到密文A异或B
（2）密文A异或B用密钥B解密，得到明文A
3.4一次性密码本——绝对不会破译的密码
一次性密码本（维纳密码）是绝对不可能破译的，无条件安全的。



3.5DES
将64bit的明文加密成64bit的密文的对称密码算法，密钥长度是56bit，DES是分组密码的一种（64bit为一组）

DES基本结构：也称为Feistel网络

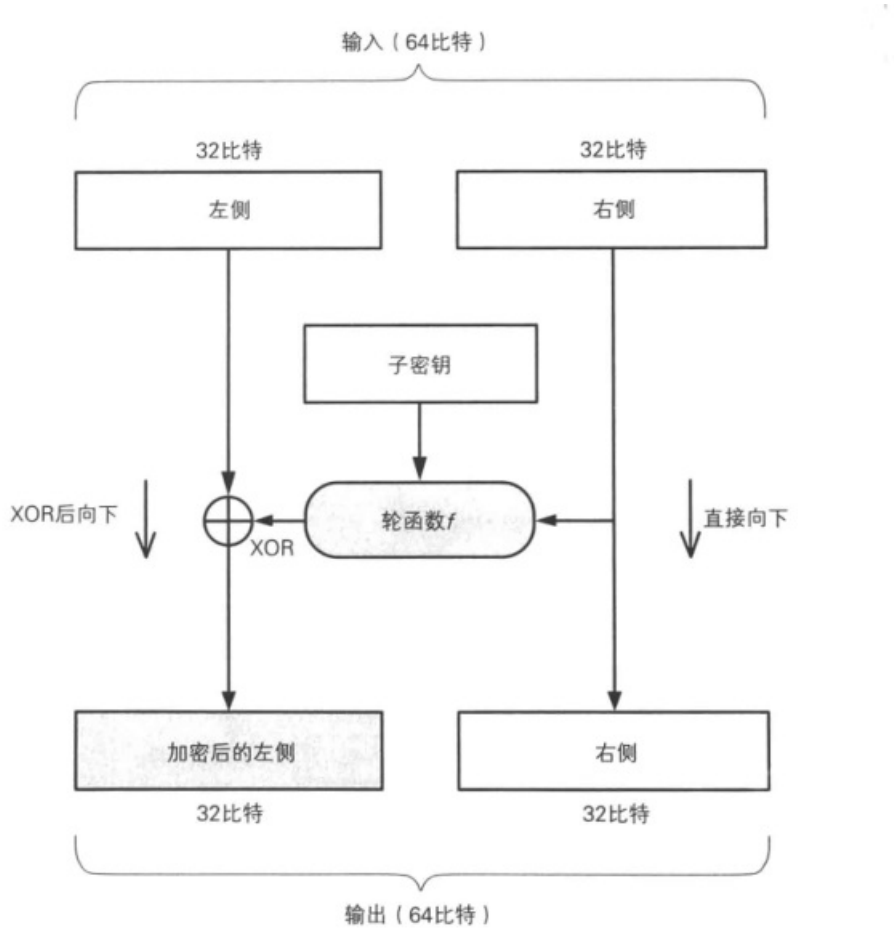


图 3-3　Feistel 网络中的一轮

具体步骤：

(1) 将输入的数据等分为左右两部分。
(2) 将输入的右侧直接发送到输出的右侧。
(3) 将输入的右侧发送到轮函数。
(4) 轮函数根据右侧数据和子密钥，计算出一串看上去是随机的比特序列。
(5) 将上一步得到的比特序列与左侧数据进行 XOR 运算，并将结果作为加密后的左侧。

注意：每一轮都需要使用不同的子密钥，如用相同的子密钥运行两次该网络就能将数据还原；网络的轮数可以任意增加，加密时无论使用任何函数作为轮函数都可以正确解密；加密和解密可以用完全相同的结构来实现
3.6三重DES（将DES重复三次：加密—解密—加密：具有兼容性）
1.目的是为了增强DES的强度
2.3DES的密钥长度是56^3=168bit
3.DES-EDE2表示密钥1，2相同，3不同
DES-EDE3表示所有密钥都不同
4.三重DES解密与加密正好相反，密钥3，2，1的顺序执行解密-加密-解密。
3.7AES的选定过程（新标准）
how：通过竞争来实现标准化
3.8Rijndael（比利时）
1.属于分组密码算法，分组长度128bit，标准规格的密钥长度只有128，192，256三种
2.使用了SPN结构
3.加密与解密不可以用同一种结构实现：

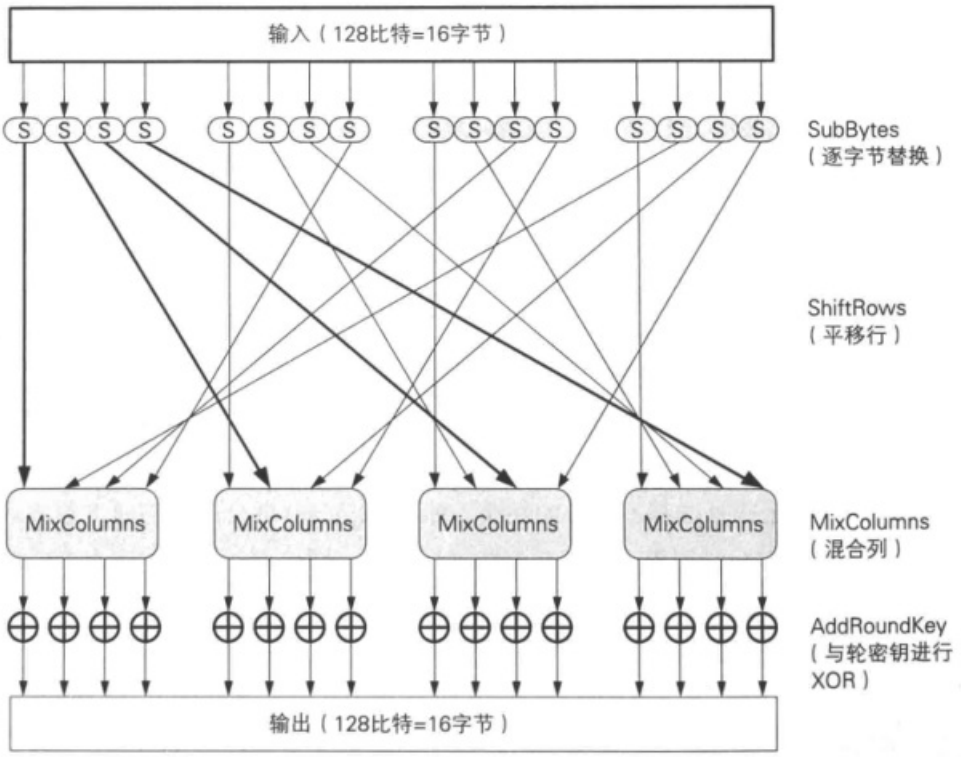


图 3-11　Rijndael 加密中的一轮

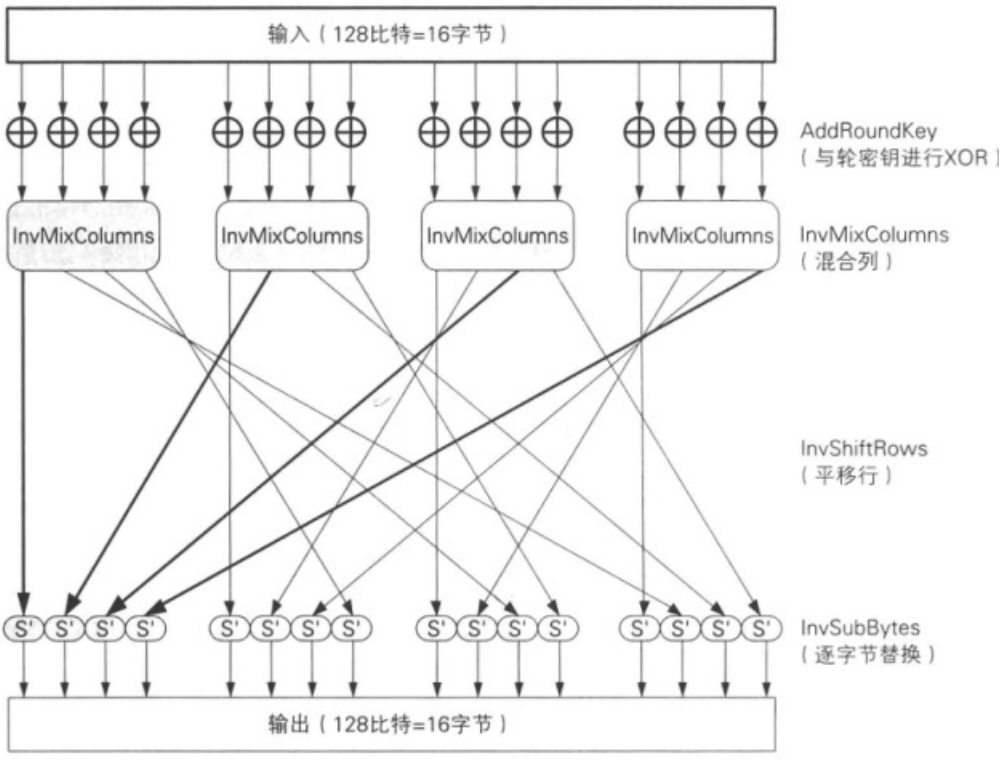


图 3-12　Rijndael 解密中的一轮

注意：以上这些密码，我们不应该使用任何我们自制的密码算法，也不该使用DES（暴力破解会破译），要用AES。

第四章 分组密码的模式——分组密码是如何迭代的

4.2分组密码的模式

1.密码算法可分为分组密码和流密码：分组密码每次只能处理特定长度的一块数据（分的组）的一类密码算法；流密码是对数据流进行连续处理的一类密码算法。一次性密码本属于流密码，而大多数的对称密码都属于分组密码。

2.模式：迭代的方法

模式有很多种类，分组密码的主要模式有以下 5 种：

- ECB 模式：Electronic CodeBook mode（电子密码本模式）
- CBC 模式：Cipher Block Chaining mode（密码分组链接模式）
- CFB 模式：Cipher FeedBack mode（密文反馈模式）
- OFB 模式：Output FeedBack mode（输出反馈模式）
- CTR 模式：CounTeR mode（计数器模式）

4.3ECB模式（电子密码本模式）

将明文加密之后的结构直接成为密文分组

风险：攻击者无需破译密码就可以操纵明文

4.4CBC模式（密文分组链接模式）

1.首先将明文分组与前一个密文分组进行XOR运算，然后再进行加密。

2.初始化向量（作为第一个明文的前一个密文分组）

3.损坏

对存在损坏的分组的密文进行解密时的情形（CBC模式）

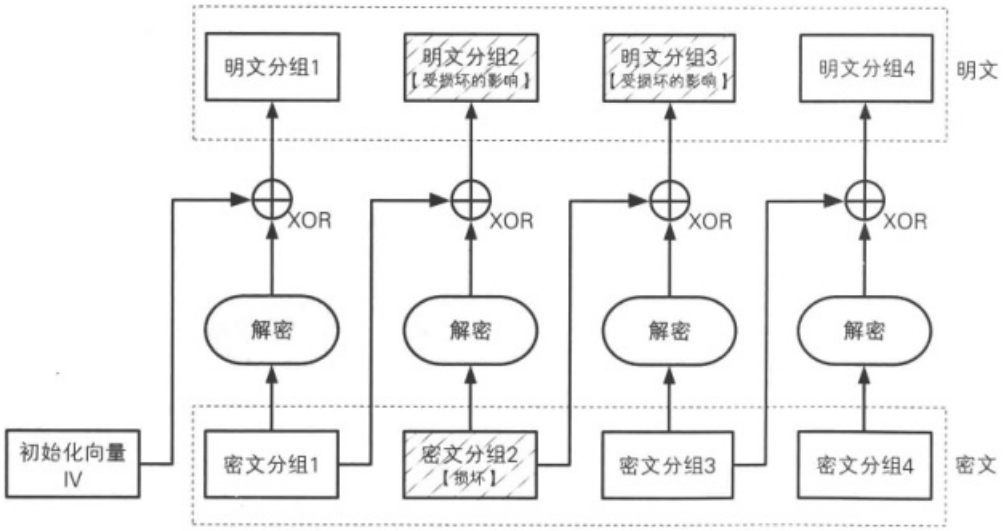


图 4-5　CBC 模式加密的密文分组损坏时，会影响 2 个明文分组

4.比特缺失

对其中1个密文分组中存在比特缺失的密文进行解密时的情形（CBC模式）

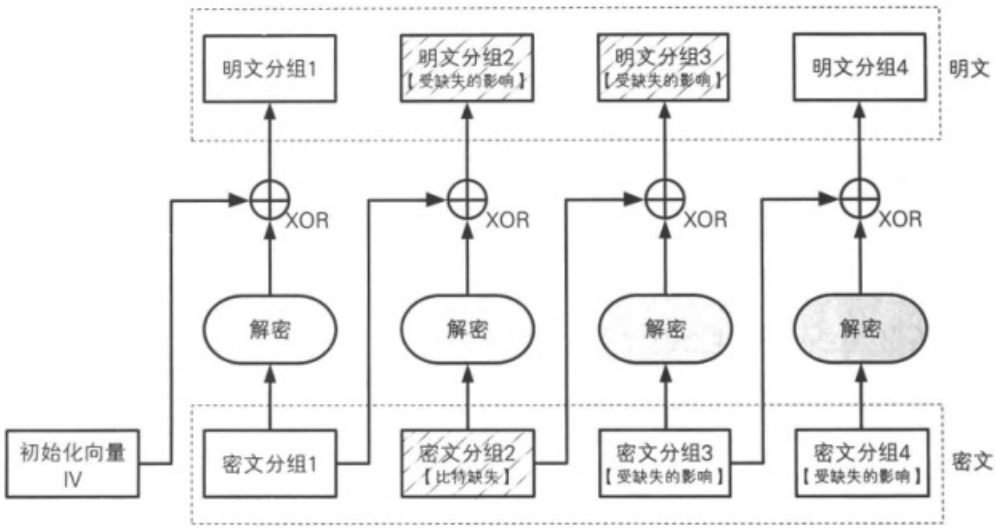


图 4-6　CBC 模式中密文分组存在缺失的比特时，之后的所有明文分组都会受到影响

4.5CFB模式（密文反馈模式）

1.明文与密文之间没有加密，只有一个XOR

2.由密码算法所生成的比特序列称为密钥流；该模式可以看作是一种使用分组密码来实现流密码的方式。

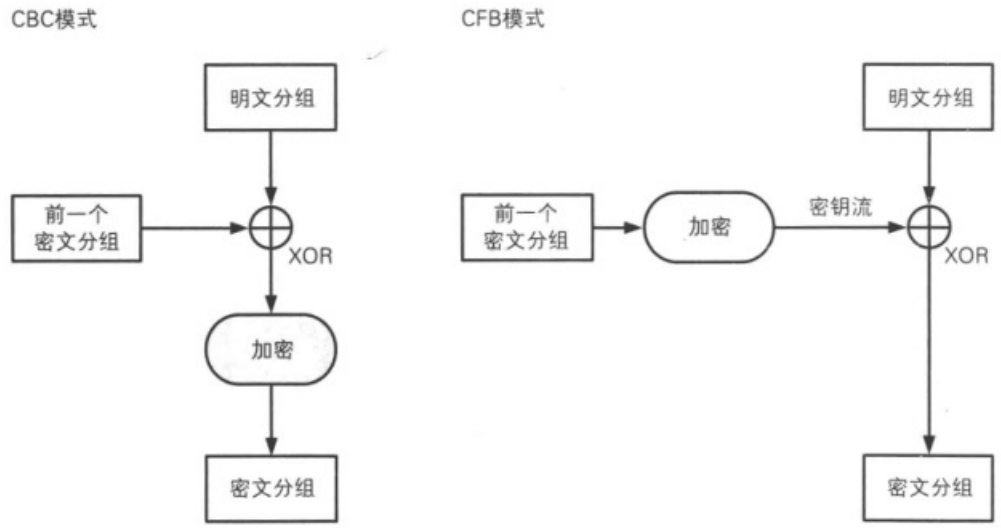


图 4-10　CBC 模式与 CFB 模式的对比

3.可实施重放攻击

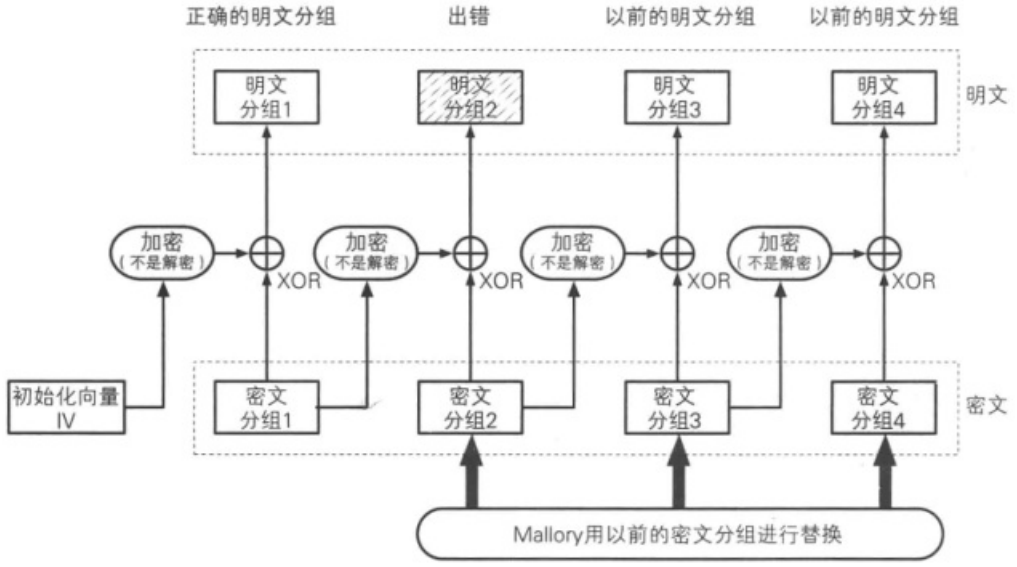
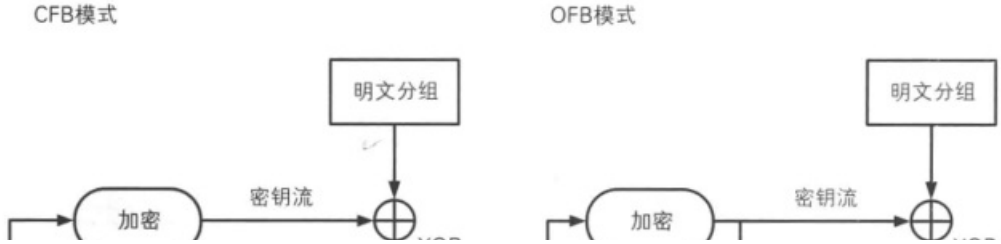


图 4-11　对 CFB 进行重放攻击

4.6OFB模式（输出反馈模式）

1.将明文分组与密码算法的输出进行XOR来产生密文分组

2.CFB模式与OFB模式的区别仅仅在于密码算法的输入



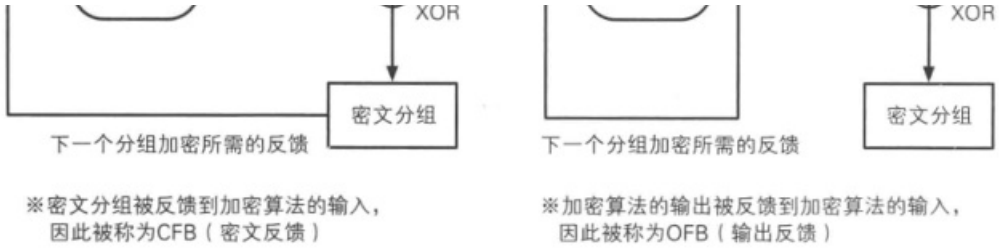


图 4-13　CFB 模式与 OFB 模式的对比

4.7CTR模式（计数器模式）

1.是一种通过将逐次累加的计数器进行加密来生成密钥流的流密码，最终的密文分组是通过计数器加密得到的比特序列，与明文分组进行XOR而得到的。

2.计数器的生成方法：

后续。。。

任务实现

006python-trade

文件发现是一个.pyc文件，用python反编译在线工具反编译这个.pyc文件得到源码，并写出求解代码：

```
1  import base64
2
3  def encode(message):
4      s = ''
5      for i in message:
6          x = ord(i) ^ 32
7          x = x + 16
8          s += chr(x)
9
10     return base64.b64encode(s)
11
12 correct = 'XlNkVmtUI1MgXWBZXCFeKY+AaXNt'
13 flag = ''
14 print 'Input flag:'
15 flag = raw_input()
16 if encode(flag) == correct:
17     print 'correct'
18 else:
19     print 'wrong'
```


test.py - C:/Users/lenovo/Desktop/test.py (3.8.3)

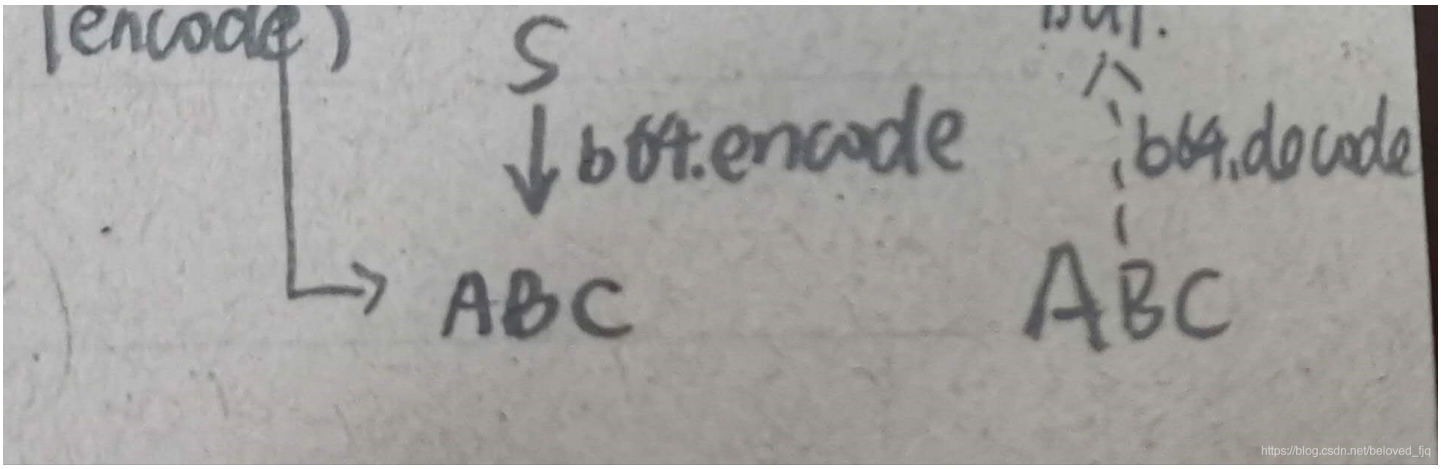
File Edit Format Run Options Window Help

```
import base64
buf = base64.b64decode('XlNkVmtUI1MgXWBZXCFeKY+AaXNt')
flag = ''
for i in buf:
    i -= 16
    i ^= 32
    flag += chr(i)
print(flag)
```


注意：ord()函数是Python的内置函数，是chr()函数的配对函数，以一个字符作为参数，返回对应的Unicode数值（Unicode是一种字符编码方案，它为每种语言中的每个字符都设定了统一唯一的二进制编码），就是说ord()函数主要用来返回对应字符的ascii码，chr()主要用来表示ascii码对应的字符。^是按位异或运算符。

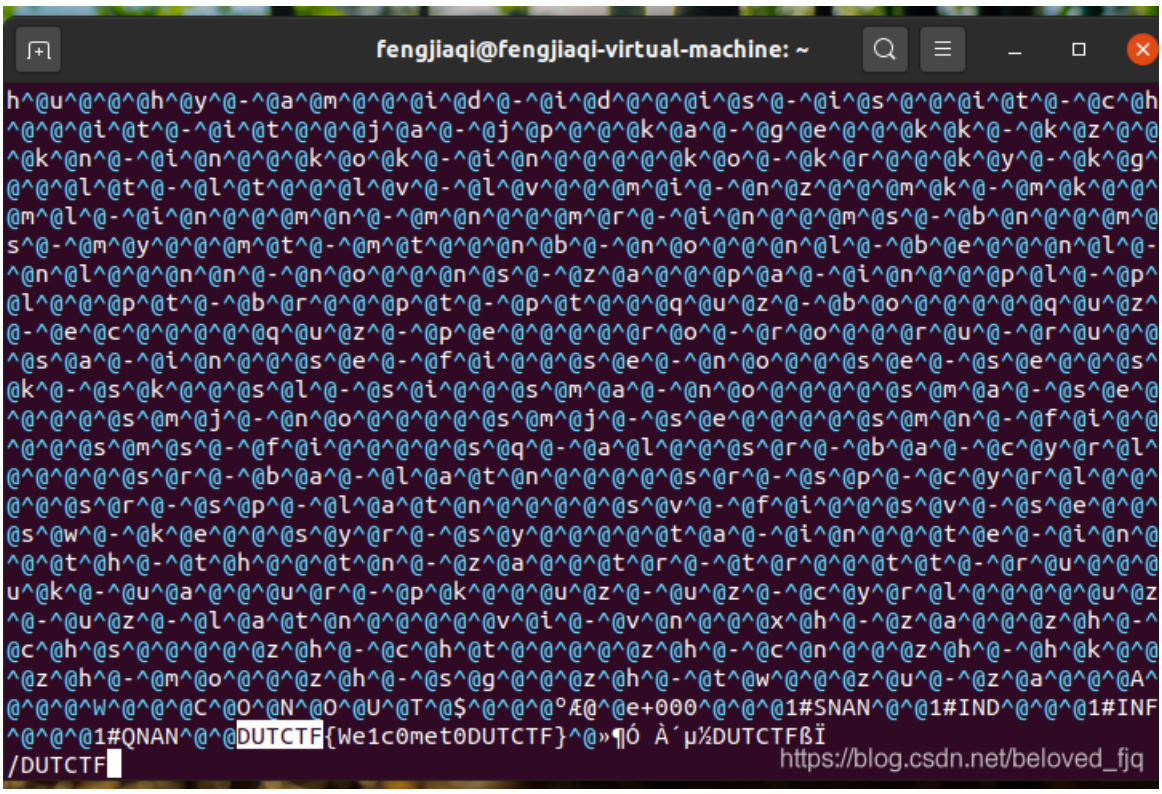
关键点：encode(flag) == “XlNkVmtUI1MgXWBZXCFeKY+AaXNt”

思路：



009re1

方法1：放到vim里面搜索字符串：DUTCTF（直接输入/查找1111）



方法2：读反编译代码

看一下代码：

```
1  int __cdecl main(int argc, const char **argv, const char **envp)
2  {
3    int v3; // eax
4    __int128 v5; // [esp+0h] [ebp-44h]
5    __int64 v6; // [esp+10h] [ebp-34h]
6    int v7; // [esp+18h] [ebp-2Ch]
7    __int16 v8; // [esp+1Ch] [ebp-28h]
8    char v9; // [esp+20h] [ebp-24h]
9
10   _mm_storeu_si128((__m128i *)&v5, _mm_loadu_si128((const __m128i *)&xmmword_413E34));
11   v7 = 0;
12   v6 = qword_413E44;
13   v8 = 0;
14   printf("欢迎来到DUTCTF呦\n");
15   printf("这是一道很可爱很简单的逆向题呦\n");
16   printf("输入flag吧:");
```

```
17  scanf("%s", &v9);
18  v3 = strcmp((const char *)&v5, &v9);
19  if ( v3 )
20    v3 = -(v3 < 0) | 1;
21  if ( v3 )
22    printf(aFlag_0);
23  else
24    printf((const char *)&unk_413E90);
25  system("pause");
26  return 0;
27}
```


指令名：void _mm_store_si128 (__m128i *p, __m128i a);
功能：可存储128位数据；
说明：将__m128i 变量a的值存储到p所指定的变量中去；
注意：p必须是一个16-bit对齐的一个变量的地址。
所以说，这个指令就是把xmmword_413E34赋值给v5，那么只要找这个东西是啥就可以了。

```
.rdata:00413E33                 align 4
.rdata:00413E34 xmmword_413E34  xmmword 3074656D30633165577B465443545544h
.rdata:00413E34                                         ; DATA XREF: _main+10↑r
.rdata:00413E44 qword_413E44    dq 7D465443545544h      ; DATA XREF: _main+27↑r
.rdata:00413E4C ; char aDutctf[]
.rdata:00413E4C aDutctf         db '欢迎来到DUTCTF呦',0Ah,0  ; DATA XREF: _main+1A↑o
.rdata:00413E5E                 align 10h
```

网上：
返回伪代码我们看到第12行，xmmword的数据通过xmmword_413E34传递（_mm_storeu_si128存储，v4代表所指示的变量xmmword）。第14行，qword_413E44所指示的地址刚好在xmmword里（_mm_storel_epi64存储，v5代表所指向的指针，指针指向的地址刚好在xmmword里）。所以v4内容由xmmword_413E34和qword_413E44组成？？？？？
总之，把那俩通过R换：

```
.rdata:00413E34 xmmword_413E34  xmmword '0tem0c1eW{FTCTUD'
.rdata:00413E34                                         ; DATA XREF: _main+10↑r
.rdata:00413E44 qword_413E44    dq '}FTCTUD'            ; DATA XREF: _main+27↑r
```

反过来读就ok
010no-strings-attached
main函数反编译：

```
1int __cdecl main(int argc, const char **argv, const char **envp)
2{
3  setlocale(6, &locale);
4  banner();
5  prompt_authentication();
6  authenticate();
7  return 0;
8}
```

authenticate中文意思就是鉴定，因此：

```
1void authenticate()
2{
3  wchar_t ws[8192]; // [esp+1Ch] [ebp-800Ch]
4  wchar_t *s2; // [esp+801Ch] [ebp-Ch]
5
```

```
  s2 = (wchar_t *)decrypt(&s, &dword_8048A90);
  if ( fgetws(ws, 0x2000, stdin) )
  {
    ws[wcslen(ws) - 1] = 0;
    if ( !wcscmp(ws, s2) )
      wprintf(&unk_8048B44);
    else
      wprintf(&unk_8048BA4);
  }
  free(s2);
}
```


代码中比较了 ws与s2 两个字符串，匹配的情况下输出了取值unk_8048B44的值：

```
.rodata:08048B44 unk_8048B44     db  53h ; S
.rodata:08048B45                 db    0
.rodata:08048B46                 db    0
.rodata:08048B47                 db    0
.rodata:08048B48                 db  75h ; u
.rodata:08048B49                 db    0
.rodata:08048B4A                 db    0
.rodata:08048B4B                 db    0
.rodata:08048B4C                 db  63h ; c
.rodata:08048B4D                 db    0
.rodata:08048B4E                 db    0
.rodata:08048B4F                 db    0
.rodata:08048B50                 db  63h ; c
.rodata:08048B51                 db    0
.rodata:08048B52                 db    0
.rodata:08048B53                 db    0
.rodata:08048B54                 db  65h ; e
.rodata:08048B55                 db    0
.rodata:08048B56                 db    0
.rodata:08048B57                 db    0
.rodata:08048B58                 db  73h ; s
.rodata:08048B59                 db    0
.rodata:08048B5A                 db    0
.rodata:08048B5B                 db    0
.rodata:08048B5C                 db  73h ; s
.rodata:08048B5D                 db    0
.rodata:08048B5E                 db    0
.rodata:08048B5F                 db    0
.rodata:08048B60                 db  21h ; !
.rodata:08048B61                 db    0
.rodata:08048B62                 db    0
.rodata:08048B63                 db    0
```


后面咋都是success！（未完。。。）