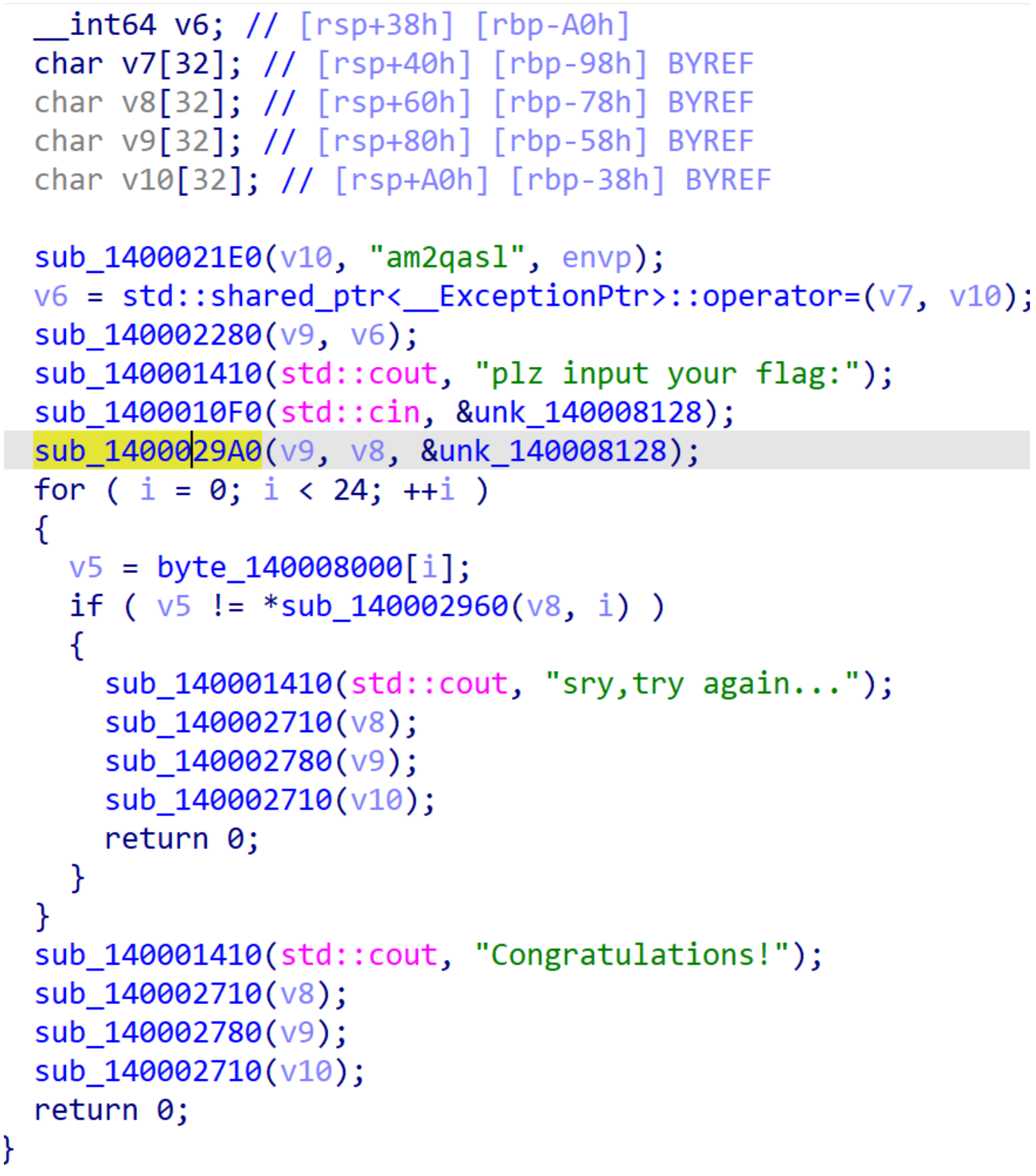
week4

reverse

change

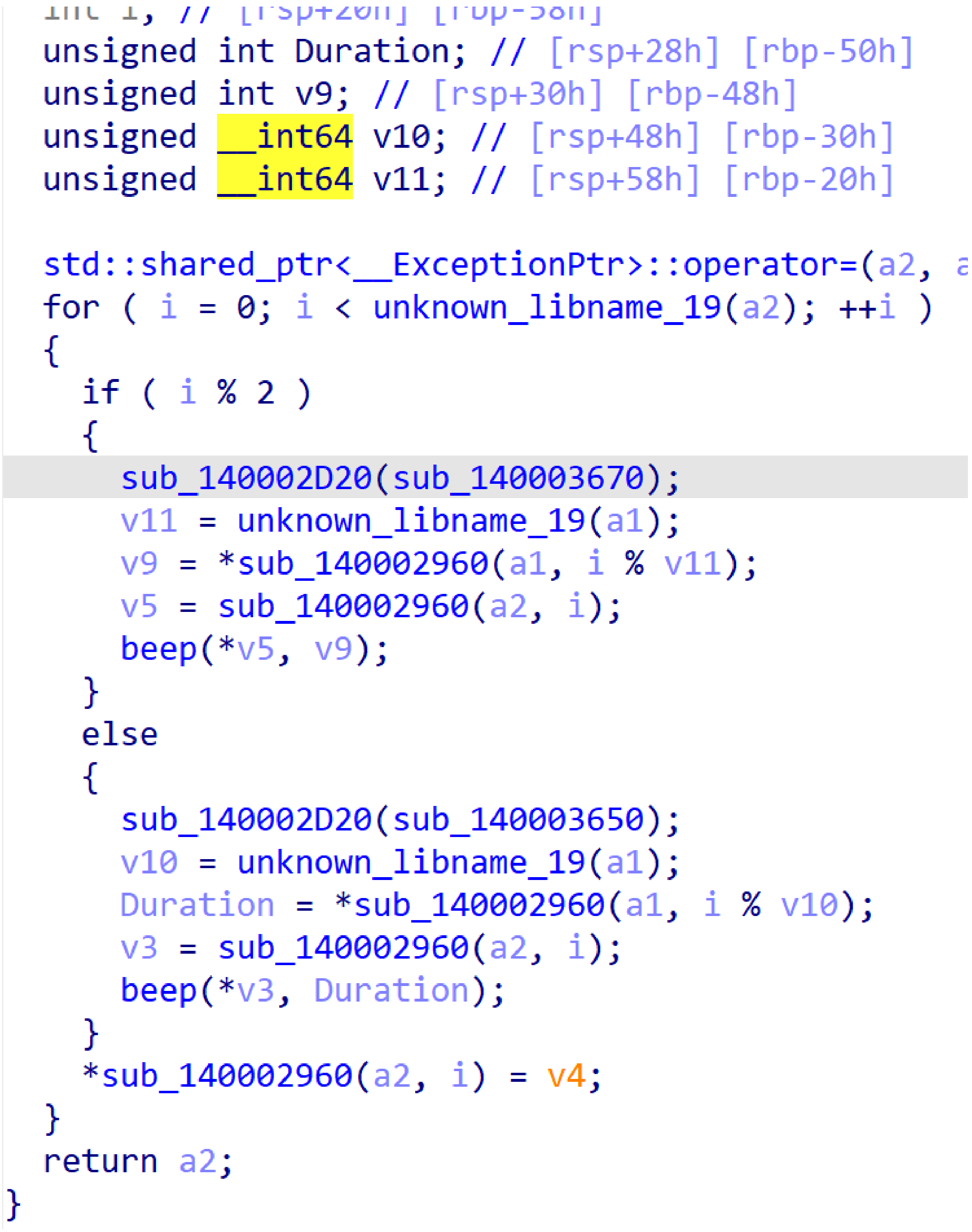
IDA打开



```
  __int64 v6; // [rsp+38h] [rbp-A0h]
  char v7[32]; // [rsp+40h] [rbp-98h] BYREF
  char v8[32]; // [rsp+60h] [rbp-78h] BYREF
  char v9[32]; // [rsp+80h] [rbp-58h] BYREF
  char v10[32]; // [rsp+A0h] [rbp-38h] BYREF

  sub_1400021E0(v10, "am2qasl", envp);
  v6 = std::shared_ptr<__ExceptionPtr>::operator=(v7, v10);
  sub_140002280(v9, v6);
  sub_140001410(std::cout, "plz input your flag:");
  sub_1400010F0(std::cin, &unk_140008128);
  sub_1400029A0(v9, v8, &unk_140008128);
  for ( i = 0; i < 24; ++i )
  {
    v5 = byte_140008000[i];
    if ( v5 != *sub_140002960(v8, i) )
    {
      sub_140001410(std::cout, "sry,try again...");
      sub_140002710(v8);
      sub_140002780(v9);
      sub_140002710(v10);
      return 0;
    }
  }
  sub_140001410(std::cout, "Congratulations!");
  sub_140002710(v8);
  sub_140002780(v9);
  sub_140002710(v10);
  return 0;
}
```

一段输入后，被`sub_1400029A0`函数加密，随后与密文比较

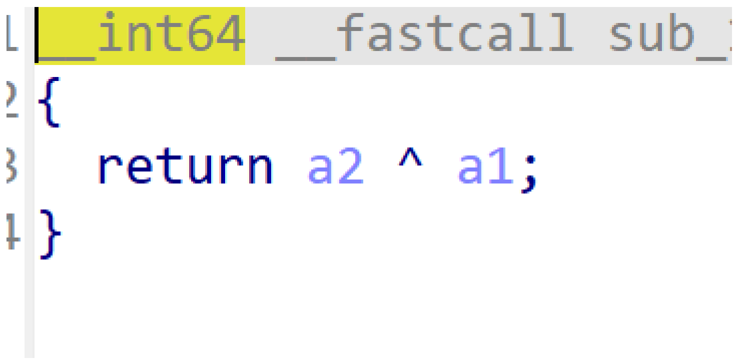


```
  int i; // [rsp+20h] [rbp-38h]
  unsigned int Duration; // [rsp+28h] [rbp-50h]
  unsigned int v9; // [rsp+30h] [rbp-48h]
  unsigned __int64 v10; // [rsp+48h] [rbp-30h]
  unsigned __int64 v11; // [rsp+58h] [rbp-20h]

  std::shared_ptr<__ExceptionPtr>::operator=(a2, a
  for ( i = 0; i < unknown_libname_19(a2); ++i )
  {
    if ( i % 2 )
    {
      sub_140002D20(sub_140003670);
      v11 = unknown_libname_19(a1);
      v9 = *sub_140002960(a1, i % v11);
      v5 = sub_140002960(a2, i);
      beep(*v5, v9);
    }
    else
    {
      sub_140002D20(sub_140003650);
      v10 = unknown_libname_19(a1);
      Duration = *sub_140002960(a1, i % v10);
      v3 = sub_140002960(a2, i);
      beep(*v3, Duration);
    }
    *sub_140002960(a2, i) = v4;
  }
  return a2;
}
```

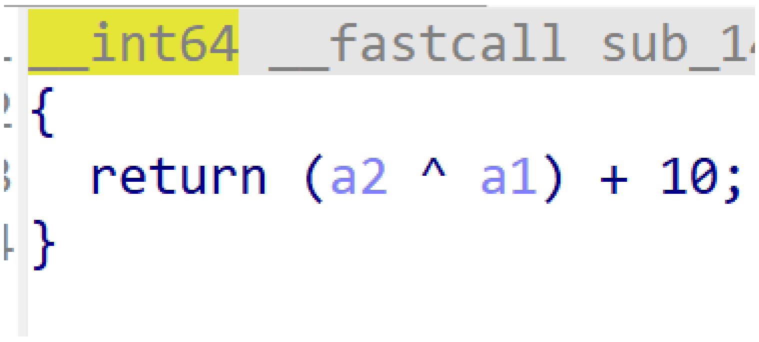
`sub_1400029A0` 函数就是根据i的奇偶进行不同的加密

若i为奇



```
__int64 __fastcall sub_
{
  return a2 ^ a1;
}
```

若i为偶



```
__int64 __fastcall sub_14
{
  return (a2 ^ a1) + 10;
}
```

这里a1和a2分别就是主函数中的key和flag

exp：

```
v = [0x13, 0x0A, 0x5D, 0x1C, 0x0E, 0x08, 0x23, 0x06, 0x0B, 0x4B,
     0x38, 0x22, 0x0D, 0x1C, 0x48, 0x0C, 0x66, 0x15, 0x48, 0x1B,
     0x0D, 0x0E, 0x10, 0x4F]
s = "am2qasl"
for i in range(len(v)):
    if i % 2 == 0:
        print(chr(ord(s[i % 7]) ^ (v[i] - 10)), end='')
    else:
        print(chr(ord(s[i % 7]) ^ (v[i])), end='')
```

again!

python逆向，解包后反编译不了

直接查看字节码

```
import dis,marshal
f=open("bin1.pyc", "rb").read()
code=marshal.loads(f[16:])
dis.dis(code)
```

```
  0           0 RESUME                   0

  1           2 LOAD_CONST               0 (0)
              4 LOAD_CONST               1 (None)
              6 IMPORT_NAME              0 (hashlib)
              8 STORE_NAME               0 (hashlib)

  2          10 PUSH_NULL
             12 LOAD_NAME                1 (print)
             14 LOAD_CONST               2 ('you should use this execute file to
decrypt "bin2"')
             16 PRECALL                  1
             20 CALL                     1
             30 POP_TOP

  3          32 PUSH_NULL
             34 LOAD_NAME                1 (print)
             36 LOAD_CONST               3 ('hint:md5')
             38 PRECALL                  1
             42 CALL                     1
             52 POP_TOP

  4          54 PUSH_NULL
             56 LOAD_NAME                2 (bytearray)
             58 PRECALL                  0
             62 CALL                     0
             72 STORE_NAME               3 (s)

  5          74 PUSH_NULL
             76 LOAD_NAME                2 (bytearray)
             78 PUSH_NULL
             80 LOAD_NAME                4 (open)
             82 LOAD_CONST               4 ('bin1.pyc')
             84 LOAD_CONST               5 ('rb')
             86 PRECALL                  2
             90 CALL                     2
            100 LOAD_METHOD              5 (read)
            122 PRECALL                  0
            126 CALL                     0
            136 PRECALL                  1
            140 CALL                     1
            150 STORE_NAME               6 (f)

  6         152 LOAD_CONST               6 ('jkasnwojasd')
            154 STORE_NAME               7 (t)

  8         156 PUSH_NULL
            158 LOAD_NAME                8 (range)
            160 LOAD_CONST               0 (0)
            162 LOAD_CONST               7 (15)
            164 PRECALL                  2
            168 CALL                     2
            178 GET_ITER
        >>  180 FOR_ITER               106 (to 394)
            182 STORE_NAME               9 (i)

  9         184 LOAD_NAME                6 (f)
            186 LOAD_NAME                9 (i)
            188 BINARY_SUBSCR
            198 LOAD_NAME                6 (f)
            200 LOAD_NAME                9 (i)
            202 LOAD_CONST               8 (6)
            204 BINARY_OP                6 (%)
            208 BINARY_SUBSCR
            218 BINARY_OP                0 (+)
            222 PUSH_NULL
            224 LOAD_NAME               10 (ord)
            226 LOAD_NAME                7 (t)
            228 LOAD_NAME                9 (i)
            230 LOAD_CONST               8 (6)
            232 BINARY_OP                6 (%)
            236 BINARY_SUBSCR
            246 PRECALL                  1
            250 CALL                     1
            260 PUSH_NULL
            262 LOAD_NAME               10 (ord)
            264 LOAD_NAME                7 (t)
            266 LOAD_NAME                9 (i)
            268 PUSH_NULL
            270 LOAD_NAME               11 (len)
            272 LOAD_NAME                7 (t)
            274 PRECALL                  1
            278 CALL                     1
            288 BINARY_OP                6 (%)
            292 BINARY_SUBSCR
            302 PRECALL                  1
            306 CALL                     1
            316 BINARY_OP                0 (+)
            320 BINARY_OP               12 (^)
            324 LOAD_CONST               9 (256)
            326 BINARY_OP                6 (%)
            330 LOAD_NAME                6 (f)
            332 LOAD_NAME                9 (i)
            334 STORE_SUBSCR

 10         338 LOAD_NAME                3 (s)
            340 LOAD_METHOD             12 (append)
            362 LOAD_NAME                6 (f)
            364 LOAD_NAME                9 (i)
            366 BINARY_SUBSCR
            376 PRECALL                  1
            380 CALL                     1
            390 POP_TOP
            392 JUMP_BACKWARD          107 (to 180)

 12     >>  394 PUSH_NULL
            396 LOAD_NAME                1 (print)
            398 LOAD_NAME                3 (s)
            400 PRECALL                  1
            404 CALL                     1
            414 POP_TOP

 13         416 PUSH_NULL
            418 LOAD_NAME                0 (hashlib)
            420 LOAD_ATTR               13 (md5)
            430 PUSH_NULL
            432 LOAD_NAME               14 (bytes)
            434 LOAD_NAME                3 (s)
            436 PRECALL                  1
            440 CALL                     1
            450 PRECALL                  1
            454 CALL                     1
            464 LOAD_METHOD             15 (hexdigest)
            486 PRECALL                  0
            490 CALL                     0
            500 STORE_NAME              16 (md5_hash)
            502 LOAD_CONST               1 (None)
            504 RETURN_VALUE
```

翻译后如下

```
import hashlib

print('you should use this execute file to decrypt "bin2"')
```

```python
print('hint:md5')

s = bytearray()
f = bytearray(open('bin1.pyc', 'rb').read())
t = 'jkasnwojasd'

for i in range(0, 15):
    f[i] = ((f[i] + f[i % 6]) ^ (ord(t[i % 6]) + ord(t[i % len(t)]))) % 256
    s.append(f[i])
print(s)
md5_hash = hashlib.md5(bytes(s)).hexdigest()
print(md5_hash)
```

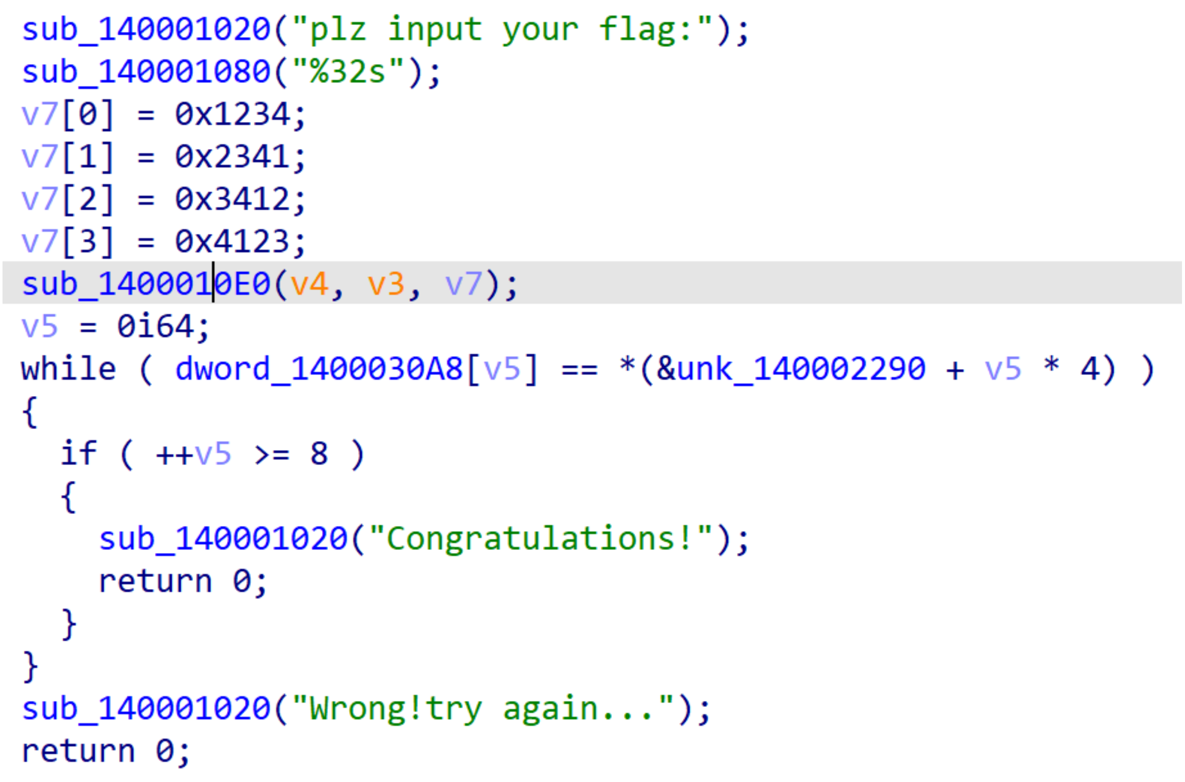
这里执行这个脚本后会得到一个md5

将这串md5与bin2异或后得到MZ头，那么就是另一个exe

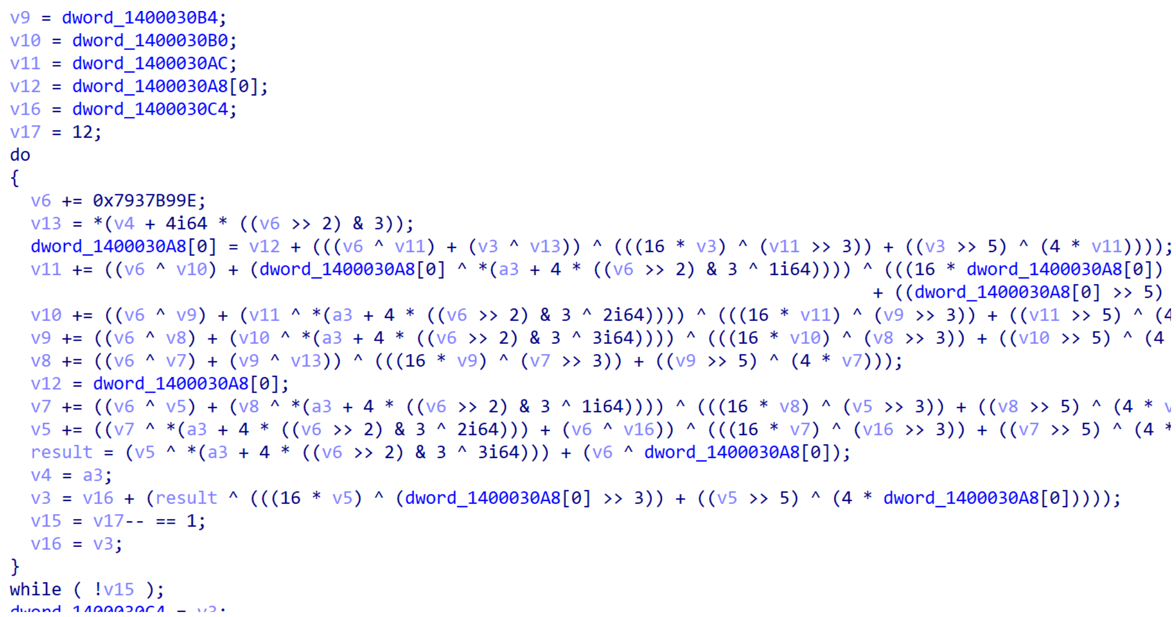
```python
t = 'a405b5d321e446459d8f9169d027bd92'
with open('bin2', 'rb') as f:
    data = bytearray(f.read())

result = bytearray(b ^ ord(t[i % len(t)]) for i, b in enumerate(data))

with open('output.exe', 'wb') as f:
    f.write(result)
```



```c
sub_140001020("plz input your flag:");
sub_140001080("%32s");
v7[0] = 0x1234;
v7[1] = 0x2341;
v7[2] = 0x3412;
v7[3] = 0x4123;
sub_1400010E0(v4, v3, v7);
v5 = 0i64;
while ( dword_1400030A8[v5] == *(&unk_140002290 + v5 * 4) )
{
  if ( ++v5 >= 8 )
  {
    sub_140001020("Congratulations!");
    return 0;
  }
}
sub_140001020("Wrong!try again...");
return 0;
```



```
v9 = dword_1400030B4;
v10 = dword_1400030B0;
v11 = dword_1400030AC;
v12 = dword_1400030A8[0];
v16 = dword_1400030C4;
v17 = 12;
do
{
  v6 += 0x7937B99E;
  v13 = *(v4 + 4i64 * ((v6 >> 2) & 3));
  dword_1400030A8[0] = v12 + (((v6 ^ v11) + (v3 ^ v13)) ^ (((16 * v3) ^ (v11 >> 3)) + ((v3 >> 5) ^ (4 * v11))));
  v11 += ((v6 ^ v10) + (dword_1400030A8[0] ^ *(a3 + 4 * ((v6 >> 2) & 3 ^ 1i64)))) ^ (((16 * dword_1400030A8[0])
                                                                              + ((dword_1400030A8[0] >> 5)
  v10 += ((v6 ^ v9) + (v11 ^ *(a3 + 4 * ((v6 >> 2) & 3 ^ 2i64)))) ^ (((16 * v11) ^ (v9 >> 3)) + ((v11 >> 5) ^ (4
  v9 += ((v6 ^ v8) + (v10 ^ *(a3 + 4 * ((v6 >> 2) & 3 ^ 3i64)))) ^ (((16 * v10) ^ (v8 >> 3)) + ((v10 >> 5) ^ (4
  v8 += ((v6 ^ v7) + (v9 ^ v13)) ^ (((16 * v9) ^ (v7 >> 3)) + ((v9 >> 5) ^ (4 * v7)));
  v12 = dword_1400030A8[0];
  v7 += ((v6 ^ v5) + (v8 ^ *(a3 + 4 * ((v6 >> 2) & 3 ^ 1i64)))) ^ (((16 * v8) ^ (v5 >> 3)) + ((v8 >> 5) ^ (4 *
  v5 += ((v7 ^ *(a3 + 4 * ((v6 >> 2) & 3 ^ 2i64))) + (v6 ^ v16)) ^ (((16 * v7) ^ (v16 >> 3)) + ((v7 >> 5) ^ (4 *
  result = (v5 ^ *(a3 + 4 * ((v6 >> 2) & 3 ^ 3i64))) + (v6 ^ dword_1400030A8[0]);
  v4 = a3;
  v3 = v16 + (result ^ (((16 * v5) ^ (dword_1400030A8[0] >> 3)) + ((v5 >> 5) ^ (4 * dword_1400030A8[0]))));
  v15 = v17-- == 1;
  v16 = v3;
}
while ( !v15 );
```

一个xxtea，直接上经典解密脚本

exp：

```
#include <stdio.h>
#include <stdint.h>
#include<stdlib.h>
#define DELTA 0x7937B99E
#define MX ((sum ^ y) + (k[(p&3)^e] ^ z)) ^ (((16 * z) ^ (y >> 3)) + ((z >> 5) ^
(4 * y)))
void decrypt(uint32_t* v, int n, uint32_t* k) {
    uint32_t y, z, sum;
    unsigned p, rounds, e;
    rounds = 12;
    sum = rounds*DELTA;
    y = v[0];
    do
    {
        e = (sum >> 2) & 3;
            for (p=n-1; p>0; p--)
            {
                z = v[p-1];
                y = v[p] -= MX;
            }
            z = v[n-1];
            y = v[0] -= MX;
            sum -= DELTA;
    } while (--rounds);


}

int main()
{
    uint32_t j,m;
    uint8_t v[33] = { 0xC3, 0xB5, 0x6F, 0x50, 0x45, 0x8F, 0x35, 0xB9, 0xC7,
0xE8,
  0x1A, 0xC9, 0x80, 0xE2, 0x20, 0x38, 0x83, 0xBA, 0x3A, 0xD1,
  0x54, 0xF5, 0x5C, 0x97, 0x6B, 0x03, 0x52, 0x43, 0x47, 0x04,
  0xD2, 0x1C};
```

```c
    uint32_t k[4] = { 0x1234, 0x2341, 0x3412, 0x4123};
    decrypt(v, 8, k);
    printf(v);
    system("pause");
    return 0;
}
```