REVERSE

1. Change

直接进调试，输个 123456 进去，分析一下代码发现 v8 复制了输入然后和 v9 进行了一系列操作

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  int i; // [rsp+20h] [rbp-B8h]
  int v5; // [rsp+24h] [rbp-B4h]
  __int64 v6; // [rsp+38h] [rbp-A0h]
  char v7[32]; // [rsp+40h] [rbp-98h] BYREF
  __int64 v8[4]; // [rsp+60h] [rbp-78h] BYREF
  char v9[32]; // [rsp+80h] [rbp-58h] BYREF
  char v10[32]; // [rsp+A0h] [rbp-38h] BYREF

  sub_7FF70B7621E0((__int64)v10, (__int64)"am2qasl");
  v6 = std::shared_ptr<__ExceptionPtr>::operator=(v7, v10);
  sub_7FF70B762280(v9, v6);
  sub_7FF70B761410(std::cout, "plz input your flag:");
  sub_7FF70B7610F0(std::cin, &unk_7FF70B768128);
  sub_7FF70B7629A0((__int64)v9, (__int64)v8, (__int64)&unk_7FF70B768128);
  for ( i = 0; i < 24; ++i )
  {
    v5 = byte_7FF70B768000[i];
    if ( v5 != *(char *)sub_7FF70B762960(v8, i) )
    {
      sub_7FF70B761410(std::cout, "sry,try again...");
      sub_7FF70B762710((__int64)v8);
      sub_7FF70B762780(v9);
      sub_7FF70B762710((__int64)v10);
```

```
std::shared_ptr<__ExceptionPtr>::operator=(a2, a3);
for ( i = 0; i < (unsigned __int64)unknown_libname_19(a2); ++i )
{
  if ( i % 2 )
  {
    sub_7FF70B762D20(sub_7FF70B763670);
    v11 = unknown_libname_19(a1);
    v9 = *(char *)sub_7FF70B762960(a1, i % v11);
    v5 = (char *)sub_7FF70B762960(a2, i);
    beep(*v5, v9);
  }
  else
  {
    sub_7FF70B762D20(sub_7FF70B763650);
    v10 = unknown_libname_19(a1);
    Duration = *(char *)sub_7FF70B762960(a1, i % v10);
    v3 = (char *)sub_7FF70B762960(a2, i);
    beep(*v3, Duration);
  }
  *(_BYTE *)sub_7FF70B762960(a2, i) = v4;
}
return a2;
```

这里看到对奇数偶数下标下的数据的操作不一样，先不管放到一边，单步调试到进行对比的地方

```
int i; // [rsp+20h] [rbp-B8h]
int v5; // [rsp+24h] [rbp-B4h]
__int64 v6; // [rsp+38h] [rbp-A0h]
char v7[32]; // [rsp+40h] [rbp-98h] BYREF
__int64 v8[4]; // [rsp+60h] [rbp-78h] BYREF
char v9[32]; // [rsp+80h] [rbp-58h] BYREF
char v10[32]; // [rsp+A0h] [rbp-38h] BYREF

sub_7FF70B7621E0((__int64)v10, (__int64)"am2qasl");
v6 = std::shared_ptr<__ExceptionPtr>::operator=(v7, v10);
sub_7FF70B762280(v9, v6);
sub_7FF70B761410(std::cout, "plz input your flag:");
sub_7FF70B7610F0(std::cin, &unk_7FF70B768128);
sub_7FF70B7629A0((__int64)v9, (__int64)v8, (__int64)&unk_7FF70B768128);
for ( i = 0; i < 24; ++i )
{
  v5 = byte_7FF70B768000[i];
  if ( v5 != *(char *)sub_7FF70B762960(v8, i) )
  {
    sub_7FF70B761410(std::cout, "sry,try again...");
    sub_7FF70B762710((__int64)v8);
    sub_7FF70B762780(v9);
```

点进去看 v8

```
C22A17FADD db    0
C22A17FADE db    0
C22A17FADF db    0
C22A17FAE0 db  5Ah ; Z
C22A17FAE1 db  5Fh ; _
C22A17FAE2 db  0Bh
C22A17FAE3 db  45h ; E
C22A17FAE4 db  5Eh ; ^
C22A17FAE5 db  45h ; E
C22A17FAE6 db    0
C22A17FAE7 db    0
C22A17FAE8 db    0
C22A17FAE9 db    0
C22A17FAEA db    0
C22A17FAEB db    0
```

Ok 开始猜，拿之前的密钥和 123456 直接玩一次异或看看会发生什么

```
:k[000020D4]:000000C22A17FADC db    0
:k[000020D4]:000000C22A17FADD db    0
:k[000020D4]:000000C22A17FADE db    0
:k[000020D4]:000000C22A17FADF db    0
:k[000020D4]:000000C22A17FAE0 db  5Ah ; Z
:k[000020D4]:000000C22A17FAE1 db  5Fh ; _
:k[000020D4]:000000C22A17FAE2 db  0Bh
:k[000020D4]:000000C22A17FAE3 db  45h ; E
:k[000020D4]:000000C22A17FAE4 db  5Eh ; ^
:k[000020D4]:000000C22A17FAE5 db  45h ; E
:k[000020D4]:000000C22A17FAE6 db    0
:k[000020D4]:000000C22A17FAE7 db    0
:k[000020D4]:000000C22A17FAE8 db    0
:k[000020D4]:000000C22A17FAE9 db    0
:k[000020D4]:000000C22A17FAEA db    0
```

```
Microsoft Visual Studio 调试控制台
505F014554456C
D:\code_c++\change\x64\Debug\change.exe (进程 6300)已
要在调试停止时自动关闭控制台，请启用“工具”->“选项”
按任意键关闭此窗口. . .
```

```
#define _CRT_SECURE_NO_WARNINGS 1
#include<stdio.h>
int main()
{
    char ch[] = { "123456" };
    char key[] = { "am2qas1" };
    for (int i = 0; i < sizeof(ch); i++)
    {
        printf("%02X", key[i % 7] ^ ch[i]);
    }
```

这里可以看到，偶数位的都对应上了，奇数位的都相差 10，这刚好对应上了我之前看到的奇数偶数位的操作不一样，所以改一下

```
Stack[000020D4]:000000C22A17FADC db    0
Stack[000020D4]:000000C22A17FADD db    0
Stack[000020D4]:000000C22A17FADE db    0
Stack[000020D4]:000000C22A17FADF db    0
Stack[000020D4]:000000C22A17FAE0 db  5Ah ; Z
Stack[000020D4]:000000C22A17FAE1 db  5Fh ; _
Stack[000020D4]:000000C22A17FAE2 db  0Bh
Stack[000020D4]:000000C22A17FAE3 db  45h ; E
Stack[000020D4]:000000C22A17FAE4 db  5Eh ; ^
Stack[000020D4]:000000C22A17FAE5 db  45h ; E
Stack[000020D4]:000000C22A17FAE6 db    0
Stack[000020D4]:000000C22A17FAE7 db    0
Stack[000020D4]:000000C22A17FAE8 db    0
Stack[000020D4]:000000C22A17FAE9 db    0
Stack[000020D4]:000000C22A17FAEA db    0
```

```
Microsoft Visual Studio 调试控制台
5A5F0B455E4576
D:\code_c++\change\x64\Debug\change.exe (进程 2808
要在调试停止时自动关闭控制台，请启用“工具”->“选
按任意键关闭此窗口. . .
```

```
#define _CRT_SECURE_NO_WARNINGS 1
#include<stdio.h>
int main()
{
    char ch[] = { "123456" };
    char key[] = { "am2qas1" };
    for (int i = 0; i < sizeof(ch); i++)
    {
        if (i % 2 == 0)
            printf("%02X", (key[i % 7] ^ ch[i]) + 10);
        else
            printf("%02X", key[i % 7] ^ ch[i]);
```

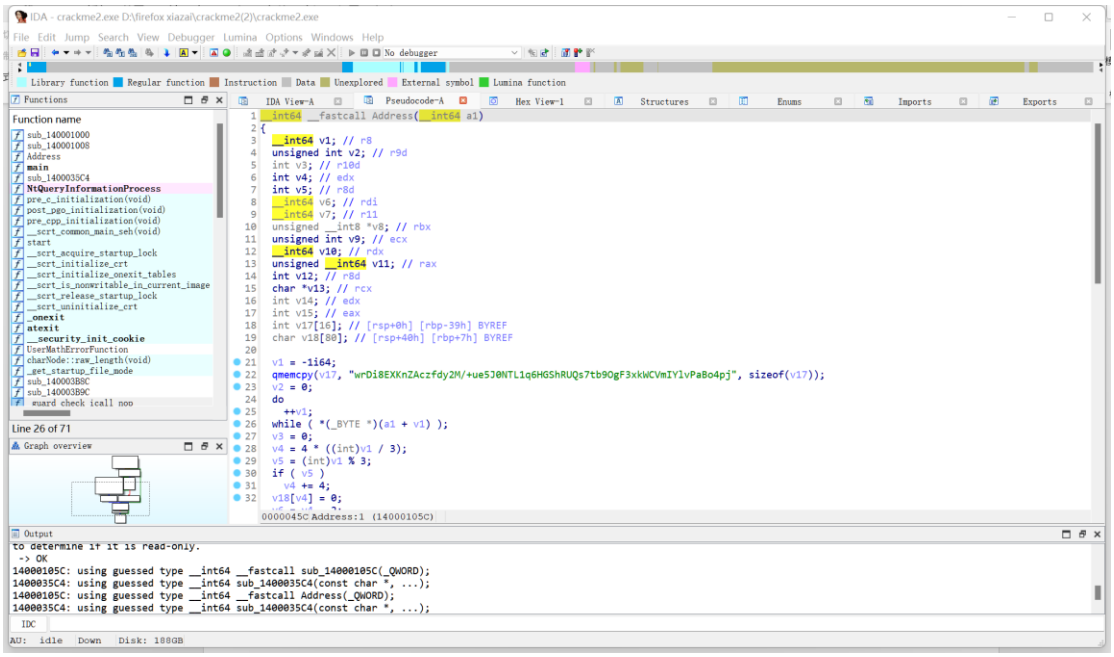
这一次对了，也就是说对输入的操作就是我猜的这样（后面多试了几次发现都是这样）然后继续进行单步调试，发现后续没有再对输入进行修改，就可以直接写脚本解密了

```
#define _CRT_SECURE_NO_WARNINGS 1


#include <stdio.h>
#include<string.h>

int main()
{

    unsigned int ch[] = { 0x13, 0x0A, 0x5D, 0x1C, 0x0E, 0x08, 0x23, 0x06, 0x0B, 0x4B,
  0x38, 0x22, 0x0D, 0x1C, 0x48, 0x0C, 0x66, 0x15, 0x48, 0x1B,
  0x0D, 0x0E, 0x10, 0x4F, 0xFF, 0xFF, 0xFF, 0xFF, 0x00, 0x00,
  0x00, 0x00 };
    char key[] = { "am2qasl" };
    for (int i = 0; i < 24; i++)
    {
        if (i % 2 == 0)
            printf("%C", key[i % 7] ^ ( ch[i] -10));
        else
            printf("%C", key[i % 7] ^ ch[i]);

    }
```

```
Microsoft Visual Studio 调试控制台
hgame{ugly_Cpp_and_hook}
D:\code_c++\测试项目\x64\Debug\测试项目.exe (进程
要在调试停止时自动关闭控制台，请启用“工具”->“
按任意键关闭此窗口. . .
```

2.Crackme2

首先反编译后发现有一个红的指令，目前还不知道是什么，还有一个函数，分析逻辑后发现是只要函数返回 1，就输出 right，点进去函数，发现是一个换表的 base64



拿着这个去解码看看

发现得到的是 fake flag。目前还没有头绪，进入调试输入后看程序会发生什么，发现触发了异常警告

```
DE loc_7FF65B0B34DE:                          ; DATA XREF: .rdata:00007FF65B0B5834↓o
DE ;   __try { // __except at loc_7FF65B0B34EB
DE mov     byte ptr ds:0, 1
E6 jmp     loc_7FF65B0B359B
E6 ;   } // starts at 7FF65B0B34DE
E6
EB ; ---------------------------------------------------------------------------
EB
EB loc_7FF65B0B34EB:                          ; DATA XREF: .rdata:00007FF65B0B5834↓o
EB ;   __except(7FF51B0B0001) // owned by 7FF65B0B34DE
EB call    cs:GetCurrentProcess
EB
F1 mov     rcx, rax                           ; ProcessHandle
F4 lea     rax, [rsp+78h+arg_10]
FC mov     [rsp+78h+ReturnLength], rax        ; ReturnLength
01 mov     r9d, 8                             ; ProcessInformationLength
07 lea     r8, [rsp+78h+ProcessInformation]   ; ProcessInformation
0F lea     edx, [r9-1]                        ; ProcessInformationClass
13 call    NtQueryInformationProcess
13
18 cmp     [rsp+78h+ProcessInformation], 0FFFFFFFFFFFFFFFFh
21 jz      short loc_7FF65B0B359B
21
23 lea     r9, [rsp+78h+flOldProtect]         ; lpflOldProtect
```

在这可以看到如果触发异常就会执行下面 loc_7FF65B0B34EB 处的代码，于是直接 nop 掉造成异常的指令

```
.text:00007FF65B0B34D8
.text:00007FF65B0B34DD nop
.text:00007FF65B0B34DD
.text:00007FF65B0B34DE
.text:00007FF65B0B34DE loc_7FF65B0B34DE:                          ; DATA XREF: .rdata:00007FF65B0B5834↓o
.text:00007FF65B0B34DE ;   __try { // __except at loc_7FF65B0B34EB
.text:00007FF65B0B34DE nop
.text:00007FF65B0B34DF nop
.text:00007FF65B0B34E0 nop
.text:00007FF65B0B34E1 nop
.text:00007FF65B0B34E2 nop
.text:00007FF65B0B34E3 nop
.text:00007FF65B0B34E4 nop
.text:00007FF65B0B34E5 nop
.text:00007FF65B0B34E6 nop
.text:00007FF65B0B34E7 nop
.text:00007FF65B0B34E8 nop
.text:00007FF65B0B34E9 nop
.text:00007FF65B0B34EA nop
.text:00007FF65B0B34EA ;   } // starts at 7FF65B0B34DE
.text:00007FF65B0B34EA
.text:00007FF65B0B34EB
.text:00007FF65B0B34EB loc_7FF65B0B34EB:                          ; DATA XREF: .rdata:00007FF65B0B5834↓o
.text:00007FF65B0B34EB ;   __except(7FF51B0B0001) // owned by 7FF65B0B34DE
.text:00007FF65B0B34EB call    cs:GetCurrentProcess
```

这个时候再 f5 反编译，就能发现代码发生了改变

```
IDA View-RIP                                    Pseudocode-B                                    Pseudocode-A
 9  DWORD flOldProtect; // [rsp+80h] [rbp+8h] BYREF
10  __int64 ProcessInformation; // [rsp+88h] [rbp+10h] BYREF
11  ULONG ReturnLength; // [rsp+90h] [rbp+18h] BYREF
12
13  sub_7FF65B0B35C4("%50s", v9);
14  CurrentProcess = GetCurrentProcess();
15  NtQueryInformationProcess(CurrentProcess, ProcessDebugPort, &ProcessInformation, 8u, &ReturnLength);
16  if ( ProcessInformation != -1 )
17  {
18    VirtualProtect(Address, 0x6000ui64, 0x40u, &flOldProtect);
19    v4 = 0;
20    v5 = 0i64;
21    do
22    {
23      *((_BYTE *)Address + v5) ^= byte_7FF65B0B6000[v5];
24      ++v4;
25      ++v5;
26    }
27    while ( (unsigned __int64)v4 < 0x246A );
28    VirtualProtect(Address, 0x6000ui64, flOldProtect, &flOldProtect);
29  }
30  v6 = Address((__int64)v9);
31  v7 = "right flag!";
32  if ( !v6 )
33    v7 = "wrong flag!";
```

在这里就可以发现，程序运行的过程中有自修改的过程，于是继续运

行，看修改后的代码，发现有问题不能直接看，于是 undefined 代码然后在最开始的地方按 p 创造函数得到修改后的代码

```
num13 = a1[13];
num10 = a1[10];
num5 = a1[5];
num4 = a1[4];
num15 = a1[15];
num17 = a1[17];
num6 = a1[6];
num9 = a1[9];
if ( num9
   + 201 * num4
   + 194 * num3
   + 142 * num15
   + 114 * num17
   + 103 * num2
   + 52 * (num26 + num13)
   + ((num12 + num16) << 6)
   + 14 * (num19 + 4 * num27 + num27)
   + 9 * (num7 + 23 * num11 + num21 + 3 * num25 + 4 * num21 + 4 * num23)
   + 5 * (num24 + 23 * num14 + 2 * (num31 + 2 * num30) + 5 * num0 + 39 * num18 + 51 * num29)
   + 24 * (num28 + 10 * num1 + 4 * (num22 + num8 + 2 * num20))
   + 62 * num6
   + 211 * num5
   + 212 * num10 != 296473 )
  return 0i64;
v38 = 2 * num24;
if ( 207 * num5
   + 195 * num6
   + 151 * num7
   + 57 * num0
   + 118 * num23
00000614 sub_7FF6F21F105C:84 (7FF6F21F1214)
```

代码很长，是一大堆约束条件，思考以后发现应该用 z3 求解，于是用 py 写 z3 得到数据后，就解出了 flag

```
            + 58 * num3
            + 108 * num15
            + 142 * num9
            + 156 * (num12 + num27)
            + 16 * (num10 + 6 * num13)
            + 126 * (num26 + 2 * num17)
            + 127 * (num29 + 2 * num11 + num7)
            + 49 * (num14 + 4 * num24)
            + 11 * (num0 + 22 * num2)
            + 5 * (num18 + num22 + 45 * num4 + 50 * num1)
            + 109 * num21
            + 124 * num23
            + 123 * num31 == 418697)



# 求解约束
if solver.check() == sat:
    model = solver.model()
    print("Solution found:")
    print(model[num0])
```

问题　输出　调试控制台　终端　端口

```
m4.py"
Solution found:
104
103
97
109
```

转到行/列

```c
#include <stdio.h>
#include<string.h>

#include <stdio.h>
#define SIZE 10
#define PAR 72
int main(void) {

    char ch[] = { 104,
103,
97,
109,
101,
123,
83,
77,
67,
95,
52,
110,
100,
95,
115,
48,
108,
118,
49,
110,
103,
95,
101,
113,
117,
52,
116,
49,
79,
110,
115,
125 };
    for (int i = 0; i < sizeof(ch); i++)
        printf("%c", ch[i]);
```

```
Microsoft Visual Studio 调试控制台
hgame{SMC_4nd_s0lv1ng_equ4t10ns}
D:\code_c++\测试项目\x64\Debug\测试项目.exe (进程 20224)已退出，代码为 0。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控
按任意键关闭此窗口. . .
```

3.again

拿到附件，解包完后尝试用 pycdc 反编译，但发现编译不完全

```
D:\pyc文件反编译器 以及相关工具>pycdc.exe bin1.pyc
# Source Generated with Decompyle++
# File: bin1.pyc (Python 3.11)

Unsupported opcode: BINARY_OP
import hashlib
print('you should use this execute file to decrypt "bin2"')
print('hint:md5')
s = bytearray()
f = bytearray(open('bin1.pyc', 'rb').read())
t = 'jkasnwojasd'
# WARNING: Decompyle incomplete

D:\pyc文件反编译器 以及相关工具>
```

这里怀疑是花指令，尝试直接通过 pyc 文件拿到字节码

```
Anaconda Prompt
(env1) C:\Users\Lenovo>python C:\Users\Lenovo\Desktop\Untitled 1.py
  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)
```

找不到有花指令的地方，尝试手搓，写完运行程序后拿到 md5 值

```
PS D:\> & d:/.venv/Scripts/python.exe c:/Users/Lenovo/Desktop/test.py
you should use this execute file to decrypt "bin2"
hint:md5
a405b5d321e446459d8f9169b027bd92
PS D:\>
```

根据题目所示，要求是解密 bin2 文件，也就是说 bin1 经过操作后得到的 md5 能解密 bin2，尝试了很多种可能最后发现是把这串 md5 与 bin2 逐字节异或，写个脚本实现

```python
with open("D:\\pyc文件反编译器 以及相关工具\\bin2", "rb") as file:
    realfile = file.read()
key = "a405b5d321e446459d8f9169b027bd92"
result = bytearray()
for i in range(len(realfile)):
    result.append(realfile[i] ^ ord(key[i % len(key)]))


with open("C:\\Users\\Lenovo\\Desktop\\out.txt", "wb") as output_file:
    output_file.write(result)
```

异或完发现 out 文件头就是 MZ，说明异或的做法没错，改成 exe 文件后拖入 ida 分析

```
int __cdecl main(int argc, const char **argv, const char **envp)
{
  __int64 v3; // rdx
  __int64 v4; // rcx
  __int64 v5; // rax
  int v7[6]; // [rsp+20h] [rbp-18h] BYREF

  sub_7FF744551020("plz input your flag:");
  sub_7FF744551080("%32s");
  v7[0] = 4660;
  v7[1] = 9025;
  v7[2] = 13330;
  v7[3] = 16675;
  sub_7FF7445510E0(v4, v3, (__int64)v7);
  v5 = 0i64;
  while ( str0[v5] == enc[v5] )
  {
    if ( ++v5 >= 8 )
    {
      sub_7FF744551020("Congratulations!");
      return 0;
    }
  }
  sub_7FF744551020("Wrong!try again...");
  return 0;
}
```

很明显加密函数在 sub_7FF7445510E0 里，点进去

```
unsigned int anothernum7; // [rsp+40h] [rbp+8h]
int v17; // [rsp+48h] [rbp+10h]

num7 = str7;
realkey = key;
num6 = str6;
delta = 0;
num5 = str5;
num4 = str4;
num3 = str3;
num2 = str2;
num1 = str1;
num0 = str0[0];
anothernum7 = str7;
v17 = 12;
do
{
  delta += 2033695134;
  v13 = *(_DWORD *)(realkey + 4i64 * ((delta >> 2) & 3));
  str0[0] = num0 + (((delta ^ num1) + (num7 ^ v13)) ^ (((16 * num7) ^ (num1 >> 3)) + ((num7 >> 5) ^ (4 * num1))));
  num1 += ((delta ^ num2) + (str0[0] ^ *(_DWORD *)(key + 4 * ((delta >> 2) & 3 ^ 1i64)))) ^ (((16 * str0[0]) ^ (num2 >> 3))
                                                                                          + (((unsigned int)str0[0] >> 5) ^ (4 * num2)));
  num2 += ((delta ^ num3) + (num1 ^ *(_DWORD *)(key + 4 * ((delta >> 2) & 3 ^ 2i64)))) ^ (((16 * num1) ^ (num3 >> 3))
                                                                                          + ((num1 >> 5) ^ (4 * num3)));
  num3 += ((delta ^ num4) + (num2 ^ *(_DWORD *)(key + 4 * ((delta >> 2) & 3 ^ 3i64)))) ^ (((16 * num2) ^ (num4 >> 3))
                                                                                          + ((num2 >> 5) ^ (4 * num4)));
  num4 += ((delta ^ num5) + (num3 ^ v13)) ^ (((16 * num3) ^ (num5 >> 3)) + ((num3 >> 5) ^ (4 * num5)));
  num0 = str0[0];
  num5 += ((delta ^ num6) + (num4 ^ *(_DWORD *)(key + 4 * ((delta >> 2) & 3 ^ 1i64)))) ^ (((16 * num4) ^ (num6 >> 3))
                                                                                          + ((num4 >> 5) ^ (4 * num6)));
  num6 += ((num5 ^ *(_DWORD *)(key + 4 * ((delta >> 2) & 3 ^ 2i64))) + (delta ^ anothernum7)) ^ (((16 * num5) ^ (anothernum7 >> 3))
                                                                                                + ((num5 >> 5) ^ (4 * anothernum7)));
  result = (num6 ^ *(_DWORD *)(key + 4 * ((delta >> 2) & 3 ^ 3i64))) + (delta ^ str0[0]);
  realkey = key;
  num7 = anothernum7 + (result ^ (((16 * num6) ^ ((unsigned int)str0[0] >> 3)) + ((num6 >> 5) ^ (4 * str0[0]))));
  v15 = v17-- == 1;
  anothernum7 = num7;
```

是魔改了 delta 的 xxtea 加密，写个脚本解密就行（其实还是不理解 key 的索引为啥会优化成那样，卡了很久）

脚本如下

```
#define _CRT_SECURE_NO_WARNINGS 1
#include<stdio.h>
#include<stdint.h>


int key[] = { 0x1234,0x2341,0x3412,0x4123 };
```

```
 unsigned char right[] =
{
  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


};

void jiemi(unsigned int* right, unsigned int* key)
{

    int round = 12;
  unsigned  int delta = 0x7937B99E;    unsigned int sum = delta * 12;// 0xAE9CB368 进调
试看



    do {
        right[7] -= ((sum ^ right[0]) + (right[6] ^ key[7&3^((sum >> 2) & 3)])) ^ (((16
* right[6]) ^ (right[0] >> 3)) + (((right[6] >> 5) ^ (4 * right[0]))));

        for (int i = 6; i > 0; i--)
        {
            right[i] -= ((sum ^ right[i + 1]) + (right[i - 1] ^ key[i & 3 ^ ((sum >> 2)
& 3)])) ^ (((16 * right[i - 1]) ^ (right[i + 1] >> 3)) + (((right[i - 1] >> 5) ^ (4 *
right[i + 1]))));
        }
        right[0] -= ((sum ^ right[1]) + (right[7] ^ key[0 & 3 ^ ((sum >> 2) & 3)])) ^
(((16 * right[7]) ^ (right[1] >> 3)) + (((right[7] >> 5) ^ (4 * right[1]))));
        sum -= delta;
        round--;
    } while (round);
}
int main()
{

    jiemi((unsigned int*)right, (unsigned int*)key);
    for (int i = 0; i < sizeof(right); i++)
        printf("%c", right[i]);
    return 0;
}
```

```
    0x1A, 0xC9, 0x80, 0xE2, 0x20, 0x38, 0x83, 0xBA, 0x3A, 0xD1,
    0x54, 0xF5, 0x5C, 0x97, 0x6B, 0x03, 0x52, 0x43, 0x47, 0x04,
    0xD2, 0x1C


};

void jiemi(unsigned int* right, unsigned int* key)
{

    int round = 12;
  unsigned  int delta = 0x7937B99E;    unsigned int sum = d


    do {
        right[7] -= ((sum ^ right[0]) + (right[6] ^ key[7&3

        for (int i = 6; i > 0; i--)
        {
            right[i] -= ((sum ^ right[i + 1]) + (right[i -
        }
        right[0] -= ((sum ^ right[1]) + (right[7] ^ key[0 &
        sum -= delta;
        round--;
```

```
hgame{btea_is_a_hard_encryption}
D:\code_c++\测试项目\x64\Debug\测试项目.exe (进程 2572
要在调试停止时自动关闭控制台，请启用“工具”->“选项”
按任意键关闭此窗口. . .
```