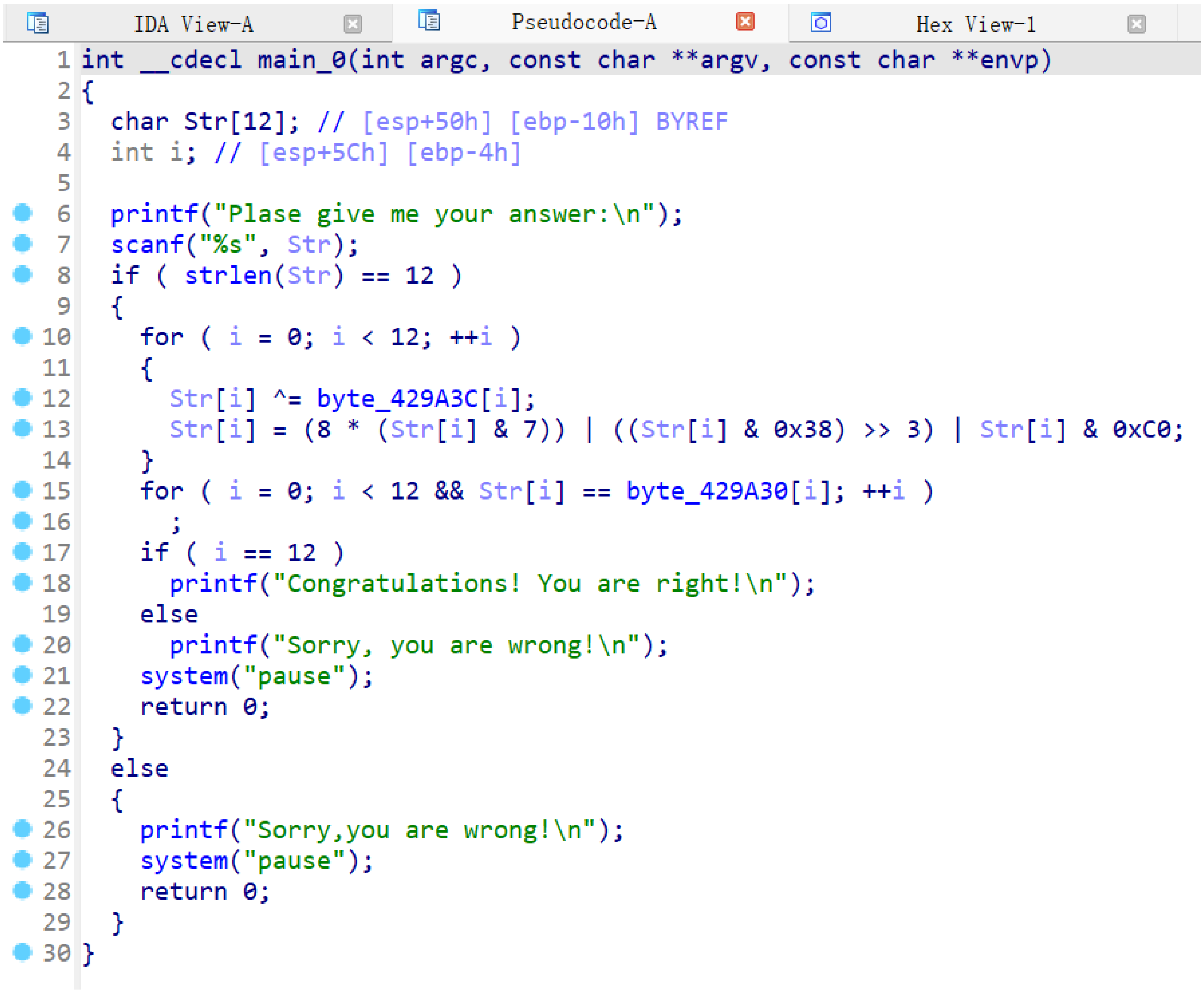
所有解密脚本都在相同目录下

CPP1

关键信息

1. 输入长度为12；
2. 使用 `byte_429A3C` 进行异或加密；
3. 位操作：后六位中的前后三位交换位置，剩下的前两位不变。



```
int __cdecl main_0(int argc, const char **argv, const char **envp)
{
  char Str[12]; // [esp+50h] [ebp-10h] BYREF
  int i; // [esp+5Ch] [ebp-4h]

  printf("Plase give me your answer:\n");
  scanf("%s", Str);
  if ( strlen(Str) == 12 )
  {
    for ( i = 0; i < 12; ++i )
    {
      Str[i] ^= byte_429A3C[i];
      Str[i] = (8 * (Str[i] & 7)) | ((Str[i] & 0x38) >> 3) | Str[i] & 0xC0;
    }
    for ( i = 0; i < 12 && Str[i] == byte_429A30[i]; ++i )
      ;
    if ( i == 12 )
      printf("Congratulations! You are right!\n");
    else
      printf("Sorry, you are wrong!\n");
    system("pause");
    return 0;
  }
  else
  {
    printf("Sorry,you are wrong!\n");
    system("pause");
    return 0;
  }
}
```

解决

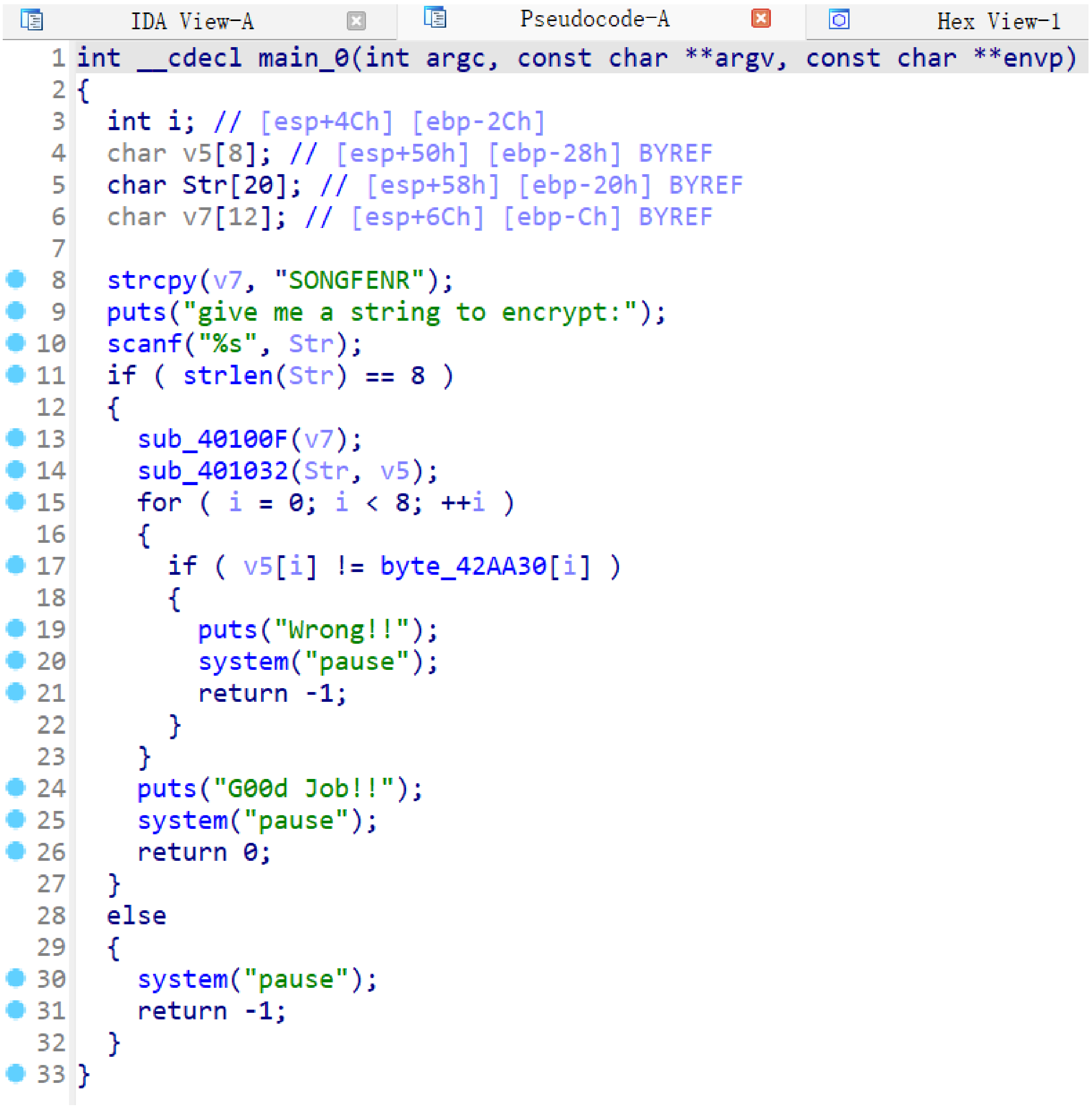
写解密脚本，先位操作再做异或。

flag: **SeventyYears**

CPP2

关键信息

1. `v7` 赋值 `SONGFENR`：根据长度判断大概率是DES密钥；
2. 输入长度为8：DES特征；
3. `sub_40100F`：参数 `v7`，密钥扩展，可以进入进一步看；
4. `sub_401032`：DES加密，可以进入进一步看；
5. `byte_42AA30`：存密文。



```
int __cdecl main_0(int argc, const char **argv, const char **envp)
{
  int i; // [esp+4Ch] [ebp-2Ch]
  char v5[8]; // [esp+50h] [ebp-28h] BYREF
  char Str[20]; // [esp+58h] [ebp-20h] BYREF
  char v7[12]; // [esp+6Ch] [ebp-Ch] BYREF

  strcpy(v7, "SONGFENR");
  puts("give me a string to encrypt:");
  scanf("%s", Str);
  if ( strlen(Str) == 8 )
  {
    sub_40100F(v7);
    sub_401032(Str, v5);
    for ( i = 0; i < 8; ++i )
    {
      if ( v5[i] != byte_42AA30[i] )
      {
        puts("Wrong!!");
        system("pause");
        return -1;
      }
    }
    puts("G00d Job!!");
    system("pause");
    return 0;
  }
  else
  {
    system("pause");
    return -1;
  }
}
```

解决

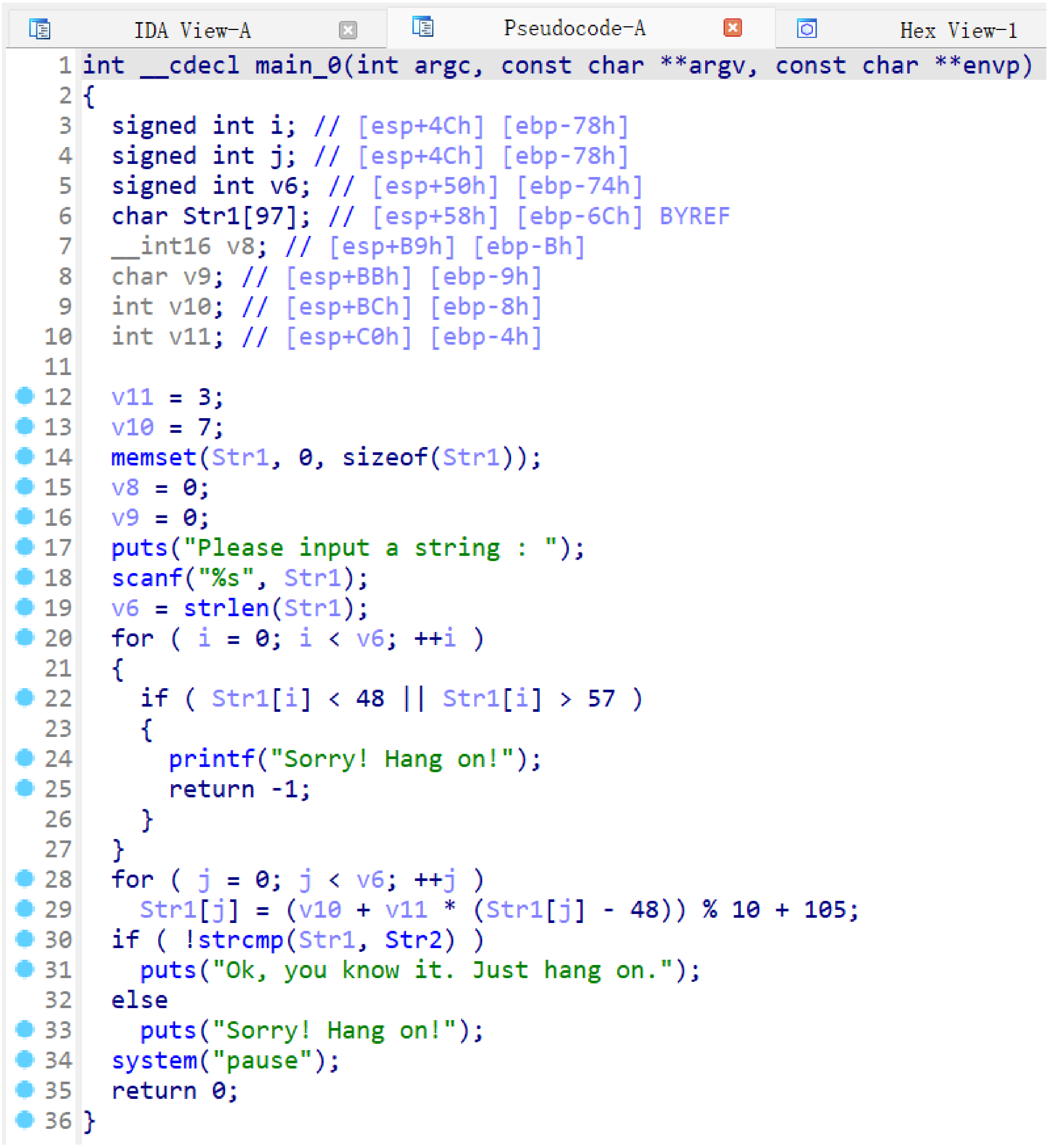
解密工具一把梭。

flag: **EZpoints**

CPP3

关键信息

1. 输入为数字；
2. 仿射加密；
3. 密文在 `Str2`。



IDA View-A | Pseudocode-A | Hex View-1

```
int __cdecl main_0(int argc, const char **argv, const char **envp)
{
  signed int i; // [esp+4Ch] [ebp-78h]
  signed int j; // [esp+4Ch] [ebp-78h]
  signed int v6; // [esp+50h] [ebp-74h]
  char Str1[97]; // [esp+58h] [ebp-6Ch] BYREF
  __int16 v8; // [esp+B9h] [ebp-Bh]
  char v9; // [esp+BBh] [ebp-9h]
  int v10; // [esp+BCh] [ebp-8h]
  int v11; // [esp+C0h] [ebp-4h]

  v11 = 3;
  v10 = 7;
  memset(Str1, 0, sizeof(Str1));
  v8 = 0;
  v9 = 0;
  puts("Please input a string : ");
  scanf("%s", Str1);
  v6 = strlen(Str1);
  for ( i = 0; i < v6; ++i )
  {
    if ( Str1[i] < 48 || Str1[i] > 57 )
    {
      printf("Sorry! Hang on!");
      return -1;
    }
  }
  for ( j = 0; j < v6; ++j )
    Str1[j] = (v10 + v11 * (Str1[j] - 48)) % 10 + 105;
  if ( !strcmp(Str1, Str2) )
    puts("Ok, you know it. Just hang on.");
  else
    puts("Sorry! Hang on!");
  system("pause");
  return 0;
}
```

解决

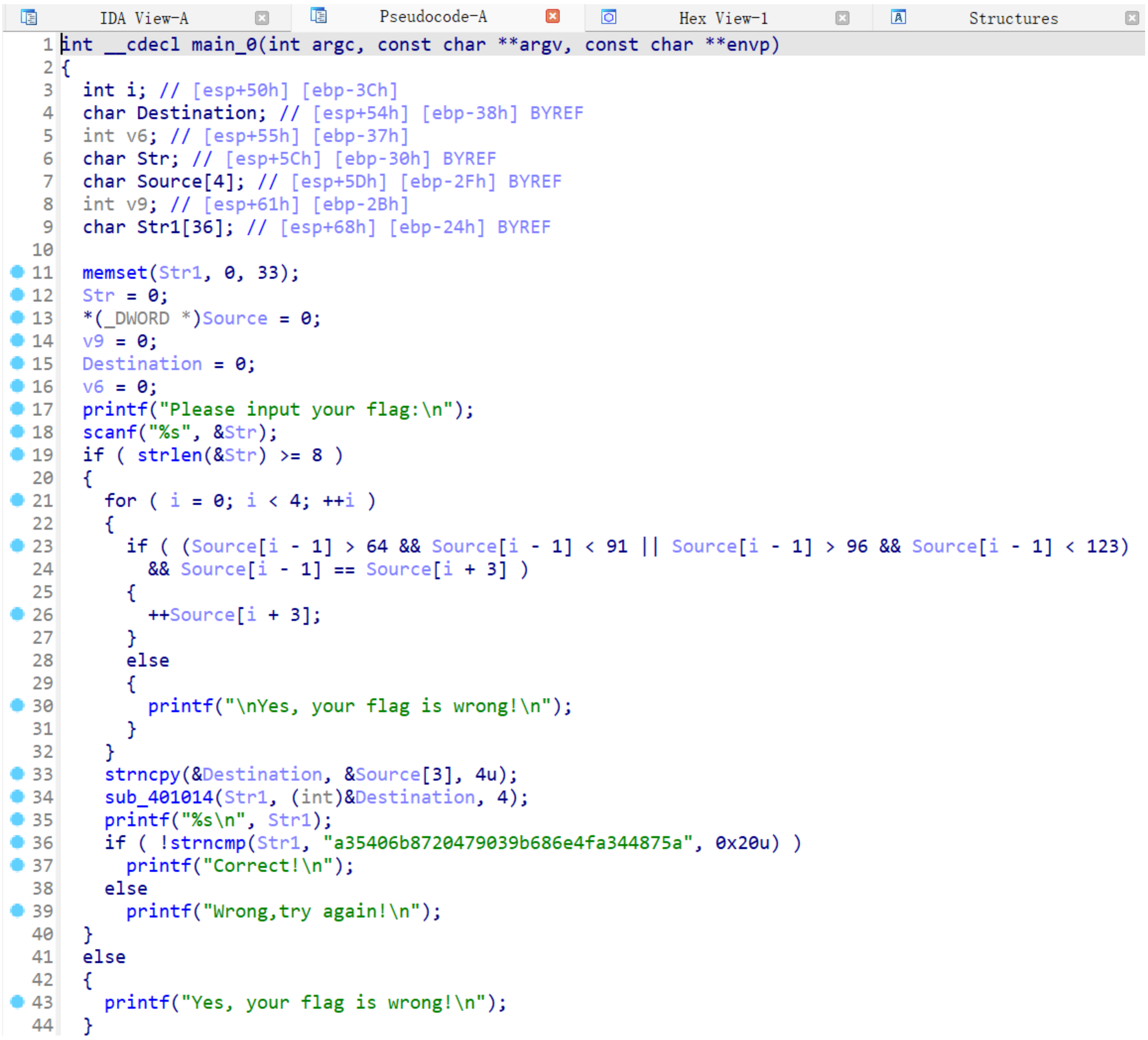
仿射解密脚本。

flag: **10013**

CPP4

关键信息

1. 输入长度为8；
2. 前四个字符和后四个字符相同（`Source - 1` 为输入起始地址）；
3. 后四个字符分别 `+1` 后用 `sub_401014` 加密；
4. 加密结果为 `a35406b8720479039b686e4fa344875a`：猜测MD5，根据 `sub_401014` 中的初始化确定。



```c
int __cdecl main_0(int argc, const char **argv, const char **envp)
{
  int i; // [esp+50h] [ebp-3Ch]
  char Destination; // [esp+54h] [ebp-38h] BYREF
  int v6; // [esp+55h] [ebp-37h]
  char Str; // [esp+5Ch] [ebp-30h] BYREF
  char Source[4]; // [esp+5Dh] [ebp-2Fh] BYREF
  int v9; // [esp+61h] [ebp-2Bh]
  char Str1[36]; // [esp+68h] [ebp-24h] BYREF

  memset(Str1, 0, 33);
  Str = 0;
  *(_DWORD *)Source = 0;
  v9 = 0;
  Destination = 0;
  v6 = 0;
  printf("Please input your flag:\n");
  scanf("%s", &Str);
  if ( strlen(&Str) >= 8 )
  {
    for ( i = 0; i < 4; ++i )
    {
      if ( (Source[i - 1] > 64 && Source[i - 1] < 91 || Source[i - 1] > 96 && Source[i - 1] < 123)
        && Source[i - 1] == Source[i + 3] )
      {
        ++Source[i + 3];
      }
      else
      {
        printf("\nYes, your flag is wrong!\n");
      }
    }
    strncpy(&Destination, &Source[3], 4u);
    sub_401014(Str1, (int)&Destination, 4);
    printf("%s\n", Str1);
    if ( !strncmp(Str1, "a35406b8720479039b686e4fa344875a", 0x20u) )
      printf("Correct!\n");
    else
      printf("Wrong,try again!\n");
  }
  else
  {
    printf("Yes, your flag is wrong!\n");
  }
```

解决

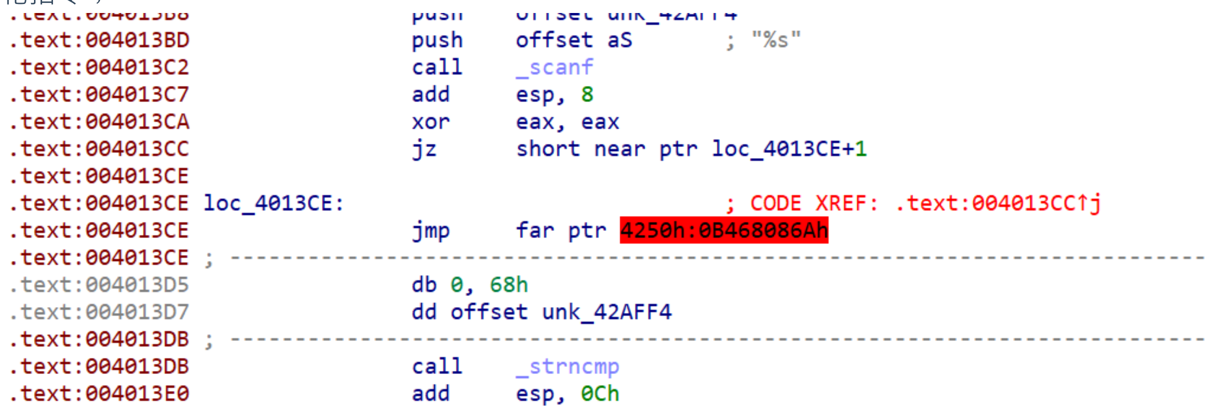
先爆破确定加密函数输入，再分别 `-1` 获得真实输入的一半。

flag: **EasyEasy**

CPP5

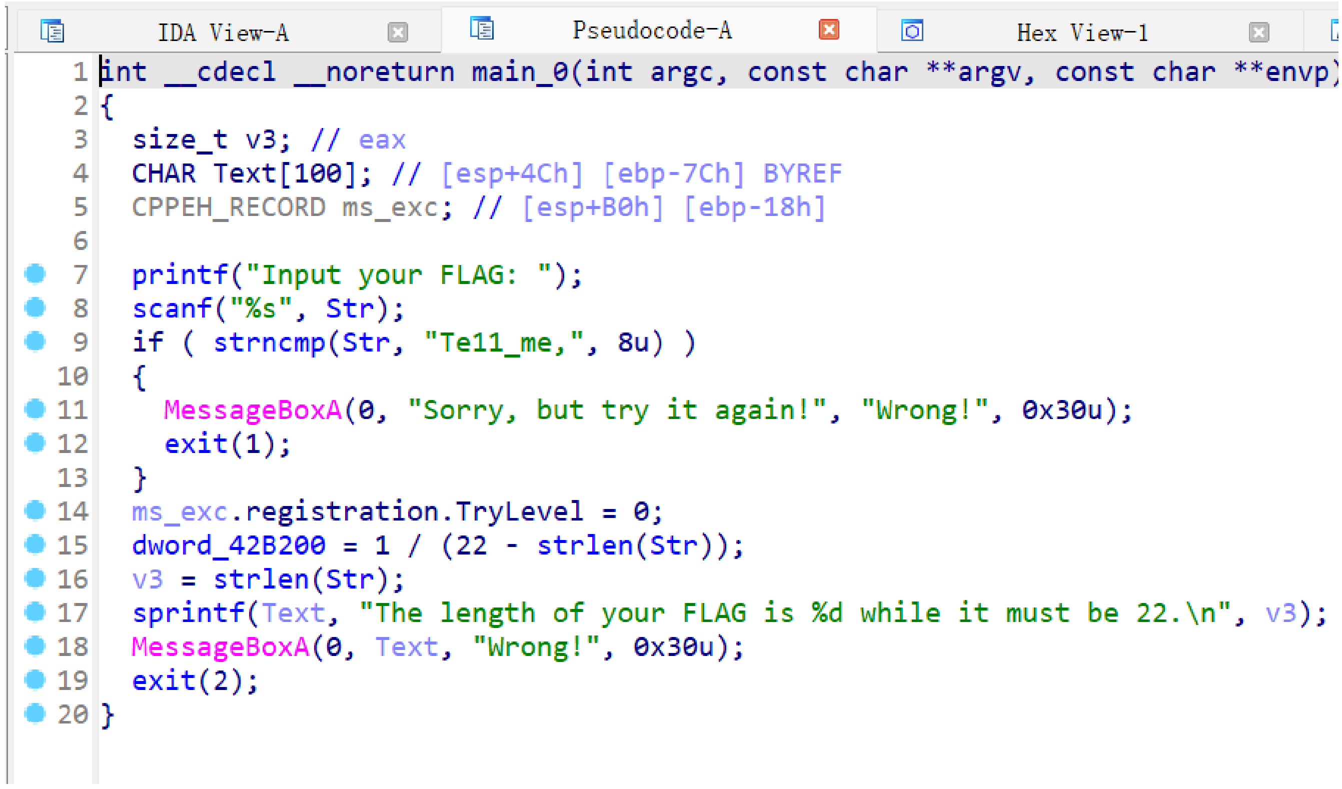
关键信息

1. 花指令；



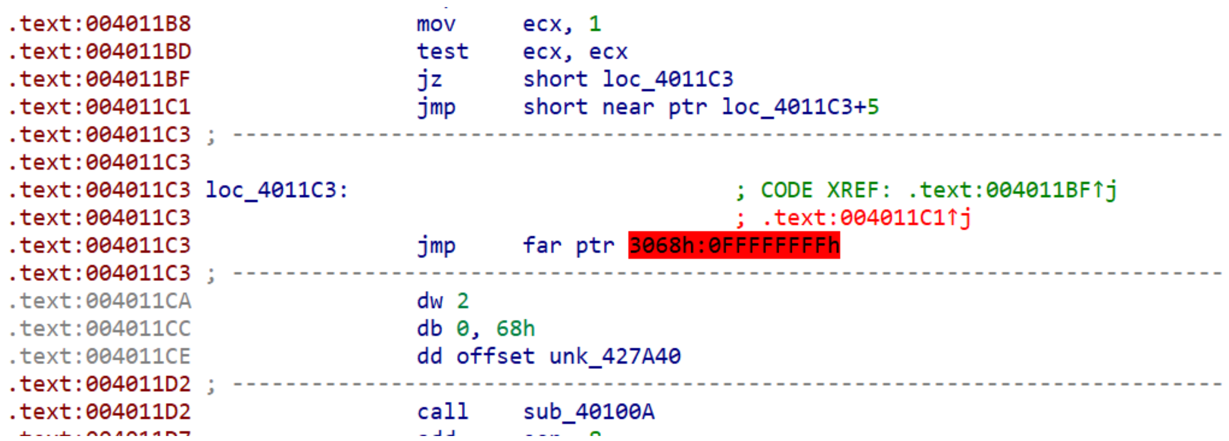
```
.text:004013BD                 push    offset aS       ; "%s"
.text:004013C2                 call    _scanf
.text:004013C7                 add     esp, 8
.text:004013CA                 xor     eax, eax
.text:004013CC                 jz      short near ptr loc_4013CE+1
.text:004013CE
.text:004013CE loc_4013CE:                             ; CODE XREF: .text:004013CC↑j
.text:004013CE                 jmp     far ptr 4250h:0B468086Ah
.text:004013CE ; ---------------------------------------------------------------------------
.text:004013D5                 db 0, 68h
.text:004013D7                 dd offset unk_42AFF4
.text:004013DB ; ---------------------------------------------------------------------------
.text:004013DB                 call    _strncmp
.text:004013E0                 add     esp, 0Ch
```

2. 总长度22，前8个字符为`Te11_me,`，除零异常：看汇编；



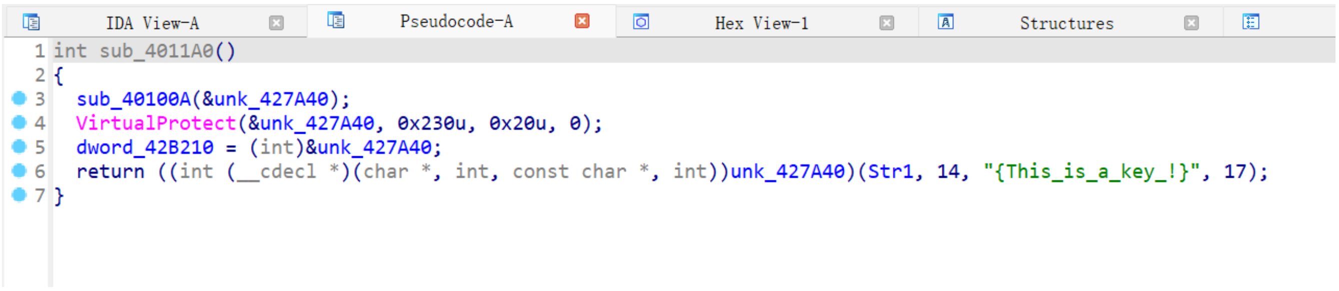
```
int __cdecl __noreturn main_0(int argc, const char **argv, const char **envp)
{
  size_t v3; // eax
  CHAR Text[100]; // [esp+4Ch] [ebp-7Ch] BYREF
  CPPEH_RECORD ms_exc; // [esp+B0h] [ebp-18h]

  printf("Input your FLAG: ");
  scanf("%s", Str);
  if ( strncmp(Str, "Te11_me,", 8u) )
  {
    MessageBoxA(0, "Sorry, but try it again!", "Wrong!", 0x30u);
    exit(1);
  }
  ms_exc.registration.TryLevel = 0;
  dword_42B200 = 1 / (22 - strlen(Str));
  v3 = strlen(Str);
  sprintf(Text, "The length of your FLAG is %d while it must be 22.\n", v3);
  MessageBoxA(0, Text, "Wrong!", 0x30u);
  exit(2);
}
```

3. 异常处理函数中有花指令；



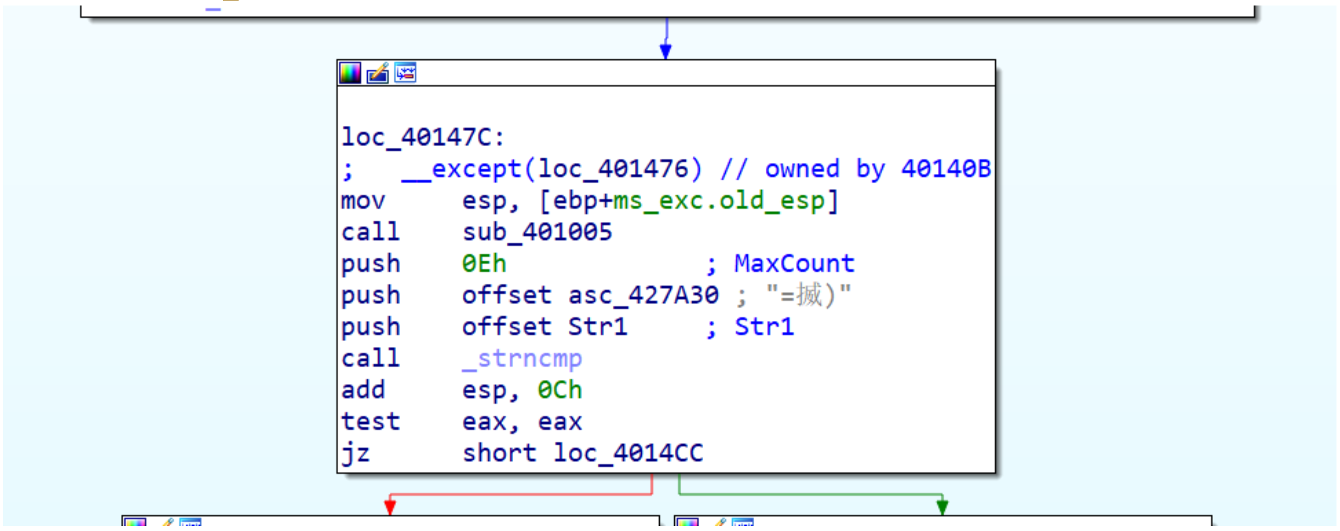
```
.text:004011B8                 mov     ecx, 1
.text:004011BD                 test    ecx, ecx
.text:004011BF                 jz      short loc_4011C3
.text:004011C1                 jmp     short near ptr loc_4011C3+5
.text:004011C3 ; ---------------------------------------------------------------------------
.text:004011C3
.text:004011C3 loc_4011C3:                             ; CODE XREF: .text:004011BF↑j
.text:004011C3                                         ; .text:004011C1↑j
.text:004011C3                 jmp     far ptr 3068h:0FFFFFFFFh
.text:004011C3 ; ---------------------------------------------------------------------------
.text:004011CA                 dw 2
.text:004011CC                 db 0, 68h
.text:004011CE                 dd offset unk_427A40
.text:004011D2 ; ---------------------------------------------------------------------------
.text:004011D2                 call    sub_40100A
```

4. SMC + 加密：`{This_is_a_key_!}` 大概率为密钥；



```
int sub_4011A0()
{
  sub_40100A(&unk_427A40);
  VirtualProtect(&unk_427A40, 0x230u, 0x20u, 0);
  dword_42B210 = (int)&unk_427A40;
  return ((int (__cdecl *)(char *, int, const char *, int))unk_427A40)(Str1, 14, "{This_is_a_key_!}", 17);
}
```

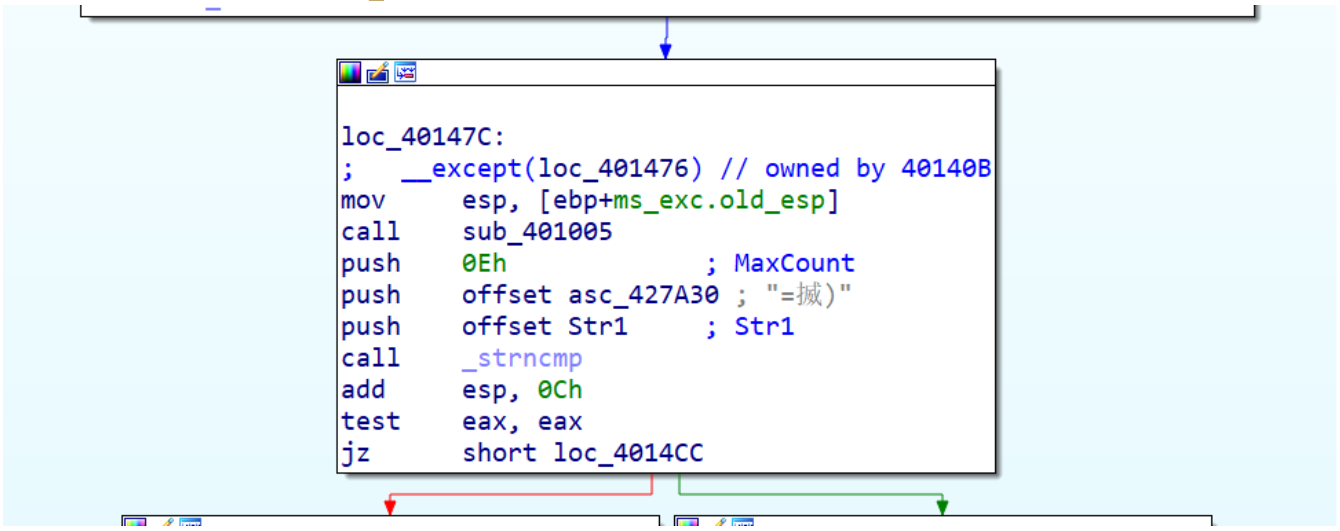
5. 解密结果存在 `asc_427A30` 中。



```
loc_40147C:
;   __except(loc_401476) // owned by 40140B
mov     esp, [ebp+ms_exc.old_esp]
call    sub_401005
push    0Eh             ; MaxCount
push    offset asc_427A30 ; "=摵)"
push    offset Str1     ; Str1
call    _strncmp
add     esp, 0Ch
test    eax, eax
jz      short loc_4014CC
```

解决

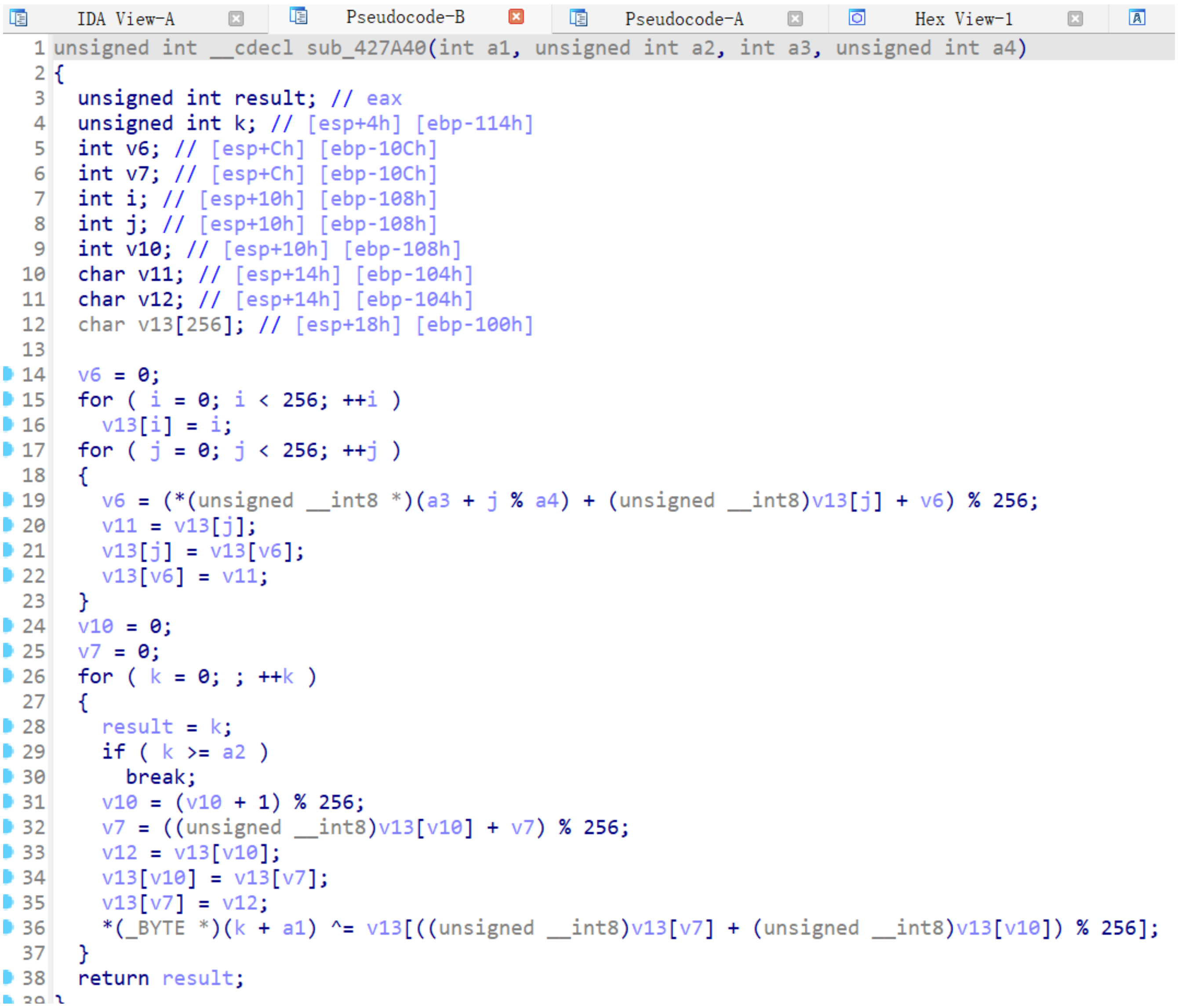
1. 去掉花指令 => NOP；
2. 看汇编找异常处理函数 => `sub_401005`；



```
loc_40147C:
;   __except(loc_401476) // owned by 40140B
mov     esp, [ebp+ms_exc.old_esp]
call    sub_401005
push    0Eh             ; MaxCount
push    offset asc_427A30 ; "=摵)"
push    offset Str1     ; Str1
call    _strncmp
add     esp, 0Ch
test    eax, eax
jz      short loc_4014CC
```

3. 去掉花指令 => NOP；

4. 使用脚本修改指定地址内容，创建函数，确认RC4加密且已知密钥；



IDA View-A | Pseudocode-B | Pseudocode-A | Hex View-1

```
unsigned int __cdecl sub_427A40(int a1, unsigned int a2, int a3, unsigned int a4)
{
  unsigned int result; // eax
  unsigned int k; // [esp+4h] [ebp-114h]
  int v6; // [esp+Ch] [ebp-10Ch]
  int v7; // [esp+Ch] [ebp-10Ch]
  int i; // [esp+10h] [ebp-108h]
  int j; // [esp+10h] [ebp-108h]
  int v10; // [esp+10h] [ebp-108h]
  char v11; // [esp+14h] [ebp-104h]
  char v12; // [esp+14h] [ebp-104h]
  char v13[256]; // [esp+18h] [ebp-100h]

  v6 = 0;
  for ( i = 0; i < 256; ++i )
    v13[i] = i;
  for ( j = 0; j < 256; ++j )
  {
    v6 = (*(unsigned __int8 *)(a3 + j % a4) + (unsigned __int8)v13[j] + v6) % 256;
    v11 = v13[j];
    v13[j] = v13[v6];
    v13[v6] = v11;
  }
  v10 = 0;
  v7 = 0;
  for ( k = 0; ; ++k )
  {
    result = k;
    if ( k >= a2 )
      break;
    v10 = (v10 + 1) % 256;
    v7 = ((unsigned __int8)v13[v10] + v7) % 256;
    v12 = v13[v10];
    v13[v10] = v13[v7];
    v13[v7] = v12;
    *(_BYTE *)(k + a1) ^= v13[((unsigned __int8)v13[v7] + (unsigned __int8)v13[v10]) % 256];
  }
  return result;
}
```

5. 解密工具解RC4，再拼接前8个字符。

flag: **Te11_me,_wh0_i5_Ne2ha?**