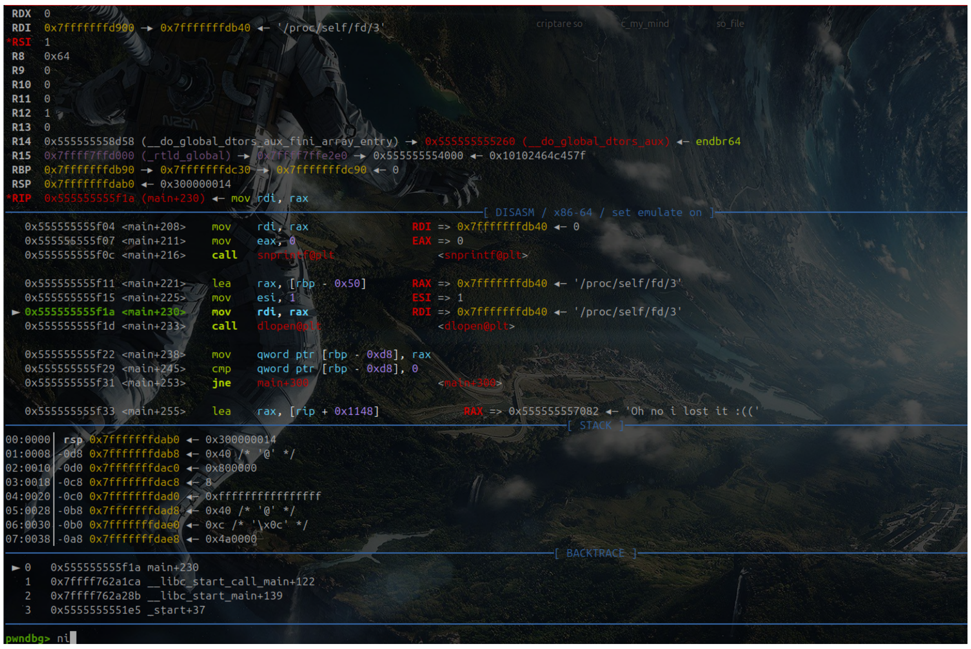
C my mind

Overview

This challenge implies founding the key and iv for the aes cbc encryption used for encrypting the so file that a has the flag in it.

1 Dumping the encrypted so file

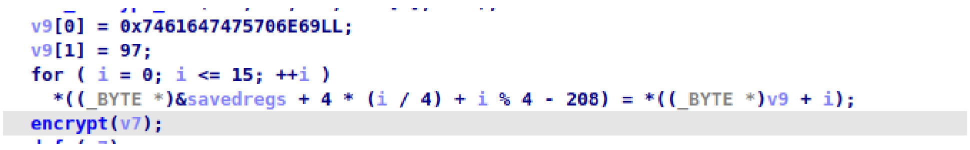
To dump the encrypted so file you need pwndbg to set the breakpoint at main function and from there to run the program untill you see that something is load in memory with dl.Pwndbg will show you the pid and the filedesciptor of the encrypted so file, with which you can dump it with cp /proc/15989/fd/3 /Desktop/cmymind.bin



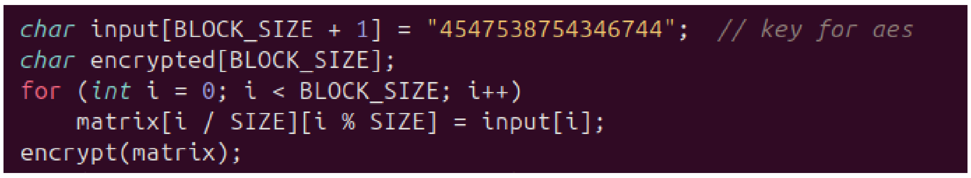
```
RDX  0
RDI  0x7fffffffd900 —▸ 0x7fffffffdb40 ◂— '/proc/self/fd/3'
*RSI  1
R8   0x64
R9   0
R10  0
R11  0
R12  1
R13  0
R14  0x555555558d58 (__do_global_dtors_aux_fini_array_entry) —▸ 0x555555555260 (__do_global_dtors_aux) ◂— endbr64
R15  0x7ffff7ffd000 (_rtld_global) —▸ 0x7ffff7ffe2e0 —▸ 0x555555554000 ◂— 0x10102464c457f
RBP  0x7fffffffdb90 —▸ 0x7fffffffdc30 —▸ 0x7fffffffdc90 ◂— 0
RSP  0x7fffffffdab0 ◂— 0x300000014
*RIP  0x555555555f1a (main+230) ◂— mov rdi, rax
────────────────────────────[ DISASM / x86-64 / set emulate on ]────────────────────────────
   0x555555555f04 <main+208>    mov    rdi, rax                 RDI => 0x7fffffffdb40 ◂— 0
   0x555555555f07 <main+211>    mov    eax, 0                   EAX => 0
   0x555555555f0c <main+216>    call   snprintf@plt                 <snprintf@plt>

   0x555555555f11 <main+221>    lea    rax, [rbp - 0x50]        RAX => 0x7fffffffdb40 ◂— '/proc/self/fd/3'
   0x555555555f15 <main+225>    mov    esi, 1                   ESI => 1
 ► 0x555555555f1a <main+230>    mov    rdi, rax                 RDI => 0x7fffffffdb40 ◂— '/proc/self/fd/3'
   0x555555555f1d <main+233>    call   dlopen@plt                   <dlopen@plt>

   0x555555555f22 <main+238>    mov    qword ptr [rbp - 0xd8], rax
   0x555555555f29 <main+245>    cmp    qword ptr [rbp - 0xd8], 0
   0x555555555f31 <main+253>    jne    main+300                     <main+300>

   0x555555555f33 <main+255>    lea    rax, [rip + 0x1148]         RAX => 0x555555557082 ◂— 'Oh no i lost it :(('
──────────────────────────────────────────────[ STACK ]──────────────────────────────────────────────
00:0000│ rsp 0x7fffffffdab0 ◂— 0x300000014
01:0008│-0d8 0x7fffffffdab8 ◂— 0x40 /* '@' */
02:0010│-0d0 0x7fffffffdac0 ◂— 0x800000
03:0018│-0c8 0x7fffffffdac8 ◂— 8
04:0020│-0c0 0x7fffffffdad0 ◂— 0xffffffffffffffff
05:0028│-0b8 0x7fffffffdad8 ◂— 0x40 /* '@' */
06:0030│-0b0 0x7fffffffdae0 ◂— 0xc /* '\x0c' */
07:0038│-0a8 0x7fffffffdae8 ◂— 0x4a0000
────────────────────────────────────────────[ BACKTRACE ]────────────────────────────────────────────
 ► 0   0x555555555f1a main+230
   1   0x7ffff762a1ca __libc_start_call_main+122
   2   0x7ffff762a28b __libc_start_main+139
   3   0x5555555551e5 _start+37
─────────────────────────────────────────────────────────────────────────────────────────────────────
pwndbg> ni
```

2 Finding and decrypting the the iv and key for the aes

Looking trough the main function you will find a for loop that add values in a matrix from the input data the key.In the binary it was a fake key the code was put there to see what was the process of encryption of the key and iv.

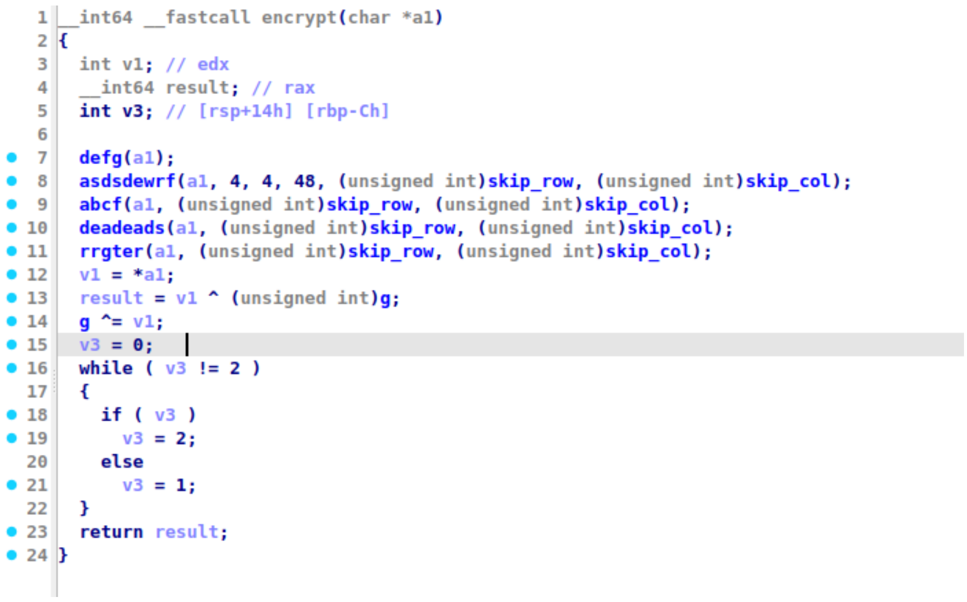


```
v9[0] = 0x7461647475706E69LL;
v9[1] = 97;
for ( i = 0; i <= 15; ++i )
  *((_BYTE *)&savedregs + 4 * (i / 4) + i % 4 - 208) = *((_BYTE *)v9 + i);
encrypt(v7);
```



```c
char input[BLOCK_SIZE + 1] = "4547538754346744";  // key for aes
char encrypted[BLOCK_SIZE];
for (int i = 0; i < BLOCK_SIZE; i++)
    matrix[i / SIZE][i % SIZE] = input[i];
encrypt(matrix);
```

Next looking in the encryption function the important thing from them was the calls for the last 4 functions the rest was junk code to mislead the viewer.

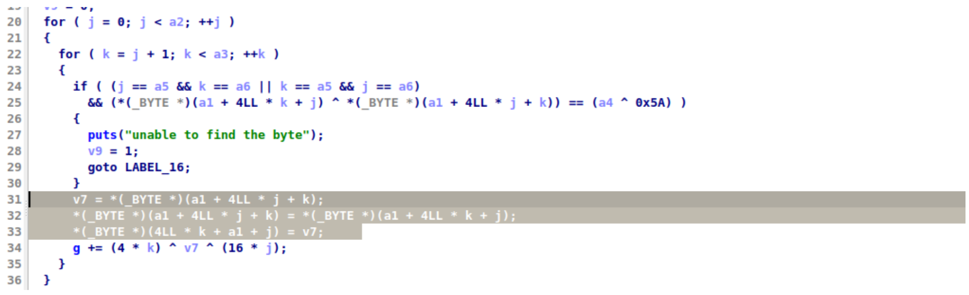


```c
__int64 __fastcall encrypt(char *a1)
{
  int v1; // edx
  __int64 result; // rax
  int v3; // [rsp+14h] [rbp-Ch]

  defg(a1);
  asdsdewrf(a1, 4, 4, 48, (unsigned int)skip_row, (unsigned int)skip_col);
  abcf(a1, (unsigned int)skip_row, (unsigned int)skip_col);
  deadeads(a1, (unsigned int)skip_row, (unsigned int)skip_col);
  rrgter(a1, (unsigned int)skip_row, (unsigned int)skip_col);
  v1 = *a1;
  result = v1 ^ (unsigned int)g;
  g ^= v1;
  v3 = 0;
  while ( v3 != 2 )
  {
    if ( v3 )
      v3 = 2;
    else
      v3 = 1;
  }
  return result;
}
```

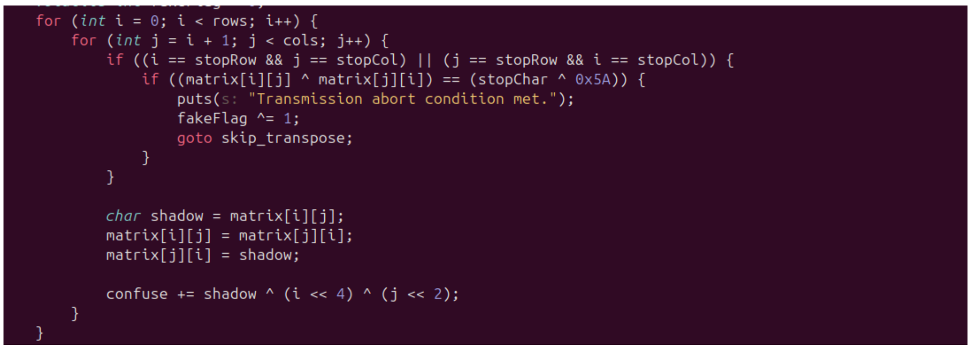
After you find the key and iv you need to make a function to decrypt the prevoius encrypted file that you obtained.To do this i use but you can use any programming language.

That functions represent the order in which the encryption was of the content of matrix with key was done

The first function do the transpose of the matrix the

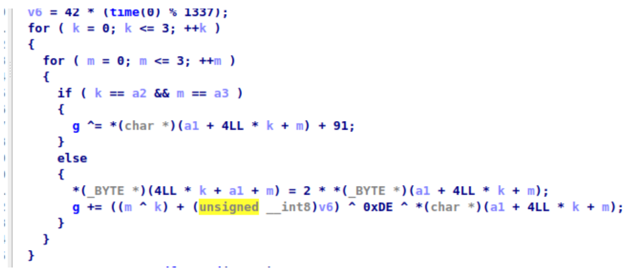


```c
  for ( j = 0; j < a2; ++j )
  {
    for ( k = j + 1; k < a3; ++k )
    {
      if ( (j == a5 && k == a6 || k == a5 && j == a6)
        && (*(_BYTE *)(a1 + 4LL * k + j) ^ *(_BYTE *)(a1 + 4LL * j + k)) == (a4 ^ 0x5A) )
      {
        puts("unable to find the byte");
        v9 = 1;
        goto LABEL_16;
      }
      v7 = *(_BYTE *)(a1 + 4LL * j + k);
      *(_BYTE *)(a1 + 4LL * j + k) = *(_BYTE *)(a1 + 4LL * k + j);
      *(_BYTE *)(4LL * k + a1 + j) = v7;
      g += (4 * k) ^ v7 ^ (16 * j);
    }
  }
```

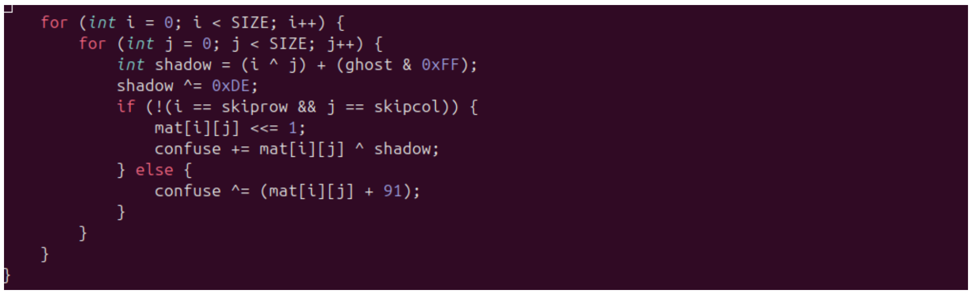


```c
    for (int i = 0; i < rows; i++) {
        for (int j = i + 1; j < cols; j++) {
            if ((i == stopRow && j == stopCol) || (j == stopRow && i == stopCol)) {
                if ((matrix[i][j] ^ matrix[j][i]) == (stopChar ^ 0x5A)) {
                    puts(s: "Transmission abort condition met.");
                    fakeFlag ^= 1;
                    goto skip_transpose;
                }
            }

            char shadow = matrix[i][j];
            matrix[i][j] = matrix[j][i];
            matrix[j][i] = shadow;

            confuse += shadow ^ (i << 4) ^ (j << 2);
        }
    }
```

The second do a multipllication with 2

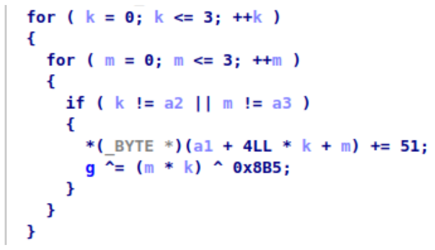


```
v6 = 42 * (time(0) % 1337);
for ( k = 0; k <= 3; ++k )
{
  for ( m = 0; m <= 3; ++m )
  {
    if ( k == a2 && m == a3 )
    {
      g ^= *(char *)(a1 + 4LL * k + m) + 91;
    }
    else
    {
      *(_BYTE *)(4LL * k + a1 + m) = 2 * *(_BYTE *)(a1 + 4LL * k + m);
      g += ((m ^ k) + (unsigned __int8)v6) ^ 0xDE ^ *(char *)(a1 + 4LL * k + m);
    }
  }
}
```

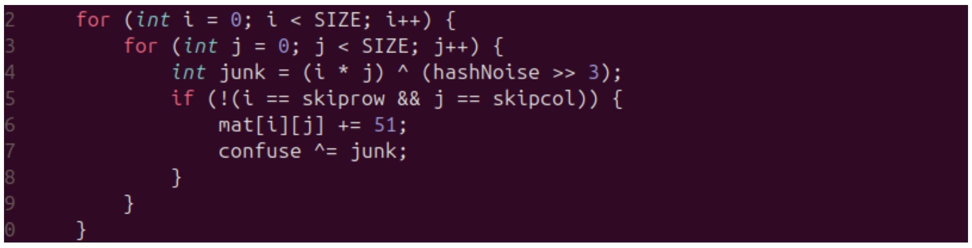


```
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            int shadow = (i ^ j) + (ghost & 0xFF);
            shadow ^= 0xDE;
            if (!(i == skiprow && j == skipcol)) {
                mat[i][j] <<= 1;
                confuse += mat[i][j] ^ shadow;
            } else {
                confuse ^= (mat[i][j] + 91);
            }
        }
    }
}
```

The third do an addition with 51

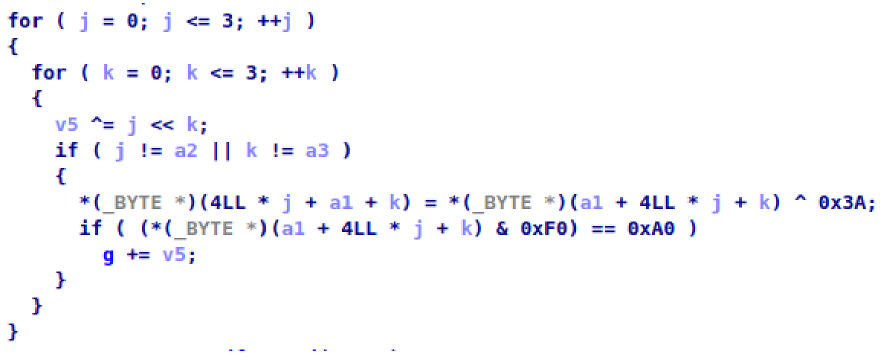


```
for ( k = 0; k <= 3; ++k )
{
  for ( m = 0; m <= 3; ++m )
  {
    if ( k != a2 || m != a3 )
    {
      *(_BYTE *)(a1 + 4LL * k + m) += 51;
      g ^= (m * k) ^ 0x8B5;
    }
  }
}
```

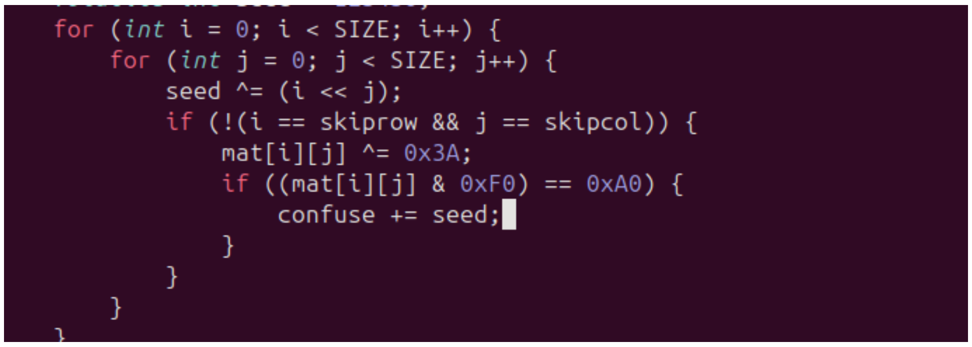


```
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            int junk = (i * j) ^ (hashNoise >> 3);
            if (!(i == skiprow && j == skipcol)) {
                mat[i][j] += 51;
                confuse ^= junk;
            }
        }
    }
```

The last do a xor between all elements of the matrix and 0xa3

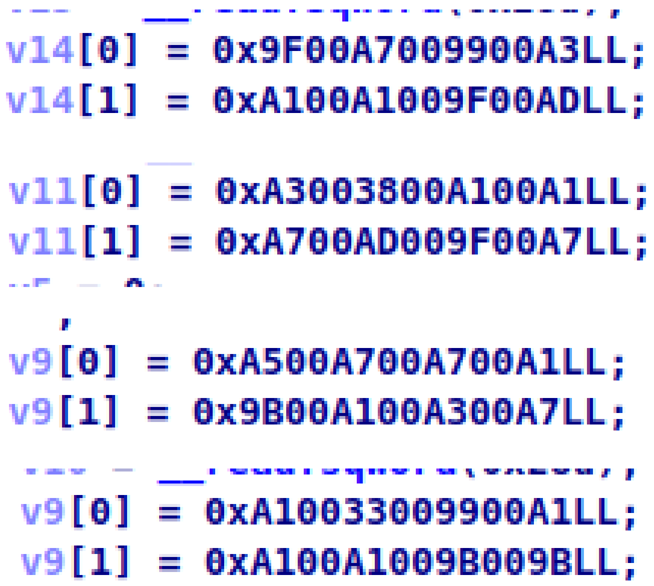


```
for ( j = 0; j <= 3; ++j )
{
  for ( k = 0; k <= 3; ++k )
  {
    v5 ^= j << k;
    if ( j != a2 || k != a3 )
    {
      *(_BYTE *)(4LL * j + a1 + k) = *(_BYTE *)(a1 + 4LL * j + k) ^ 0x3A;
      if ( (*(_BYTE *)(a1 + 4LL * j + k) & 0xF0) == 0xA0 )
        g += v5;
    }
  }
}
```



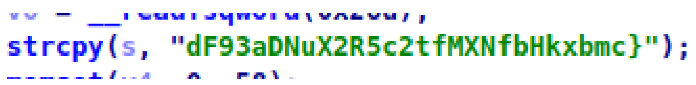
```
for (int i = 0; i < SIZE; i++) {
    for (int j = 0; j < SIZE; j++) {
        seed ^= (i << j);
        if (!(i == skiprow && j == skipcol)) {
            mat[i][j] ^= 0x3A;
            if ((mat[i][j] & 0xF0) == 0xA0) {
                confuse += seed;
            }
        }
    }
}
```

Another important things from the functions were the hex values that you can see at the begining of the functions.In every function of that 4 you will find 8 hex values the decompiler show theme in little endian.The 8 hex values from the first function represent the first 8 bytes of the key the second the last 8 bytes of the key.In the third function you will find the first 8 bytes of the iv and in the last you will find the last 8 bytes from the iv.The bytes for the key and iv that found in that functions we re encrypted with the matrix algorithm that you sawn. So you should reverse the encryption that encrypt function do.

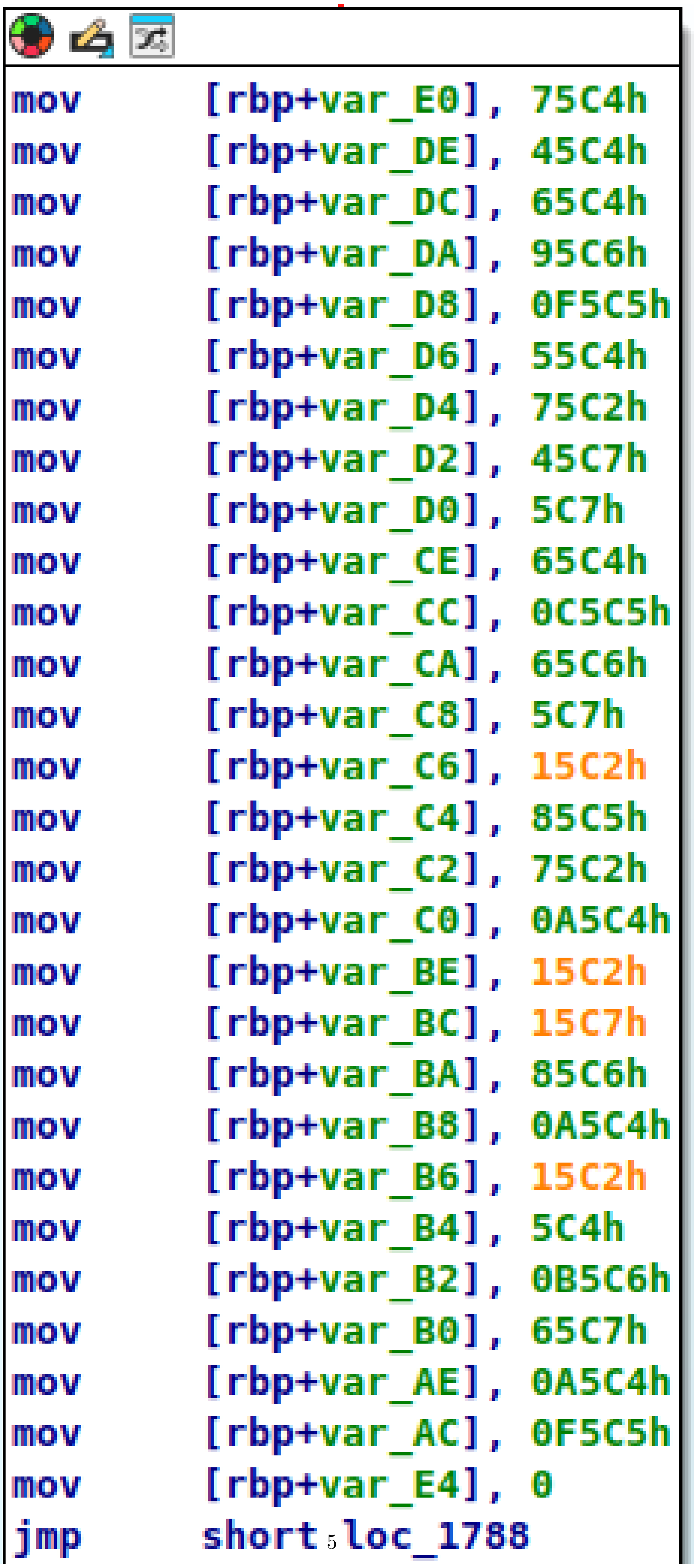


```
v14[0] = 0x9F00A7009900A3LL;
v14[1] = 0xA100A1009F00ADLL;

v11[0] = 0xA3003800A100A1LL;
v11[1] = 0xA700AD009F00A7LL;

v9[0] = 0xA500A700A700A1LL;
v9[1] = 0x9B00A100A300A7LL;

v9[0] = 0xA10033009900A1LL;
v9[1] = 0xA100A1009B009BLL;
```

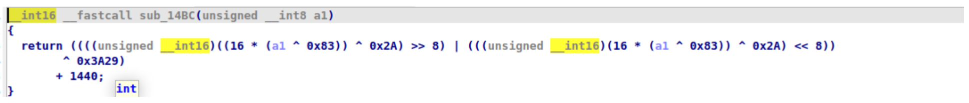
/ After you find the key and iv you decrypt the encrypted blob and get the so file where you will find half in base64 of the flag and an array of hex that represent the second part of the flag in base64.The hex values were encrypted one by one with the algorithm that you can find in the so file.



```
strcpy(s, "dF93aDNuX2R5c2tfMXNfbHkxbmc}");
```



```
mov     [rbp+var_E0], 75C4h
mov     [rbp+var_DE], 45C4h
mov     [rbp+var_DC], 65C4h
mov     [rbp+var_DA], 95C6h
mov     [rbp+var_D8], 0F5C5h
mov     [rbp+var_D6], 55C4h
mov     [rbp+var_D4], 75C2h
mov     [rbp+var_D2], 45C7h
mov     [rbp+var_D0], 5C7h
mov     [rbp+var_CE], 65C4h
mov     [rbp+var_CC], 0C5C5h
mov     [rbp+var_CA], 65C6h
mov     [rbp+var_C8], 5C7h
mov     [rbp+var_C6], 15C2h
mov     [rbp+var_C4], 85C5h
mov     [rbp+var_C2], 75C2h
mov     [rbp+var_C0], 0A5C4h
mov     [rbp+var_BE], 15C2h
mov     [rbp+var_BC], 15C7h
mov     [rbp+var_BA], 85C6h
mov     [rbp+var_B8], 0A5C4h
mov     [rbp+var_B6], 15C2h
mov     [rbp+var_B4], 5C4h
mov     [rbp+var_B2], 0B5C6h
mov     [rbp+var_B0], 65C7h
mov     [rbp+var_AE], 0A5C4h
mov     [rbp+var_AC], 0F5C5h
mov     [rbp+var_E4], 0
jmp     short loc_1788
```



```
__int16 __fastcall sub_14BC(unsigned __int8 a1)
{
  return ((((unsigned __int16)((16 * (a1 ^ 0x83)) ^ 0x2A) >> 8) | (((unsigned __int16)(16 * (a1 ^ 0x83)) ^ 0x2A) << 8))
        ^ 0x3A29)
       + 1440;
}
```

3 UVT{1n_m3mory_w3_trust_wh3n_dysk_1s_ly1ng}

4 Thanks for reading this write-up!Hopefully,the explanantion helped clear things up.