[2018看雪] - 第七题 - 有限域数学变换(2)

原创

Flying_Fatty 于 2018-07-28 23:38:44 发布 216 收藏
分类专栏： reverse

本文链接：https://blog.csdn.net/kevin66654/article/details/81266336
版权
reverse 专栏收录该内容
24 篇文章 0 订阅
订阅专栏

既然是个逆向题，就一定有逆向的做法

学习链接：https://bbs.pediy.com/thread-229327.htm

把题目中的运算理解成代数运算中的双射：

正向时：已知（a，b，c）可求d，那么逆向是已知d的，a，b，c三个数可以“知二求一”

也即，a、b、c、d四个数知三求一

于是，构造CM的逆运算成为了可能，倒过来算20步可以把Input运算出来（再正向检验）

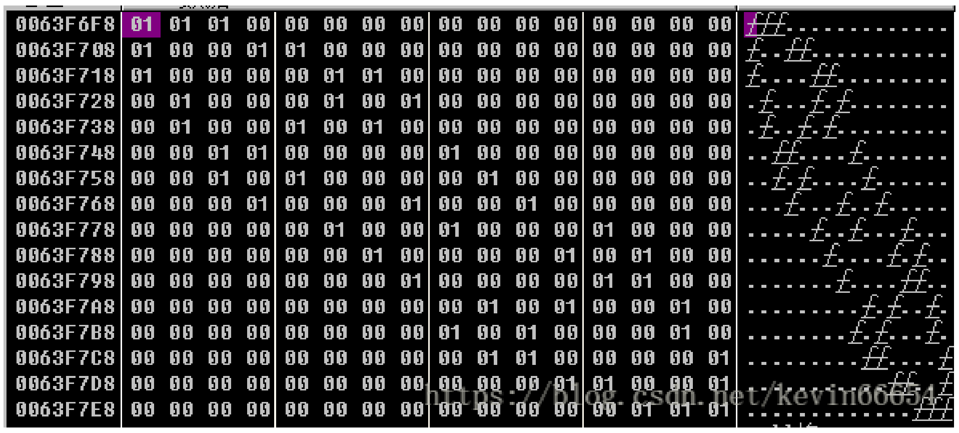
逆向思路：

我们要从数列的第Xn项，逆向算到数列的第1项，本质上是一个重复的不断爆破密钥a，b，c的过程

由于a，b，c满足题意中的某种数学关系，导致暴力枚举成为了可能（64 * 64 *64，且满足出现频率的关系）

深度固定的while循环爆破

所以，如果全部爆破，每个数有64种选择，总共要爆破出来16个数，达到了64 ^ 16的计算量，还得计算20次，这样做是不现实的，于是，可以提前把表格打好进行预先处理



```
0063F6F8  01 01 01 00  00 00 00 00  00 00 00 00  00 00 00 00  £££.............
0063F708  01 00 00 01  01 00 00 00  00 00 00 00  00 00 00 00  £..££...........
0063F718  01 00 00 00  00 01 01 00  00 00 00 00  00 00 00 00  £....££.........
0063F728  00 01 00 00  00 01 00 01  00 00 00 00  00 00 00 00  .£...£.£........
0063F738  00 01 00 00  01 00 01 00  00 00 00 00  00 00 00 00  .£..£.£.........
0063F748  00 00 01 01  00 00 00 00  01 00 00 00  00 00 00 00  ..££....£.......
0063F758  00 00 01 00  01 00 00 00  00 01 00 00  00 00 00 00  ..£.£....£......
0063F768  00 00 00 01  00 00 00 01  00 00 01 00  00 00 00 00  ...£...£..£.....
0063F778  00 00 00 00  00 01 00 00  01 00 00 00  01 00 00 00  .....£..£...£...
0063F788  00 00 00 00  00 00 01 00  00 00 00 01  00 01 00 00  ......£....£.£..
0063F798  00 00 00 00  00 00 00 01  00 00 00 00  01 01 00 00  .......£....££..
0063F7A8  00 00 00 00  00 00 00 00  00 01 00 01  00 00 01 00  .........£.£..£.
0063F7B8  00 00 00 00  00 00 00 00  01 00 01 00  00 00 01 00  ........£.£...£.
0063F7C8  00 00 00 00  00 00 00 00  00 01 01 00  00 00 00 01  .........££....£
0063F7D8  00 00 00 00  00 00 00 00  00 00 00 01  01 00 00 01  ...........££..£
0063F7E8  00 00 00 00  00 00 00 00  00 00 00 00  00 01 01 01  .............£££
```

```
result[0]: 0 1 2
result[1]: 3 4 0
result[2]: 5 6 0
result[3]: 5 7 1
result[4]: 4 6 1
result[5]: 8 2 3
result[6]: 9 2 4
result[7]: 7 10 3
result[8]: 8 12 5
result[9]: 11 13 6
result[10]: 12 13 7
result[11]: 11 14 9
result[12]: 14 8 10
result[13]: 15 9 10
result[14]: 15 11 12
result[15]: 15 13 14
```

假设，我们已知的值是result[16]这个数组，要求的是前一轮的result，不妨把这16个值设为x0，x1，……，x15

我们对x0，x1进行枚举之后，可以发现：x2不需要枚举，可以通过x0、x1、result[0]求出来，继续这么来的话：

枚举x3之后，x4可以通过x0，x3，result[1]求出

枚举x5之后，x6可以通过x0，x5，result[2]求出（同时需要检验：x1、x4、x6与result[4]的关系）

x7可以通过x1，x5，result[3]求出

x8可以通过x2，x3，result[5]求出

得到如下的爆破代码：

```
#include <iostream>
#include <stdio.h>
using namespace std;

typedef unsigned char u8;
u8 cube[64][64][64];
u8 GetNumber1[64][64][64];
u8 GetNumber2[64][64][64];
u8 GetNumber3[64][64][64];
u8 result[16] = {0x14,0x22,0x1E,0x10,0x38,0x30,0x18,0x10,4,0x1A,0x24,8,2,0x26,0x38,0x2A};

void Print(int x){
 printf("<%d>: ",x);
 for(int i = 0; i < 16; i++)
  printf("%02X ",result[i]);
 char *sz = "abcdefghijklmnopqrstuvwxyz+-ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789";
    for(int i = 0; i < 16; i++)
        printf("%c",sz[result[i]]);
    puts("");
}

void test(){
    u8 x0,x1,x2,x3,x4,x5,x6,x7,x8,x9,x10,x11,x12,x13,x14,x15;
    for(x0 = 0; x0 < 64; x0++)
        for(x1 = 0; x1 < 64; x1++){
            x2 = GetNumber3[x0][x1][result[0]];
            for(x3 = 0; x3 < 64; x3++){
                x4 = GetNumber2[x3][result[1]][x0];
                for(x5 = 0; x5 < 64; x5++){
                    x6 = GetNumber2[x5][result[2]][x0];
                    if (x6 == GetNumber2[x4][result[4]][x1]){
                        x7 = GetNumber2[x5][result[3]][x1];
                        x8 = GetNumber1[result[5]][x2][x3];
                        x9 = GetNumber1[result[6]][x2][x4];
                        x10 = GetNumber2[x7][result[7]][x3];
                        x12 = GetNumber2[x8][result[8]][x5];
                        x15 = GetNumber1[result[13]][x9][x10];
                        x11 = GetNumber2[x15][result[14]][x12];
                        x14 = GetNumber2[x11][result[11]][x9];
                        x13 = GetNumber2[x11][result[9]][x6];
                        if (GetNumber1[result[10]][x13][x7] == x12 &&
                            GetNumber1[result[12]][x8][x10] == x14 &&
                            GetNumber1[result[15]][x13][x14] == x15){
                                result[0] = x0;
                                result[1] = x1;
                                result[2] = x2;
                                result[3] = x3;
                                result[4] = x4;
                                result[5] = x5;
                                result[6] = x6;
                                result[7] = x7;
                                result[8] = x8;
                                result[9] = x9;
                                result[10] = x10;
                                result[11] = x11;
                                result[12] = x12;
                                result[13] = x13;
                                result[14] = x14;
                                result[15] = x15;
                                return;
                            }
                    }
                }
            }
        }
}

int main(){
    FILE *f = fopen("Escape.exe","rb");
    fseek(f,0xe4f0,0);
    fread(cube,64*64*64,1,f);
    fclose(f);
    for(int i = 0; i < 64; i++)
     for(int j = 0; j < 64; j++)
      for(int k = 0; k < 64; k++){
       GetNumber1[cube[i][j][k]][j][k] = i;
       GetNumber2[i][cube[i][j][k]][k] = j;
       GetNumber3[i][j][cube[i][j][k]] = k;
      }
    Print(0);
    for(int i = 1; i <= 0x500; i++){
        test();
        Print(i);
    }
    return 0;
}
```