writeup

通过字符串定位到关键函数，sub_4024A9。

```
.rdata:000000000040C006                 db    8
.rdata:000000000040C007                 db    0
.rdata:000000000040C008 aInputKey       db 'input key: ',0      ; DATA XREF: sub_4022C9+12↑o
.rdata:000000000040C014 aCorrectFlagS   db 'correct!, flag{%s}',0
.rdata:000000000040C014                                         ; DATA XREF: sub_4024A9+E8↑o
.rdata:000000000040C027                 align 10h
.rdata:000000000040C030 off_40C030      dq offset qword_120F660 ; DATA XREF: sub_402AF0+B7↑o
.rdata:000000000040C038                 dq offset unk_120F180
```

函数sub_4022C9获取32位的输入。

```
1  __int64 __fastcall sub_4022C9(__int64 a1, __int64 a2, __int64 a3, __int64 a4)
2  {
3    __int64 v4; // rax
4    __int64 v6; // [rsp+40h] [rbp-40h]
5
6    v6 = a4;
7    v4 = ZStlsISt11char_traitsIcEERSt13basic_ostreamIcT_ES5_PKc(a1, a2, "input key: ", &ZSt4cout);
8    ZNSolsEPFRSoS_E(a1, a2, &ZSt4endlIcSt11char_traitsIcEERSt13basic_ostreamIT_T0_ES6_, v4);
9    sub_409B80(a1);
10   ZStrsIcSt11char_traitsIcESaIcEERSt13basic_istreamIT_T0_ES7_RNSt7__cxx1112basic_stringIS4_S5_T1_EE(
11     a1,
12     a2,
13     v6,
14     &ZSt3cin);
15   if ( sub_408FA0(a1) != 32 )
16     exit(a1);
17   return v6;
18 }
```

函数sub_402357接受一个字符串，按照二进制将它转换成vector。

```
13
14  v14 = a4;
15  ZNSaIcEC1Ev(a1, a2, a3, (char *)&v8 + 63);
16  v4 = sub_408FA0(a1);
17  sub_409900(a1, a2, 8 * v4 + 1);
18  ZNSaIcED1Ev(a1, a2, v5, (__int64)&v10);
19  for ( i = 0; ; ++i )
20  {
21    v6 = i;
22    if ( v6 >= sub_408FA0(a1) )
23      break;
24    v12 = *(char *)sub_409C10(a1, a2);
25    v11 = 0;
26    while ( v12 > 0 )
27    {
28      if ( v12 & 1 )
29        *(_BYTE *)sub_4099C0(a1, a2, 8 * i + v11, &v9) = 1;
30      ++v11;
31      v12 >>= 1;
32    }
33  }
34  sub_409780(a1, a2);
35  sub_401C7D(a1, a2);
36  sub_409970(a1);
37  sub_409970(a1);
38  return v14;
39}
```

函数sub_401F6B为多项式乘法、函数sub_401CE9为多项式加法。

接下来将结果与byte_40B020数据进行比较。

将上述计算过程求逆即可，先进行减法，再进行除法：

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>
using namespace std;

const int MOD = 2;
const int FFTN = 1<<19;
#define poly vector<char>
typedef unsigned long long int ull;

int ksm(int a, int b) {
    int ret = 1;
    while(b) {
        if(b&1) ret = 1ll * ret * a % MOD;
        b >>= 1;
```

```
        a = 1ll * a * a % MOD;
    }
    return ret;
}

namespace FFT {
    int w[FFTN+5],W[FFTN+5],R[FFTN+5];
    void FFTinit() {
        W[0]=1;
        W[1]=ksm(3,(MOD-1)/FFTN);
        for(int i = 2; i <= FFTN; i++) W[i]=1ll*W[i-1]*W[1]%MOD;
    }
    int FFTinit(int n) {
        int L=1;
        for (; L<=n; L<<=1);
        for(int i = 0; i <= L - 1; i++) R[i]=(R[i>>1]>>1)|((i&1)?
(L>>1):0);
        return L;
    }
    int A[FFTN+5],B[FFTN+5];
    ull p[FFTN+5];
    void DFT(int *a,int n) {
        for(int i = 0; i < n; i++) p[R[i]]=a[i];
        for(int d = 1; d < n; d <<= 1) {
            int len=FFTN/(d<<1);
            for(int i = 0, j = 0; i < d; i++, j += len) w[i]=W[j];
            for(int i = 0; i < n; i += (d<<1))
                for (int j = 0; j < d; j++) {
                    int y=p[i+j+d]*w[j]%MOD;
                    p[i+j+d]=p[i+j]+MOD-y;
                    p[i+j]+=y;
                }
            if (d==1<<15)
                for(int i = 0; i < n; i++) p[i]%=MOD;
        }
        for(int i = 0; i < n; i++) a[i]=p[i]%MOD;
    }
    void IDFT(int *a,int n) {
        for(int i = 0; i < n; i++) p[R[i]]=a[i];
        for (int d=1; d<n; d<<=1) {
            int len=FFTN/(d<<1);
            for (int i=0,j=FFTN; i<d; i++,j-=len) w[i]=W[j];
            for (int i=0; i<n; i+=(d<<1))
                for (int j=0; j<d; j++) {
                    int y=p[i+j+d]*w[j]%MOD;
                    p[i+j+d]=p[i+j]+MOD-y;
                    p[i+j]+=y;
                }
            if (d==1<<15)
                for(int i = 0; i < n; i++) p[i]%=MOD;
        }
        int val=ksm(n,MOD-2);
        for(int i = 0; i < n; i++) a[i]=p[i]*val%MOD;
    }
```

```
/*
 * 去掉多项式前面多余的0
 */
poly Norm(poly a) {
    for (; a.size()&&!a.back(); a.pop_back());
    return a;
}

/*
 * 多项式减法、低位在数组低位置
 */
poly Minus(const poly &a,const poly &b) {
    int sza=a.size()-1, szb=b.size()-1;
    poly ans(max(sza,szb)+1);
    for(int i = 0; i <= sza; i++) ans[i]=a[i];
    for(int i = 0; i <= szb; i++) ans[i]=ans[i]^b[i];
    return ans;
}

poly Mul(const poly &a, const poly &b) {
    int sza=a.size()-1,szb=b.size()-1;
    poly ans(sza+szb+1);
    if (sza<=30||szb<=30) {
        for(int i = 0; i <= sza; i++) for(int j = 0; j <= szb; j++)
                ans[i+j]=(ans[i+j]+1ll*a[i]*b[j])%MOD;
        return ans;
    }
    int L=FFTinit(sza+szb);
    for(int i = 0; i < L; i++) A[i]=(i<=sza?a[i]:0);
    for(int i = 0; i < L; i++) B[i]=(i<=szb?b[i]:0);
    DFT(A,L);
    DFT(B,L);
    for(int i = 0; i < L; i++) A[i]=1ll*A[i]*B[i]%MOD;
    IDFT(A,L);
    for(int i = 0; i <= sza + szb; i++) ans[i]=A[i];
    return ans;
}

/*
 * 得到当前多项式的度
 */
int det(poly p){
    for(int i=p.size()-1;i>=0;i--){
        if(p[i])    return i;
    }
}

poly Div(poly a, poly b){
    int sza=a.size(), szb=b.size();
    if (sza<szb) return poly(0);
    poly q(sza), r=a;
    for(int i=sza-1;i>=0;i--){
        int tt = det(r)-det(b);
```

```
            poly pt(tt+1);
            pt[tt] = 1;
            r = Minus(r, Mul(b, pt));
            q[tt] = 1;
            if(det(r)<det(b)) break;
        }
        return Norm(q);
    }

}

using FFT::FFTinit;
using FFT::Div;
using FFT::Minus;
using FFT::Norm;

poly fx = {1 , 0 , 1 , 1 , 1 , 0 , 1 , 0 , 0 , 1 , 0 , 1 , 1 , 0 , 0 , 0 ,
0 , 1 , 0 , 1 , 0 , 1 , 1 , 0 , 0 , 1 , 1 , 1 , 0 , 0 , 1 ,};
poly gx = {1 , 1 , 0 , 0 , 0 , 1 , 0 , 1 , 1 , 0 , 0 , 1 , 0 , 1 , 1 , 1 ,
0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 1 , 0 , 1 , 1 , 1 , 0 , 0 , 1 , 1 , 1 , 0
, 0 , 1 , 0 , 1 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 1 , 1 , 1 , 1 , 0 , 1 ,
1 , 0 , 0 , 0 , 0 , 0 , 1 , 1 , 0 , 0 , 0 , 0 , 0 , 0 , 1 , 0 , 1 , 0 , 0
, 0 , 1 , 1 , 0 , 0 , 1 , 0 , 1 , 0 , 1 , 1 , 0 , 1 , 0 , 1 , 1 , 1 , 1 ,
0 , 1 , 0 , 0 , 1 , 0 , 1 , 1 , 1 , 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 0
, 1 , 0 , 0 , 1 , 1 , 0 , 0 , 0 , 0 , 1 , 1 , 1 , 1 , 0 , 0 , 1 , 0 , 0 ,
1 , 1 , 1 , 0 , 0 , 0 , 0 , 1 , 0 , 1 , 1 , 1 , 0 , 0 , 0 , 0 , 1 , 0 , 1
, 1 , 1 , 1 , 1 , 0 , 0 , 1 , 1 , 1 , 1 , 0 , 0 , 1 , 1 , 1 , 0 , 1 , 0 ,
1 , 1 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 0 , 0 , 1 , 0 , 1 , 0 , 0 , 1 , 1 , 0
, 1 , 1 , 1 , 1 , 0 , 1 , 1 , 1 , 0 , 1 , 1 , 1 , 1 , 1 , 1 , 1 , 0 , 1 ,
1 , 0 , 1 , 0 , 1 , 0 , 0 , 1 , 1 , 0 , 1 , 0 , 1 , 0 , 1 , 1 , 1 , 1 , 1
, 0 , 1 , 1 , 0 , 0 , 0 , 0 , 1 , 0 , 0 , 1 , 1 , 1 , 0 , 0 , 1 , 1 , 1 ,
1 , 1 , 1 , 1 , 0 , 0 , 0 , 1 , 0 , 1 , 1 , 0 , 1 , 1 , 1 , 1 , 1 , 1 };

char ans[] = {0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0,
1, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1,
            1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1, 0,
            1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1,
1, 1, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 0, 0,
            1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1,
1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 0, 0, 0,
            0, 0, 1, 0, 0, 1, 1, 0, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 0,
1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 0, 0, 1, 1, 0, 0,
            1, 1, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0,
1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1,
            1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1,
1, 0, 0, 0, 1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1,
            0, 1, 0, 1, 1,};

int main(int argc, char** argv){
    poly dt(sizeof(ans));
    for(int i=0;i<dt.size();i++){
        dt[i] = ans[i];
    }
```

```
    poly res = Div(Minus(dt, gx), fx);
    for(int i=0;i<res.size();i+=8){
        int j=6;
        int cur = res[i+7];
        while(j>=0){
            cur <<= 1; cur |= res[i+j]; j--;
        }
        printf("%c", cur);
    }
    return 0;
    # 7b2777572a713c13336ed7aedc6a67bb
}
```