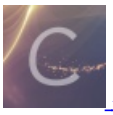
D^3CTF(Crypto-D3bug详解 LFSR题目）

原创

Mango|Feng 于 2022-03-15 12:50:05 发布 367 收藏 1

分类专栏： 密码学 write up python 文章标签： 安全 python 学习

本文链接：https://blog.csdn.net/ZoeMG/article/details/123499857


密码学 同时被 3 个专栏收录

12 篇文章 0 订阅

订阅专栏


write up

5 篇文章 0 订阅

订阅专栏


python

10 篇文章 0 订阅

订阅专栏

D3bug详解(LFSR题目)

Author: MangoFeng

题目

```
from Crypto.Util.number import *
from secret import flag
assert flag.startswith("D3CTF{")
assert flag.endswith("}")
message = bytes_to_long(flag[6:-1])
assert message < 2**64
mask = 0b1010010000001000000010001001010010100100000010000000100010010100

def lfsr_MyCode(R,mask):
    output = (R << 1) & 0xffffffffffffffff
    i = (R ^ mask) & 0xffffffffffffffff
    lastbit = 0
    while i != 0:
        lastbit ^= (i & 1)
        i = i>>1
    output ^= lastbit
    return (output,lastbit)

def lfsr_CopiedfromInternet(R,mask):
    output = (R << 1) & 0xffffffffffffffff
    i = (R & mask) & 0xffffffffffffffff
    lastbit = 0
    while i != 0:
        lastbit ^= (i & 1)
        i = i>>1
    output ^= lastbit
    return (output,lastbit)

f=open("standardResult","w")
R=message
outout=0
for i in range(35):
    (R, out) = lfsr_CopiedfromInternet(R,mask)
    if(i==34):
        outout1=out
    f.write(str(out))
f.close()

f=open("myResult","w")
R=message
for i in range(35):
    (R, out) = lfsr_MyCode(R,mask)
    f.write(str(out))
f.close()

#Why are the results always different?!!
#Can you help me debug my code? QAQ
```

分析一下题目：

题目代码中含有两个函数 `lfsr_MyCode` 和 `lfsr_CopiedfromInternet`

看函数名可以看出，一个是出题人写的LFSR `线性移位反馈寄存器` 一个是 `FromInternet` 。后者可以从网上找到一些类似的例子。

关于 `LFSR` 可以查阅一些相关文章:

例如:

https://ctf-wiki.org/crypto/streamcipher/fsr/lfsr/(线性反馈移位寄存器 - LFSR)

https://www.anquanke.com/post/id/181811(深入分析CTF中的LFSR类题目)

这里我着重描述这道题与其他题不同之处。

在复现的时候，查看许多write up里只有一个关键词: 爆破 (头都看大了好吧，主要还说是一个签到题)

(lfsr_MyCode)step by step：

```
def lfsr_MyCode(R,mask):
    output = (R << 1) & 0xffffffffffffffff
    i = (R ^ mask) & 0xffffffffffffffff
    lastbit = 0
    while i != 0:
        lastbit ^= (i & 1)
        i = i>>1
    output ^= lastbit
    return (output,lastbit)
```

```
def lfsr_MyCode(R,mask):
```

定义了一个函数，名为lsfr_MyCode,其需要两个参数R与mask

```
output = (R<<1) &0xffffffffffffffff
```

先使传入的R向左移一位在&0xffffffffffffffff(即丢弃掉最高位且在低位补零)，将此值赋给output

```
i = (R ^ mask) & 0xffffffffffffffff
```

将传入的R与传入的mask做异或运算之后&0xffffffffffffffff(即R与mask异或取64位),将此值赋给i

```
lastbit = 0
while i != 0:
 lastbit ^= (i & 1)
 i = i>>1
```

(1)初始化一个变量lastbit使其值为0

(2)一个while循环(当i不等于零时循环)

- 让lastbit的值与i&1的值进行异或，再赋给lastbit。(i&1即为取出i的最低位)这句代码再翻译一下即：将i的最低位半加到lastbit中。半加器百度百科(即异或)
- 将i向右移一位(即丢弃掉i的最低位)

整个循环的意思为：将i的每一位的异或值赋给lastbit(lastbit初值为0，不影响异或)
而i的值在上面描述过，是R与mask对位进行异或得到的值。
综合循环解释，最后lastbit的值即为R与mask的每一位进行异或运算的值

```
output ^= lastbit
```

将lastbit的值半加到output上
开始：output为R向左移一位，丢弃掉最高位低位补零
现在：output最低位半加上lastbit

```
return (output,lastbit)
```

输入(R,mask),返回(output,lastbit)

其对应调用的代码为:

```
assert message < 2**64
mask = 0b1010010000001000000010001001010010100100000010000000100010010100

f=open("myResult","w")
R=message
for i in range(35):
    (R, out) = lfsr_MyCode(R,mask)
    f.write(str(out))
f.close()
#myResult:00100110001000110001101010101001001
```

`assert message < 2**64` 保证传给R的message小于64位
`mask已知`
然后循环调用 `lfsr_MyCode` ,初始状态为(R,mask)，之后每一次传参为返回的(output,mask)mask为常数且输出lastbit在myResult文件中:00100110001000110001101010101001001

在上面分析 `lfsr_MyCode` 时提到过，lastbit为传入的R与mask每一位的异或值。

接下来我们分析两轮的lastbit:

(1):

- 传入R，mask，R与mask每一位异或即为lastbit1。此时output为R向左移动一位，丢弃高位，低位加上lastbit1。

(2):

传入output,mask，output与mask每一位异或即为lastbit2。此时用lastbit2去对比lastbit1

lastbit1为R的64位与mask64位异或而得
lastbit2为R的后63位+lastbit1与mask64位异或而得
做一个简单示意图：(A为R高一位，B为剩下63位(简单看成一位），M为mask)
lastbit1=(A^B) ^M
lastbit2=(B^ (A^B) ^M) ^M=A!
后面都可以类推
意思就是，lastbit(n)就为R的第n-1位($n\geq 2$)

> 所以我们可以通过上述分析与35位myResult确定下34位的R高位:
>
> `0100110001000110001101010101001001`

这里确定高位我在做题的时候，没有直接推出上面 `lastbit(n)就为R的第n-1位` 这个结论，我是通过分析lastbit为0和为1时对下一轮操作的影响。

当$lastbit(n)=$ [illegible]

当$lastbit(n)=$ [illegible]

代码实现：

```
lastbit="00100110001000110001101010101001001"
R=''
for i in range(len(lastbit)-1):
    if(lastbit[i]=='1'):
        R+=str(int(lastbit[i])^1^int(lastbit[i+1]))
    else:
        R+=str(int(lastbit[i])^int(lastbit[i+1]))
print(R)
print(type(R))
print(len(R))
#0100110001000110001101010101001001
#<class 'str'>
#34
```

现在我们已经知道了message的高位(34位)，而整个message长64位，即还剩30位低位未求出。

lfsr_CopiedfromInternet()

`lfsr_CopiedfromInternet` 和 `lfsr_MyCode` 的区别在于 `i = (R & mask) & 0xffffffffffffffff`

其实很直接的会想到去 `爆破` 空间为2**30

```
R="0100110001000110001101010101001001"
for i in range(0,2**30):
    RR=(int(R,2)<<30)+i
    beganRR=RR
    s=''
    for j in range(35):
        (RR, out) = lfsr_CopiedfromInternet(RR,mask)
        s+=str(out)
    #print(s)
    if(s=='01111101111010111000010010111001101'):
        print(f"i:{bin(i)[2:]}")
        print(f"R:{RR}")
        print(long_to_bytes(beganRR))
        exit()
    else:
        print(f"not found in {i}")
        continue
```

当遍历到的低位加上我们的高位通过：

```
f=open("standardResult","w")
R=message
outout=0
for i in range(35):
    (R, out) = lfsr_CopiedfromInternet(R,mask)
    if(i==34):
        outout1=out
    f.write(str(out))
f.close()
```

得到的结果与我们已知的时候，即找到了明文。

但他花的时间讲道理，有亿点久。所以下面我们分析一下：

因为mask已知，其以下位值为1

```
mask = 0b10100100000010000000100010010100101001000000100000000100010010100
a="10100100000010000000100010010100101001000000100000000100010010100"
b=list(a)
#b.reverse()
b=''.join(b)
for j in range(35):
    for i in range(len(b)):
        if(b[i]=="1"):
            print(f"a[{i+1+j}]",end="^")
    print('\n')
            #逆3^5^8^12^20^27^30^32^35^37^40^44^52^59^62^64(逆着数)
            如参考安全客文章中就是这样子去数的
            #正0^2^5^12^20^24^27^29^32^34^37^44^52^56^59^61（正着数从0开始）
            #1^3^6^13^21^25^28^30^33^35^38^45^53^57^60^62（正着数从1开始）
```

根据&运算的特性，要参与运算的两个值都为1才能产生1，即对应$mask$中的位值为1的地方，i中才可能出现

当位置相对应的R中有奇数个1的时候，导致i中有奇数个1，可得出lastbit=1

当位置相对应的R中有偶数个1的时候，导致i中有偶数个1，可得出lastbit=0

（这些相对应mask中1的位置的位置我们暂时称为有效位）

且每参与一次运算，有效位在R中都会向右移动一位（R在向左移，有效位在mask中不变，所以对应着R中的有效位在向右移动）

例如第一次调用函数中：

$$Lastbit1= 1⊕ 3⊕ 6⊕$$

第二次调用中：

$$Lastbit2= 2⊕ 4⊕ 7⊕$$

依次类推…

需要注意的就是，我们的message即R总长64位，当大于64位以后，需要的就是我们补上的lastbit

```
output ^= lastbit 由这里可知
```

未知数位为30位，已知Lastbit为35位，即可列出35个方程。可解

这里用一下z3约束器去解方程，列方程的过程用上了正则表达式以及各种Ctrl+H替换

需要注意的是：

为了想少定义一些变量，想着能不能通过下标的形式，定义数组来简洁定义变量。

```
"""so.add(a[1]^a[3]^a[6]^a[13]^a[21]^a[25]^a[28]^a[30]^a[33]^a[35]^a[38]^a[45]^a[53]^a[57]^a[60]^a[62]==0)
so.add(a[2]^a[4]^a[7]^a[14]^a[22]^a[26]^a[29]^a[31]^a[34]^a[36]^a[39]^a[46]^a[54]^a[58]^a[61]^a[63]==1)
so.add(a[3]^a[5]^a[8]^a[15]^a[23]^a[27]^a[30]^a[32]^a[35]^a[37]^a[40]^a[47]^a[55]^a[59]^a[62]^a[64]==1)
so.add(a[4]^a[6]^a[9]^a[16]^a[24]^a[28]^a[31]^a[33]^a[36]^a[38]^a[41]^a[48]^a[56]^a[60]^a[63]^b[1]==1)
so.add(a[5]^a[7]^a[10]^a[17]^a[25]^a[29]^a[32]^a[34]^a[37]^a[39]^a[42]^a[49]^a[57]^a[61]^a[64]^b[2]==1)
so.add(a[6]^a[8]^a[11]^a[18]^a[26]^a[30]^a[33]^a[35]^a[38]^a[40]^a[43]^a[50]^a[58]^a[62]^b[1]^b[3]==1)
so.add(a[7]^a[9]^a[12]^a[19]^a[27]^a[31]^a[34]^a[36]^a[39]^a[41]^a[44]^a[51]^a[59]^a[63]^b[2]^b[4]==0)
so.add(a[8]^a[10]^a[13]^a[20]^a[28]^a[32]^a[35]^a[37]^a[40]^a[42]^a[45]^a[52]^a[60]^a[64]^b[3]^b[5]==1)
so.add(a[9]^a[11]^a[14]^a[21]^a[29]^a[33]^a[36]^a[38]^a[41]^a[43]^a[46]^a[53]^a[61]^b[1]^b[4]^b[6]==1)
so.add(a[10]^a[12]^a[15]^a[22]^a[30]^a[34]^a[37]^a[39]^a[42]^a[44]^a[47]^a[54]^a[62]^b[2]^b[5]^b[7]==1)
so.add(a[11]^a[13]^a[16]^a[23]^a[31]^a[35]^a[38]^a[40]^a[43]^a[45]^a[48]^a[55]^a[63]^b[3]^b[6]^b[8]==1)
so.add(a[12]^a[14]^a[17]^a[24]^a[32]^a[36]^a[39]^a[41]^a[44]^a[46]^a[49]^a[56]^a[64]^b[4]^b[7]^b[9]==0)
so.add(a[13]^a[15]^a[18]^a[25]^a[33]^a[37]^a[40]^a[42]^a[45]^a[47]^a[50]^a[57]^b[1]^b[5]^b[8]^b[10]==1)
so.add(a[14]^a[16]^a[19]^a[26]^a[34]^a[38]^a[41]^a[43]^a[46]^a[48]^a[51]^a[58]^b[2]^b[6]^b[9]^b[11]==0)
so.add(a[15]^a[17]^a[20]^a[27]^a[35]^a[39]^a[42]^a[44]^a[47]^a[49]^a[52]^a[59]^b[3]^b[7]^b[10]^b[12]==1)
so.add(a[16]^a[18]^a[21]^a[28]^a[36]^a[40]^a[43]^a[45]^a[48]^a[50]^a[53]^a[60]^b[4]^b[8]^b[11]^b[13]==1)
so.add(a[17]^a[19]^a[22]^a[29]^a[37]^a[41]^a[44]^a[46]^a[49]^a[51]^a[54]^a[61]^b[5]^b[9]^b[12]^b[14]==1)
so.add(a[18]^a[20]^a[23]^a[30]^a[38]^a[42]^a[45]^a[47]^a[50]^a[52]^a[55]^a[62]^b[6]^b[10]^b[13]^b[15]==0)
so.add(a[19]^a[21]^a[24]^a[31]^a[39]^a[43]^a[46]^a[48]^a[51]^a[53]^a[56]^a[63]^b[7]^b[11]^b[14]^b[16]==0)
so.add(a[20]^a[22]^a[25]^a[32]^a[40]^a[44]^a[47]^a[49]^a[52]^a[54]^a[57]^a[64]^b[8]^b[12]^b[15]^b[17]==0)
so.add(a[21]^a[23]^a[26]^a[33]^a[41]^a[45]^a[48]^a[50]^a[53]^a[55]^a[58]^b[1]^b[9]^b[13]^b[16]^b[18]==0)
so.add(a[22]^a[24]^a[27]^a[34]^a[42]^a[46]^a[49]^a[51]^a[54]^a[56]^a[59]^b[2]^b[10]^b[14]^b[17]^b[19]==1)
so.add(a[23]^a[25]^a[28]^a[35]^a[43]^a[47]^a[50]^a[52]^a[55]^a[57]^a[60]^b[3]^b[11]^b[15]^b[18]^b[20]==0)
so.add(a[24]^a[26]^a[29]^a[36]^a[44]^a[48]^a[51]^a[53]^a[56]^a[58]^a[61]^b[4]^b[12]^b[16]^b[19]^b[21]==0)
so.add(a[25]^a[27]^a[30]^a[37]^a[45]^a[49]^a[52]^a[54]^a[57]^a[59]^a[62]^b[5]^b[13]^b[17]^b[20]^b[22]==1)
so.add(a[26]^a[28]^a[31]^a[38]^a[46]^a[50]^a[53]^a[55]^a[58]^a[60]^a[63]^b[6]^b[14]^b[18]^b[21]^b[23]==0)
so.add(a[27]^a[29]^a[32]^a[39]^a[47]^a[51]^a[54]^a[56]^a[59]^a[61]^a[64]^b[7]^b[15]^b[19]^b[22]^b[24]==1)
so.add(a[28]^a[30]^a[33]^a[40]^a[48]^a[52]^a[55]^a[57]^a[60]^a[62]^b[1]^b[8]^b[16]^b[20]^b[23]^b[25]==1)
so.add(a[29]^a[31]^a[34]^a[41]^a[49]^a[53]^a[56]^a[58]^a[61]^a[63]^b[2]^b[9]^b[17]^b[21]^b[24]^b[26]==1)
so.add(a[30]^a[32]^a[35]^a[42]^a[50]^a[54]^a[57]^a[59]^a[62]^a[64]^b[3]^b[10]^b[18]^b[22]^b[25]^b[27]==0)
so.add(a[31]^a[33]^a[36]^a[43]^a[51]^a[55]^a[58]^a[60]^a[63]^b[1]^b[4]^b[11]^b[19]^b[23]^b[26]^b[28]==0)
so.add(a[32]^a[34]^a[37]^a[44]^a[52]^a[56]^a[59]^a[61]^a[64]^b[2]^b[5]^b[12]^b[20]^b[24]^b[27]^b[29]==1)
so.add(a[33]^a[35]^a[38]^a[45]^a[53]^a[57]^a[60]^a[62]^b[1]^b[3]^b[6]^b[13]^b[21]^b[25]^b[28]^b[30]==1)
so.add(a[34]^a[36]^a[39]^a[46]^a[54]^a[58]^a[61]^a[63]^b[2]^b[4]^b[7]^b[14]^b[22]^b[26]^b[29]^b[31]==0)
so.add(a[35]^a[37]^a[40]^a[47]^a[55]^a[59]^a[62]^a[64]^b[3]^b[5]^b[8]^b[15]^b[23]^b[27]^b[30]^b[32]==1)"""
```

于是我加上了这样的条件

但最后告诉我不行，所以我就把[]删掉，直接定义变量如：a12,a23,a33,a44

```
"""import re
with open(r"D:\\Mango\\D3CTF\\Crypto\\D^3ctf\\Crypto\\d3bug\\D3BUG\\this.txt") as f:
    r=f.read()
print(r)
print(re.sub(']','',re.sub('\[','',r)))
f.close()"""
```

变成了这样子：

```
so.add((a1^a3^a6^a13^a21^a25^a28^a30^a33^a35^a38^a45^a53^a57^a60^a62)==0)
so.add((a2^a4^a7^a14^a22^a26^a29^a31^a34^a36^a39^a46^a54^a58^a61^a63)==1)
so.add((a3^a5^a8^a15^a23^a27^a30^a32^a35^a37^a40^a47^a55^a59^a62^a64)==1)
so.add((a4^a6^a9^a16^a24^a28^a31^a33^a36^a38^a41^a48^a56^a60^a63^b1)==1)
so.add((a5^a7^a10^a17^a25^a29^a32^a34^a37^a39^a42^a49^a57^a61^a64^b2)==1)
so.add((a6^a8^a11^a18^a26^a30^a33^a35^a38^a40^a43^a50^a58^a62^b1^b3)==1)
so.add((a7^a9^a12^a19^a27^a31^a34^a36^a39^a41^a44^a51^a59^a63^b2^b4)==0)
so.add((a8^a10^a13^a20^a28^a32^a35^a37^a40^a42^a45^a52^a60^a64^b3^b5)==1)
so.add((a9^a11^a14^a21^a29^a33^a36^a38^a41^a43^a46^a53^a61^b1^b4^b6)==1)
so.add((a10^a12^a15^a22^a30^a34^a37^a39^a42^a44^a47^a54^a62^b2^b5^b7)==1)
so.add((a11^a13^a16^a23^a31^a35^a38^a40^a43^a45^a48^a55^a63^b3^b6^b8)==1)
so.add((a12^a14^a17^a24^a32^a36^a39^a41^a44^a46^a49^a56^a64^b4^b7^b9)==0)
so.add((a13^a15^a18^a25^a33^a37^a40^a42^a45^a47^a50^a57^b1^b5^b8^b10)==1)
so.add((a14^a16^a19^a26^a34^a38^a41^a43^a46^a48^a51^a58^b2^b6^b9^b11)==0)
so.add((a15^a17^a20^a27^a35^a39^a42^a44^a47^a49^a52^a59^b3^b7^b10^b12)==1)
so.add((a16^a18^a21^a28^a36^a40^a43^a45^a48^a50^a53^a60^b4^b8^b11^b13)==1)
so.add((a17^a19^a22^a29^a37^a41^a44^a46^a49^a51^a54^a61^b5^b9^b12^b14)==1)
so.add((a18^a20^a23^a30^a38^a42^a45^a47^a50^a52^a55^a62^b6^b10^b13^b15)==0)
so.add((a19^a21^a24^a31^a39^a43^a46^a48^a51^a53^a56^a63^b7^b11^b14^b16)==0)
so.add((a20^a22^a25^a32^a40^a44^a47^a49^a52^a54^a57^a64^b8^b12^b15^b17)==0)
so.add((a21^a23^a26^a33^a41^a45^a48^a50^a53^a55^a58^b1^b9^b13^b16^b18)==0)
so.add((a22^a24^a27^a34^a42^a46^a49^a51^a54^a56^a59^b2^b10^b14^b17^b19)==1)
so.add((a23^a25^a28^a35^a43^a47^a50^a52^a55^a57^a60^b3^b11^b15^b18^b20)==0)
so.add((a24^a26^a29^a36^a44^a48^a51^a53^a56^a58^a61^b4^b12^b16^b19^b21)==0)
so.add((a25^a27^a30^a37^a45^a49^a52^a54^a57^a59^a62^b5^b13^b17^b20^b22)==1)
so.add((a26^a28^a31^a38^a46^a50^a53^a55^a58^a60^a63^b6^b14^b18^b21^b23)==0)
so.add((a27^a29^a32^a39^a47^a51^a54^a56^a59^a61^a64^b7^b15^b19^b22^b24)==1)
so.add((a28^a30^a33^a40^a48^a52^a55^a57^a60^a62^b1^b8^b16^b20^b23^b25)==1)
so.add((a29^a31^a34^a41^a49^a53^a56^a58^a61^a63^b2^b9^b17^b21^b24^b26)==1)
so.add((a30^a32^a35^a42^a50^a54^a57^a59^a62^a64^b3^b10^b18^b22^b25^b27)==0)
so.add((a31^a33^a36^a43^a51^a55^a58^a60^a63^b1^b4^b11^b19^b23^b26^b28)==0)
so.add((a32^a34^a37^a44^a52^a56^a59^a61^a64^b2^b5^b12^b20^b24^b27^b29)==1)
so.add((a33^a35^a38^a45^a53^a57^a60^a62^b1^b3^b6^b13^b21^b25^b28^b30)==1)
so.add((a34^a36^a39^a46^a54^a58^a61^a63^b2^b4^b7^b14^b22^b26^b29^b31)==0)
so.add((a35^a37^a40^a47^a55^a59^a62^a64^b3^b5^b8^b15^b23^b27^b30^b32)==1)
```

然后加上定义：

```
a1=BitVec('a1',1)
a2=BitVec('a2',1)
a3=BitVec('a3',1)
a4=BitVec('a4',1)
a5=BitVec('a5',1)
a6=BitVec('a6',1)
a7=BitVec('a7',1)
a8=BitVec('a8',1)
a9=BitVec('a9',1)
a10=BitVec('a10',1)
a11=BitVec('a11',1)
a12=BitVec('a12',1)
a13=BitVec('a13',1)
a14=BitVec('a14',1)
a15=BitVec('a15',1)
a16=BitVec('a16',1)
a17=BitVec('a17',1)
a18=BitVec('a18',1)
a19=BitVec('a19',1)
a20=BitVec('a20',1)
a21=BitVec('a21',1)
a22=BitVec('a22',1)
```

```
a22=BitVec('a22',1)
a23=BitVec('a23',1)
a24=BitVec('a24',1)
a25=BitVec('a25',1)
a26=BitVec('a26',1)
a27=BitVec('a27',1)
a28=BitVec('a28',1)
a29=BitVec('a29',1)
a30=BitVec('a30',1)
a31=BitVec('a31',1)
a32=BitVec('a32',1)
a33=BitVec('a33',1)
a34=BitVec('a34',1)
a35=BitVec('a35',1)
a36=BitVec('a36',1)
a37=BitVec('a37',1)
a38=BitVec('a38',1)
a39=BitVec('a39',1)
a40=BitVec('a40',1)
a41=BitVec('a41',1)
a42=BitVec('a42',1)
a43=BitVec('a43',1)
a44=BitVec('a44',1)
a45=BitVec('a45',1)
a46=BitVec('a46',1)
a47=BitVec('a47',1)
a48=BitVec('a48',1)
a49=BitVec('a49',1)
a50=BitVec('a50',1)
a51=BitVec('a51',1)
a52=BitVec('a52',1)
a53=BitVec('a53',1)
a54=BitVec('a54',1)
a55=BitVec('a55',1)
a56=BitVec('a56',1)
a57=BitVec('a57',1)
a58=BitVec('a58',1)
a59=BitVec('a59',1)
a60=BitVec('a60',1)
a61=BitVec('a61',1)
a62=BitVec('a62',1)
a63=BitVec('a63',1)
a64=BitVec('a64',1)
b1=BitVec('b1',1)
b2=BitVec('b2',1)
b3=BitVec('b3',1)
b4=BitVec('b4',1)
b5=BitVec('b5',1)
b6=BitVec('b6',1)
b7=BitVec('b7',1)
b8=BitVec('b8',1)
b9=BitVec('b9',1)
b10=BitVec('b10',1)
b11=BitVec('b11',1)
b12=BitVec('b12',1)
b13=BitVec('b13',1)
b14=BitVec('b14',1)
b15=BitVec('b15',1)
b16=BitVec('b16',1)
b17=BitVec('b17',1)
```

```
b18=BitVec('b18',1)
b19=BitVec('b19',1)
b20=BitVec('b20',1)
b21=BitVec('b21',1)
b22=BitVec('b22',1)
b23=BitVec('b23',1)
b24=BitVec('b24',1)
b25=BitVec('b25',1)
b26=BitVec('b26',1)
b27=BitVec('b27',1)
b28=BitVec('b28',1)
b29=BitVec('b29',1)
b30=BitVec('b30',1)
b31=BitVec('b31',1)
b32=BitVec('b32',1)
b33=BitVec('b33',1)
b34=BitVec('b34',1)
b35=BitVec('b35',1)
so=Solver()
so.add(a1==0)
so.add(a2==1)
so.add(a3==0)
so.add(a4==0)
so.add(a5==1)
so.add(a6==1)
so.add(a7==0)
so.add(a8==0)
so.add(a9==0)
so.add(a10==1)
so.add(a11==0)
so.add(a12==0)
so.add(a13==0)
so.add(a14==1)
so.add(a15==1)
so.add(a16==0)
so.add(a17==0)
so.add(a18==0)
so.add(a19==1)
so.add(a20==1)
so.add(a21==0)
so.add(a22==1)
so.add(a23==0)
so.add(a24==1)
so.add(a25==0)
so.add(a26==1)
so.add(a27==0)
so.add(a28==1)
so.add(a29==0)
so.add(a30==0)
so.add(a31==1)
so.add(a32==0)
so.add(a33==0)
so.add(a34==1)
so.add(b1==0)
so.add(b2==1)
so.add(b3==1)
so.add(b4==1)
so.add(b5==1)
so.add(b6==1)
```

```
so.add(b7==0)
so.add(b8==1)
so.add(b9==1)
so.add(b10==1)
so.add(b11==1)
so.add(b12==0)
so.add(b13==1)
so.add(b14==0)
so.add(b15==1)
so.add(b16==1)
so.add(b17==1)
so.add(b18==0)
so.add(b19==0)
so.add(b20==0)
so.add(b21==0)
so.add(b22==1)
so.add(b23==0)
so.add(b24==0)
so.add(b25==1)
so.add(b26==0)
so.add(b27==1)
so.add(b28==1)
so.add(b29==1)
so.add(b30==0)
so.add(b31==0)
so.add(b32==1)
so.add(b33==1)
so.add(b34==0)
so.add(b35==1)
产生方法（举一个）：
e="01001100010001100011010101010001001"
bb="011111011110101110000100101110011101"
for i in range(1,36):
    k=i-1
    print(f"so.add(b{i}=={bb[k]})")
```

然后就可以完整的通过z3解出来啦！

完整z3解低位：

```
from z3 import *
import re
a1=BitVec('a1',1)
a2=BitVec('a2',1)
a3=BitVec('a3',1)
a4=BitVec('a4',1)
a5=BitVec('a5',1)
a6=BitVec('a6',1)
a7=BitVec('a7',1)
a8=BitVec('a8',1)
a9=BitVec('a9',1)
a10=BitVec('a10',1)
a11=BitVec('a11',1)
a12=BitVec('a12',1)
a13=BitVec('a13',1)
a14=BitVec('a14',1)
a15=BitVec('a15',1)
a16=BitVec('a16',1)
a17=BitVec('a17',1)
a18=BitVec('a18',1)
```

```
a19=BitVec('a19',1)
a20=BitVec('a20',1)
a21=BitVec('a21',1)
a22=BitVec('a22',1)
a23=BitVec('a23',1)
a24=BitVec('a24',1)
a25=BitVec('a25',1)
a26=BitVec('a26',1)
a27=BitVec('a27',1)
a28=BitVec('a28',1)
a29=BitVec('a29',1)
a30=BitVec('a30',1)
a31=BitVec('a31',1)
a32=BitVec('a32',1)
a33=BitVec('a33',1)
a34=BitVec('a34',1)
a35=BitVec('a35',1)
a36=BitVec('a36',1)
a37=BitVec('a37',1)
a38=BitVec('a38',1)
a39=BitVec('a39',1)
a40=BitVec('a40',1)
a41=BitVec('a41',1)
a42=BitVec('a42',1)
a43=BitVec('a43',1)
a44=BitVec('a44',1)
a45=BitVec('a45',1)
a46=BitVec('a46',1)
a47=BitVec('a47',1)
a48=BitVec('a48',1)
a49=BitVec('a49',1)
a50=BitVec('a50',1)
a51=BitVec('a51',1)
a52=BitVec('a52',1)
a53=BitVec('a53',1)
a54=BitVec('a54',1)
a55=BitVec('a55',1)
a56=BitVec('a56',1)
a57=BitVec('a57',1)
a58=BitVec('a58',1)
a59=BitVec('a59',1)
a60=BitVec('a60',1)
a61=BitVec('a61',1)
a62=BitVec('a62',1)
a63=BitVec('a63',1)
a64=BitVec('a64',1)
b1=BitVec('b1',1)
b2=BitVec('b2',1)
b3=BitVec('b3',1)
b4=BitVec('b4',1)
b5=BitVec('b5',1)
b6=BitVec('b6',1)
b7=BitVec('b7',1)
b8=BitVec('b8',1)
b9=BitVec('b9',1)
b10=BitVec('b10',1)
b11=BitVec('b11',1)
b12=BitVec('b12',1)
b13=BitVec('b13',1)
```

```
b14=BitVec('b14',1)
b15=BitVec('b15',1)
b16=BitVec('b16',1)
b17=BitVec('b17',1)
b18=BitVec('b18',1)
b19=BitVec('b19',1)
b20=BitVec('b20',1)
b21=BitVec('b21',1)
b22=BitVec('b22',1)
b23=BitVec('b23',1)
b24=BitVec('b24',1)
b25=BitVec('b25',1)
b26=BitVec('b26',1)
b27=BitVec('b27',1)
b28=BitVec('b28',1)
b29=BitVec('b29',1)
b30=BitVec('b30',1)
b31=BitVec('b31',1)
b32=BitVec('b32',1)
b33=BitVec('b33',1)
b34=BitVec('b34',1)
b35=BitVec('b35',1)
so=Solver()
so.add(a1==0)
so.add(a2==1)
so.add(a3==0)
so.add(a4==0)
so.add(a5==1)
so.add(a6==1)
so.add(a7==0)
so.add(a8==0)
so.add(a9==0)
so.add(a10==1)
so.add(a11==0)
so.add(a12==0)
so.add(a13==0)
so.add(a14==1)
so.add(a15==1)
so.add(a16==0)
so.add(a17==0)
so.add(a18==0)
so.add(a19==1)
so.add(a20==1)
so.add(a21==0)
so.add(a22==1)
so.add(a23==0)
so.add(a24==1)
so.add(a25==0)
so.add(a26==1)
so.add(a27==0)
so.add(a28==1)
so.add(a29==0)
so.add(a30==0)
so.add(a31==1)
so.add(a32==0)
so.add(a33==0)
so.add(a34==1)
so.add(b1==0)
so.add(b2==1)
so.add(b3==1)
```

```
so.add(b4==1)
so.add(b5==1)
so.add(b6==1)
so.add(b7==0)
so.add(b8==1)
so.add(b9==1)
so.add(b10==1)
so.add(b11==1)
so.add(b12==0)
so.add(b13==1)
so.add(b14==0)
so.add(b15==1)
so.add(b16==1)
so.add(b17==1)
so.add(b18==0)
so.add(b19==0)
so.add(b20==0)
so.add(b21==0)
so.add(b22==1)
so.add(b23==0)
so.add(b24==0)
so.add(b25==1)
so.add(b26==0)
so.add(b27==1)
so.add(b28==1)
so.add(b29==1)
so.add(b30==0)
so.add(b31==0)
so.add(b32==1)
so.add(b33==1)
so.add(b34==0)
so.add(b35==1)
"""so.add(a[1]^a[3]^a[6]^a[13]^a[21]^a[25]^a[28]^a[30]^a[33]^a[35]^a[38]^a[45]^a[53]^a[57]^a[60]^a[62]==0)
so.add(a[2]^a[4]^a[7]^a[14]^a[22]^a[26]^a[29]^a[31]^a[34]^a[36]^a[39]^a[46]^a[54]^a[58]^a[61]^a[63]==1)
so.add(a[3]^a[5]^a[8]^a[15]^a[23]^a[27]^a[30]^a[32]^a[35]^a[37]^a[40]^a[47]^a[55]^a[59]^a[62]^a[64]==1)
so.add(a[4]^a[6]^a[9]^a[16]^a[24]^a[28]^a[31]^a[33]^a[36]^a[38]^a[41]^a[48]^a[56]^a[60]^a[63]^b[1]==1)
so.add(a[5]^a[7]^a[10]^a[17]^a[25]^a[29]^a[32]^a[34]^a[37]^a[39]^a[42]^a[49]^a[57]^a[61]^a[64]^b[2]==1)
so.add(a[6]^a[8]^a[11]^a[18]^a[26]^a[30]^a[33]^a[35]^a[38]^a[40]^a[43]^a[50]^a[58]^a[62]^b[1]^b[3]==1)
so.add(a[7]^a[9]^a[12]^a[19]^a[27]^a[31]^a[34]^a[36]^a[39]^a[41]^a[44]^a[51]^a[59]^a[63]^b[2]^b[4]==0)
so.add(a[8]^a[10]^a[13]^a[20]^a[28]^a[32]^a[35]^a[37]^a[40]^a[42]^a[45]^a[52]^a[60]^a[64]^b[3]^b[5]==1)
so.add(a[9]^a[11]^a[14]^a[21]^a[29]^a[33]^a[36]^a[38]^a[41]^a[43]^a[46]^a[53]^a[61]^b[1]^b[4]^b[6]==1)
so.add(a[10]^a[12]^a[15]^a[22]^a[30]^a[34]^a[37]^a[39]^a[42]^a[44]^a[47]^a[54]^a[62]^b[2]^b[5]^b[7]==1)
so.add(a[11]^a[13]^a[16]^a[23]^a[31]^a[35]^a[38]^a[40]^a[43]^a[45]^a[48]^a[55]^a[63]^b[3]^b[6]^b[8]==1)
so.add(a[12]^a[14]^a[17]^a[24]^a[32]^a[36]^a[39]^a[41]^a[44]^a[46]^a[49]^a[56]^a[64]^b[4]^b[7]^b[9]==0)
so.add(a[13]^a[15]^a[18]^a[25]^a[33]^a[37]^a[40]^a[42]^a[45]^a[47]^a[50]^a[57]^b[1]^b[5]^b[8]^b[10]==1)
so.add(a[14]^a[16]^a[19]^a[26]^a[34]^a[38]^a[41]^a[43]^a[46]^a[48]^a[51]^a[58]^b[2]^b[6]^b[9]^b[11]==0)
so.add(a[15]^a[17]^a[20]^a[27]^a[35]^a[39]^a[42]^a[44]^a[47]^a[49]^a[52]^a[59]^b[3]^b[7]^b[10]^b[12]==1)
so.add(a[16]^a[18]^a[21]^a[28]^a[36]^a[40]^a[43]^a[45]^a[48]^a[50]^a[53]^a[60]^b[4]^b[8]^b[11]^b[13]==1)
so.add(a[17]^a[19]^a[22]^a[29]^a[37]^a[41]^a[44]^a[46]^a[49]^a[51]^a[54]^a[61]^b[5]^b[9]^b[12]^b[14]==1)
so.add(a[18]^a[20]^a[23]^a[30]^a[38]^a[42]^a[45]^a[47]^a[50]^a[52]^a[55]^a[62]^b[6]^b[10]^b[13]^b[15]==0)
so.add(a[19]^a[21]^a[24]^a[31]^a[39]^a[43]^a[46]^a[48]^a[51]^a[53]^a[56]^a[63]^b[7]^b[11]^b[14]^b[16]==0)
so.add(a[20]^a[22]^a[25]^a[32]^a[40]^a[44]^a[47]^a[49]^a[52]^a[54]^a[57]^a[64]^b[8]^b[12]^b[15]^b[17]==0)
so.add(a[21]^a[23]^a[26]^a[33]^a[41]^a[45]^a[48]^a[50]^a[53]^a[55]^a[58]^b[1]^b[9]^b[13]^b[16]^b[18]==0)
so.add(a[22]^a[24]^a[27]^a[34]^a[42]^a[46]^a[49]^a[51]^a[54]^a[56]^a[59]^b[2]^b[10]^b[14]^b[17]^b[19]==1)
so.add(a[23]^a[25]^a[28]^a[35]^a[43]^a[47]^a[50]^a[52]^a[55]^a[57]^a[60]^b[3]^b[11]^b[15]^b[18]^b[20]==0)
so.add(a[24]^a[26]^a[29]^a[36]^a[44]^a[48]^a[51]^a[53]^a[56]^a[58]^a[61]^b[4]^b[12]^b[16]^b[19]^b[21]==0)
so.add(a[25]^a[27]^a[30]^a[37]^a[45]^a[49]^a[52]^a[54]^a[57]^a[59]^a[62]^b[5]^b[13]^b[17]^b[20]^b[22]==1)
so.add(a[26]^a[28]^a[31]^a[38]^a[46]^a[50]^a[53]^a[55]^a[58]^a[60]^a[63]^b[6]^b[14]^b[18]^b[21]^b[23]==0)
so.add(a[27]^a[29]^a[32]^a[39]^a[47]^a[51]^a[54]^a[56]^a[59]^a[61]^a[64]^b[7]^b[15]^b[19]^b[22]^b[24]==1)
```

```
so.add(a[28]^a[30]^a[33]^a[40]^a[48]^a[52]^a[55]^a[57]^a[60]^a[62]^b[1]^b[8]^b[16]^b[20]^b[23]^b[25]==1)
so.add(a[29]^a[31]^a[34]^a[41]^a[49]^a[53]^a[56]^a[58]^a[61]^a[63]^b[2]^b[9]^b[17]^b[21]^b[24]^b[26]==1)
so.add(a[30]^a[32]^a[35]^a[42]^a[50]^a[54]^a[57]^a[59]^a[62]^a[64]^b[3]^b[10]^b[18]^b[22]^b[25]^b[27]==0)
so.add(a[31]^a[33]^a[36]^a[43]^a[51]^a[55]^a[58]^a[60]^a[63]^b[1]^b[4]^b[11]^b[19]^b[23]^b[26]^b[28]==0)
so.add(a[32]^a[34]^a[37]^a[44]^a[52]^a[56]^a[59]^a[61]^a[64]^b[2]^b[5]^b[12]^b[20]^b[24]^b[27]^b[29]==1)
so.add(a[33]^a[35]^a[38]^a[45]^a[53]^a[57]^a[60]^a[62]^b[1]^b[3]^b[6]^b[13]^b[21]^b[25]^b[28]^b[30]==1)
so.add(a[34]^a[36]^a[39]^a[46]^a[54]^a[58]^a[61]^a[63]^b[2]^b[4]^b[7]^b[14]^b[22]^b[26]^b[29]^b[31]==0)
so.add(a[35]^a[37]^a[40]^a[47]^a[55]^a[59]^a[62]^a[64]^b[3]^b[5]^b[8]^b[15]^b[23]^b[27]^b[30]^b[32]==1)"""
so.add((a1^a3^a6^a13^a21^a25^a28^a30^a33^a35^a38^a45^a53^a57^a60^a62)==0)
so.add((a2^a4^a7^a14^a22^a26^a29^a31^a34^a36^a39^a46^a54^a58^a61^a63)==1)
so.add((a3^a5^a8^a15^a23^a27^a30^a32^a35^a37^a40^a47^a55^a59^a62^a64)==1)
so.add((a4^a6^a9^a16^a24^a28^a31^a33^a36^a38^a41^a48^a56^a60^a63^b1)==1)
so.add((a5^a7^a10^a17^a25^a29^a32^a34^a37^a39^a42^a49^a57^a61^a64^b2)==1)
so.add((a6^a8^a11^a18^a26^a30^a33^a35^a38^a40^a43^a50^a58^a62^b1^b3)==1)
so.add((a7^a9^a12^a19^a27^a31^a34^a36^a39^a41^a44^a51^a59^a63^b2^b4)==0)
so.add((a8^a10^a13^a20^a28^a32^a35^a37^a40^a42^a45^a52^a60^a64^b3^b5)==1)
so.add((a9^a11^a14^a21^a29^a33^a36^a38^a41^a43^a46^a53^a61^b1^b4^b6)==1)
so.add((a10^a12^a15^a22^a30^a34^a37^a39^a42^a44^a47^a54^a62^b2^b5^b7)==1)
so.add((a11^a13^a16^a23^a31^a35^a38^a40^a43^a45^a48^a55^a63^b3^b6^b8)==1)
so.add((a12^a14^a17^a24^a32^a36^a39^a41^a44^a46^a49^a56^a64^b4^b7^b9)==0)
so.add((a13^a15^a18^a25^a33^a37^a40^a42^a45^a47^a50^a57^b1^b5^b8^b10)==1)
so.add((a14^a16^a19^a26^a34^a38^a41^a43^a46^a48^a51^a58^b2^b6^b9^b11)==0)
so.add((a15^a17^a20^a27^a35^a39^a42^a44^a47^a49^a52^a59^b3^b7^b10^b12)==1)
so.add((a16^a18^a21^a28^a36^a40^a43^a45^a48^a50^a53^a60^b4^b8^b11^b13)==1)
so.add((a17^a19^a22^a29^a37^a41^a44^a46^a49^a51^a54^a61^b5^b9^b12^b14)==1)
so.add((a18^a20^a23^a30^a38^a42^a45^a47^a50^a52^a55^a62^b6^b10^b13^b15)==0)
so.add((a19^a21^a24^a31^a39^a43^a46^a48^a51^a53^a56^a63^b7^b11^b14^b16)==0)
so.add((a20^a22^a25^a32^a40^a44^a47^a49^a52^a54^a57^a64^b8^b12^b15^b17)==0)
so.add((a21^a23^a26^a33^a41^a45^a48^a50^a53^a55^a58^b1^b9^b13^b16^b18)==0)
so.add((a22^a24^a27^a34^a42^a46^a49^a51^a54^a56^a59^b2^b10^b14^b17^b19)==1)
so.add((a23^a25^a28^a35^a43^a47^a50^a52^a55^a57^a60^b3^b11^b15^b18^b20)==0)
so.add((a24^a26^a29^a36^a44^a48^a51^a53^a56^a58^a61^b4^b12^b16^b19^b21)==0)
so.add((a25^a27^a30^a37^a45^a49^a52^a54^a57^a59^a62^b5^b13^b17^b20^b22)==1)
so.add((a26^a28^a31^a38^a46^a50^a53^a55^a58^a60^a63^b6^b14^b18^b21^b23)==0)
so.add((a27^a29^a32^a39^a47^a51^a54^a56^a59^a61^a64^b7^b15^b19^b22^b24)==1)
so.add((a28^a30^a33^a40^a48^a52^a55^a57^a60^a62^b1^b8^b16^b20^b23^b25)==1)
so.add((a29^a31^a34^a41^a49^a53^a56^a58^a61^a63^b2^b9^b17^b21^b24^b26)==1)
so.add((a30^a32^a35^a42^a50^a54^a57^a59^a62^a64^b3^b10^b18^b22^b25^b27)==0)
so.add((a31^a33^a36^a43^a51^a55^a58^a60^a63^b1^b4^b11^b19^b23^b26^b28)==0)
so.add((a32^a34^a37^a44^a52^a56^a59^a61^a64^b2^b5^b12^b20^b24^b27^b29)==1)
so.add((a33^a35^a38^a45^a53^a57^a60^a62^b1^b3^b6^b13^b21^b25^b28^b30)==1)
so.add((a34^a36^a39^a46^a54^a58^a61^a63^b2^b4^b7^b14^b22^b26^b29^b31)==0)
so.add((a35^a37^a40^a47^a55^a59^a62^a64^b3^b5^b8^b15^b23^b27^b30^b32)==1)
print(so.check())
print(so.model())
m=so.model()
"""for i in range(1,65):
    print(f"m[a{i}]")"""
print(m[a1],
m[a2],
m[a3],
m[a4],
m[a5],
m[a6],
m[a7],
m[a8],
m[a9],
m[a10],
m[a11],
m[a12],
```

```
m[a13],
m[a14],
m[a15],
m[a16],
m[a17],
m[a18],
m[a19],
m[a20],
m[a21],
m[a22],
m[a23],
m[a24],
m[a25],
m[a26],
m[a27],
m[a28],
m[a29],
m[a30],
m[a31],
m[a32],
m[a33],
m[a34],
m[a35],
m[a36],
m[a37],
m[a38],
m[a39],
m[a40],
m[a41],
m[a42],
m[a43],
m[a44],
m[a45],
m[a46],
m[a47],
m[a48],
m[a49],
m[a50],
m[a51],
m[a52],
m[a53],
m[a54],
m[a55],
m[a56],
m[a57],
m[a58],
m[a59],
m[a60],
m[a61],
m[a62],
m[a63],
m[a64])
final="0 1 0 0 1 1 0 0 0 1 0 0 0 1 1 0 0 0 1 1 0 1 0 1 0 1 0 1 0 0 1 0 0 1 1 1 0 0 1 1 0 1 1 1 0 1 0 1 0 1 1 0 1 0 1 1 0 0 1 0 0 0 0 1"
print(re.sub(" ","",final))
#0100110001000110001101010101001001110011011101010110101100100001
message =0b0100110001000110001101010101001001110011011101010110101100100001
from Crypto.Util.number import *
print(long_to_bytes(message))
"""sat
```

[a44 = 1,
a52 = 0,
a63 = 0,
a64 = 1,
a36 = 1,
a47 = 0,
a46 = 1,
a45 = 0,
a35 = 1,
a57 = 0,
a38 = 0,
a48 = 1,
a51 = 1,
a62 = 0,
a56 = 1,
a59 = 1,
a39 = 1,
a54 = 0,
a37 = 0,
a60 = 0,
a61 = 0,
a49 = 0,
a50 = 1,
a58 = 0,
a43 = 1,
a53 = 1,
a41 = 0,
a55 = 1,
a40 = 1,
a42 = 1,
b35 = 1,
b34 = 0,
b33 = 1,
b32 = 1,
b31 = 0,
b30 = 0,
b29 = 1,
b28 = 1,
b27 = 1,
b26 = 0,
b25 = 1,
b24 = 0,
b23 = 0,
b22 = 1,
b21 = 0,
b20 = 0,
b19 = 0,
b18 = 0,
b17 = 1,
b16 = 1,
b15 = 1,
b14 = 0,
b13 = 1,
b12 = 0,
b11 = 1,
b10 = 1,
b9 = 1,
b8 = 1,
b7 = 0,
b6 = 1

```
b5 = 1,
b4 = 1,
b3 = 1,
b2 = 1,
b1 = 0,
a34 = 1,
a33 = 0,
a32 = 0,
a31 = 1,
a30 = 0,
a29 = 0,
a28 = 1,
a27 = 0,
a26 = 1,
a25 = 0,
a24 = 1,
a23 = 0,
a22 = 1,
a21 = 0,
a20 = 1,
a19 = 1,
a18 = 0,
a17 = 0,
a16 = 0,
a15 = 1,
a14 = 1,
a13 = 0,
a12 = 0,
a11 = 0,
a10 = 1,
a9 = 0,
a8 = 0,
a7 = 0,
a6 = 1,
a5 = 1,
a4 = 0,
a3 = 0,
a2 = 1,
a1 = 0]
0 1 0 0 1 1 0 0 0 1 0 0 0 1 1 0 0 0 1 1 0 1 0 1 0 1 0 1 0 0 1 0 0 1 1 1 0 0 1 1 0 1 1 1 0 1 0 1 0 1 1 0 1 0 1 1 0 0 1 0 0 0 0 1
01001100010001100011010101010010011100110111010101101011001000001
b'LF5Rsuk!'
"""
```