【CTF WriteUp】2020网鼎杯第三场Crypto题解

原创

零食商人 于 2020-05-18 12:22:37 发布 1492 收藏 3

Crypto

simple

根据题目名称提示，本题为简单仿射密码，爆破得映射关系为：y = 8a + 22 (mod 26)，密钥使用方法为：123456 % 26 = 8，321564 % 26 = 22与爆破结果相同。于是得到正向替换表a-w b-e c-m d-u e-c f-k，反向替换得到flag。

RUA

题目给出了三组n和c，且三个n之间两两互质，所以本题只可能是利用中国剩余定理求出me，然后爆破e求解m。解题代码如下：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
import gmpy2
from libnum import n2s

c0 = 80246672933100191996608551744360551443480105561393008869907671453199197333698372068490702079554173569572543
31839203914525519504562595117422955140319552013305532068903324132309109484106720045613714716627620318471048195232
22096722129702695697906771444505013052896707835729192829097967651242422871087171897506627402838139812429186714722
89312649479614087741250236503718765990503419390163351636020898773132259997461260294586647775234008078329626839600
44532883548423045471565356810753599618810964317690395898263698123505876052304469769153374038403491084285836952033
49509780982492995977753061416719351469339586444564992002216960
n0 = 18856599160001833299560082802925753595735945621023660831294740454109973698430284916320395522883536507135735
38351792605096351244016248306509725688404093825909258289225965734082597126027838740639852916830942624153055139600
56450450728728601248269612166083300938497235910244979946020059799495231539400114422748104072550004260736766137350
45722528724371400634746032681469565707871430104412932683216410927430108056399531035789776682487265006361910439300
77003678731792837217993936051017943843666559175594022415613146027176391286832277460455831481211133569110888731980
27579162188169744014973478052491398688611046800951698773893393
c1 = 17388575106047489057419896548519877859896701790215215809457689651011062680688058437206227492035908101852130
41690197877374883285488889857682247724368287478468912770533424389996789632183668856760232355198698063488470004560
27950473546069670440078998428940082620044462222475031805594211784370238038168894827559017562364252406425134530710
99110577806920737600582033459363442698332069069996255809118560115646978112580099373145114105144167064825714718520
50375667541117708091635089944510600222639289564544321552267115531171563775961827630521746889207628737640151612460
40727839779731609203202530346427613422430202271506248285086956
n1 = 21996468204721630460566169654781925102402634427772676287751800587544894952838038401189546149401344752771866
37688222687607220142604169788202665377298764856905323845199287780881103454546336314605787964648546573031797773970
06776287970278094261290398668538232727000322458605289913900919015380904209692398479885177984131014170652915222060
22674484466421583941506570588463280334043092108362192416518829030837198227699471312835412997602835479387955740200
47885283904480355309382573044712679666823813157973591654623588972625718405890885290224142216992972089802562233650
08382492878690496154797198800699611812166851455110635853297883
c2 = 51708269421306583746272674705485493963288961086667170369993956265881548825313773936715939391927792921515846
78688653835775920356845071292462816417186595460417761844407911946323815187102170021222644920874070699813549492713
96766673681594782220086735346126457941920575650092621829460461669696918479337738162281838173335220245652400287633
63630446508265661263430432762725949426484083868720752967688204199776120400454905290081665834186798959333335663031
11753611684503882509908534560220564732967267289698948155748840638078043549523143917646181791475834478488712201030
94864884879810254237774776126305288789413579605152188117960 7
n2 = 22182114562385985868993176463839749402849876738564142471647983947408274900941377521795379832791801082248237
43213065802701138800963858797945093770302916822284284980198564604411646370340953193858041051109723893943128435210
09949200312466658018635489121157805030775386698514705824737070792739967925773549468095396944503293347398507980920
47470591807052690644410845771773162271627122493009004900145192131020709111050447923639355534223116839479410278467
79360829917046748301219913284968311264065891535939843729087279578335094414754634269328552000276041155464728425947
73777888584007026980376463757296179071968120796742375210877789

assert gmpy2.gcd(n0, n1) == 1
assert gmpy2.gcd(n0, n2) == 1
assert gmpy2.gcd(n1, n2) == 1

N = n0*n1*n2
s = c0 * (N//n0) * gmpy2.invert(N//n0, n0) + c1 * (N//n1) * gmpy2.invert(N//n1, n1) + c2 * (N//n2) * gmpy2.inver
t(N//n2, n2)
s = s % N
prime = 2
while True:
    if not gmpy2.iroot(s, prime)[1]:
        prime = gmpy2.next_prime(prime)
        continue
    print n2s(int(gmpy2.iroot(s, prime)[0]))
    break
```

Guess-game（更新）

查看题目代码，发现题目主要涉及两个问题：问题1是根据现有的d = (d*a+c) (mod b)算出下一个d，取得(500, 10)的硬币，否则无法继续往下做；问题2是如何利用有限的10个硬币确定r和key的值，使得后续问题能够全对。

首先看问题1。设7个d分别为d0、d1、…、d6，根据条件有

```
d1 = d0*a+c % b
d2 = d1*a+c % b
……
```

所以

```
d2-d1 = (d1-d0)*a % b
d3-d2 = (d2-d1)*a % b
d4-d3 = (d3-d2)*a % b
```

所以

```
b | (d2-d1)^2 - (d3-d2)*(d1-d0)
b | (d3-d2)^2 - (d4-d3)*(d2-d1)
b | (d4-d3)^2 - (d5-d4)*(d3-d2)
```

右边三个求最大公约数，即为b。然后带入原式得

```
a = (d2-d1)*(d1-d0)^-1 求得a
c = d1-d0*a % b 求得c
d6 = (d5*a+c) % b 求得d6
```

接下来看问题2。在我们有10个硬币的时候，可以允许9次错误机会。每次无论正确或错误，都可以知道当前r用过的连续8位，9次可以知道72位。由于算法流程类似求解mask的LFSR。有另一道类似的题目，其解答过程参考如下：

本题为 LFSR 中需要求解 mask 的情况。假设在 LFSR 中，每一次运算的规则为：

$$a_{n+1} = c_1*a_1 + c_2*a_2 + \ldots + c_n*a_n$$

那么我们多写几个：

$$a_{n+1} = c_1*a_1 + c_2*a_2 + \ldots + c_n*a_n$$

$$a_{n+2} = c_1*a_2 + c_2*a_3 + \ldots + c_n*a_{n+1}$$

$$\ldots\ldots$$

$$a_{2n} = c_1*a_n + c_2*a_{n+1} + \ldots + c_n*a_{2n-1}$$

即

$$\begin{bmatrix} c_1 & c_2 & \ldots & c_n \end{bmatrix} \bullet \begin{bmatrix} a_1 & a_2 & \ldots & a_n \\ a_2 & a_3 & \ldots & a_{n+1} \\ \ldots & \ldots & & \ldots \\ a_n & a_{n+1} & \ldots & a_{2n-1} \end{bmatrix} = \begin{bmatrix} a_{n+1} & a_{n+2} & \ldots & a_{2n} \end{bmatrix}$$

$$\begin{bmatrix} c_1 & c_2 & \ldots & c_n \end{bmatrix} = \begin{bmatrix} a_{n+1} & a_{n+2} & \ldots & a_{2n} \end{bmatrix} \bullet \begin{bmatrix} a_1 & a_2 & \ldots & a_n \\ a_2 & a_3 & \ldots & a_{n+1} \\ \ldots & \ldots & & \ldots \\ a_n & a_{n+1} & \ldots & a_{2n-1} \end{bmatrix}^{-1}$$

因此如果我们有 2n 位已知的情况，可以通过上边式子直接算出 mask，再倒推得到明文。题目的输出文件中只有 504 位，需要我们爆破 8 位，然后反推明文，再根据 sha256 的约束条件去判断哪个是真正的 flag。

本题非常糟糕的一点是，我们有40个变量，但是只有32个方程。即使爆破其中的8位，每种组合至少也有一个解，而当这些解作为mask时，每一个都能导出正确的后续32位，但33-40位各不相同。因此说到底，本题必须在前期撞大运正确一次，才有希望继续往下做。（一直刷，其实概率不低）

本地搭了一个，示意如下：

```
SEND: 123
Input your answer:
Wrong! The answer is 45!
Your coin: 7
SEND: 123
Input your answer:
Wrong! The answer is 165!
Your coin: 6
SEND: 123
Input your answer:
Wrong! The answer is 100!
Your coin: 5
SEND: 123
Input your answer:
Wrong! The answer is 158!
Your coin: 4
SEND: 123
Input your answer:
Right! The answer is 123!
Your coin: 5
SEND: 123
Input your answer:
Wrong! The answer is 145!
Your coin: 4
SEND: 123
Input your answer:
Wrong! The answer is 104!
11100110011111010010110110100101011001001001111001111011100100010110100 0
[*] Switching to interactive mode
Your coin: 3
Input your answer:
$
```

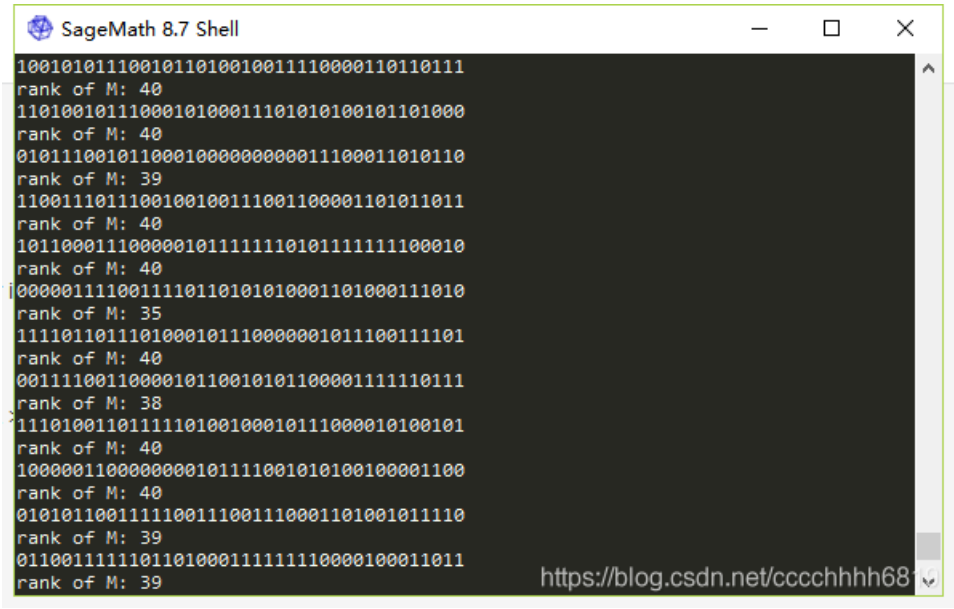

（接上图，只有蒙对一个以后，后续才能逐个求出来，这是因为可以允许多错一次，那我就知道了完整的40个变量40个方程，可以求解）

```
SEND: 123
Input your answer:
Wrong! The answer is 104!
11100110011111010010110110100101011001001001111001111011100100010110100 0
[*] Switching to interactive mode
Your coin: 3
Input your answer:
$ 1
Wrong! The answer is 101!
Your coin: 2
Input your answer:
$ 209
Right! The answer is 209!
Your coin: 3
Input your answer:
$ 100
Right! The answer is 100!
Your coin: 4
Input your answer:
$ 154
Right! The answer is 154!
Your coin: 5
Input your answer:
$ 225
Right! The answer is 225!
Your coin: 6
Input your answer:
$ 29
Right! The answer is 29!
Your coin: 7
Input your answer:
$
```


只有32个方程时的sage脚本：

```
from sage.all_cmdline import *
import hashlib

GF2 = GF(2);
def pad(m):
    pad_length = 8 - len(m)
    return pad_length*'0' + m

for x in range(256):
    a = '11100110011111010010110110100101011001001001111001111011100100010110100 0'
    a = a + pad(bin(x)[2:])

    A = []
    for i in range(40):
        A.append([int(op) for op in a[i:i+40]])
    A = matrix(GF2,A)

    if A.rank() != 40:
        continue
    last = a[40:]

    b = [int(op) for op in last]
    b = vector(GF2, b)


    mask = A.solve_right(b)
    sss = ''
    for x in range(40):
        sss += str(mask[x])
    print  sss
    mask = int(sss, 2)


    N = 40
    F = GF(2)

    b = a
    R = [vector(F, N) for i in range(N)]

    for i in range(N):
        R[i][N - 1] = mask >> (N-1 - i) & 1

    for i in range(N - 1):
        R[i + 1][i] = 1

    M = Matrix(F, R)
    M = M ** N
    vec = vector(F, N)
    row = 0
    for i in range(N / 8):
        t = int(a[i*8:i*8+8],2)
        for j in xrange(7, -1, -1):
            vec[row] = t >> j & 1
            row += 1

    print 'rank of M:',rank(M)
    if M.rank() != 40:
        continue
```

解题截图：



这一堆mask全是满足知道前40位能求出后边正确32位的，没法做。

再来看有40个方程时的sage脚本：

```
from sage.all_cmdline import *
import hashlib

GF2 = GF(2);
def pad(m):
    pad_length = 8 - len(m)
    return pad_length*'0' + m

for x in range(1):
    a = '11100110011111010010110110100101011001001001111001111011100100010110100001100101'
    #a = a + pad(bin(x)[2:])

    A = []
    for i in range(40):
        A.append([int(op) for op in a[i:i+40]])
    A = matrix(GF2,A)

    #if A.rank() != 40:
    #    continue
    last = a[40:]

    b = [int(op) for op in last]
    b = vector(GF2, b)


    mask = A.solve_right(b)
    sss = ''
    for x in range(40):
        sss += str(mask[x])
    print  sss
    mask = int(sss, 2)


    N = 40
    F = GF(2)

    b = a
    R = [vector(F, N) for i in range(N)]

    for i in range(N):
        R[i][N - 1] = mask >> (N-1 - i) & 1

    for i in range(N - 1):
        R[i + 1][i] = 1

    M = Matrix(F, R)
    M = M ** N
    vec = vector(F, N)
    row = 0
    for i in range(N / 8):
        t = int(a[i*8:i*8+8],2)
        for j in xrange(7, -1, -1):
            vec[row] = t >> j & 1
            row += 1

    print 'rank of M:',rank(M)
    if M.rank() != 40:
        continue
```

```
(sage-sh) Chainer@DESKTOP-FGPG11H:sagemath$ sage test.sage
(sage-sh) Chainer@DESKTOP-FGPG11H:sagemath$ sage test.sage
1100001010110111010100011011000101010100
rank of M: 40
(sage-sh) Chainer@DESKTOP-FGPG11H:sagemath$
```

就只剩一个了，然后随便写个程序就能继续往下求：

```
def solve(text, key):
    res = ''
    for k in range(40):
        text = text[-40:]
        tmp = 0
        for i in range(40):
            if(text[i]=='1')and(key[i]=='1'):
                tmp += 1
        res += str(tmp % 2)
        text = text + str(tmp % 2)
    print int(res[0:8],2)
    print int(res[8:16],2)
    print int(res[16:24],2)
    print int(res[24:32],2)
    print int(res[32:40],2)
    return res

text = "1001111001111011100100010110100001100101"
key  = "1100001010110111010100011011000101010100"
print solve(text[:40], key)
```

```
C:\Python27\python.exe C:/Users/Chainer/Desktop/temp.py
209
100
154
225
29
11010001011001001001101011100001000111 01

Process finished with exit code 0
```

就是刚才那张图一路right的结果。由于这题sage和python混用，本人菜鸡，只能输出500个数一个一个往里填拿到flag，这样也算是能做了吧。一键搞定的脚本我是写不出来，求大佬指点。