BUUCTF-CRYPTO 刷题记录（一）

原创

lysecl 于 2020-06-18 21:40:22 发布 749 收藏 1

文章标签： 密码学 信息安全 加密解密

本文链接：https://blog.csdn.net/weixin_43826280/article/details/106843134

buuctf crypto部分刷题的记录，此文是第一部分。后续会继续更新。

0x1 救世捷径

其实是算法题，利用Dijkstra算法即可求出

0x2 [NCTF2019]childRSA

题目

```python
def getPrime(bits):
    while True:
        n = 2
        while n.bit_length() < bits:
            n *= choice(primes)
        if isPrime(n + 1):
            return n + 1

e = 0x10001
m = int.from_bytes(flag.encode(), 'big')
p, q = [getPrime(2048) for _ in range(2)]
n = p * q
c = pow(m, e, n)

# n = 32849718197337581823002243717057659218502519004386996660885100592872201948834155543125924395614928962750579667346279456710633774501407292473006312537723894221717638059058796679686953564471994009285384798450493756900459225040360430847240975678450171551048783818642467506711424027848778367427338647282428667393241157151675410661015044633282064056800913282016363415202171926089293431012379261585078566301060173689328363696699811123592090204578098276704877408688525618732848817623879899628629300385790344366046641825507767709276622692835393219811283244303899850483748651722336996164724553364097066493953127153066970594638491950199605713033004684970381605908909693802373826516622872100822213645899846325022476318425889580091613323747640467299866189070780620292627043349618839126919699862658057999488750773383856176858193302907748803332605606637886917016938981954292889948393670552171042390512873201312153849509695994488907670547192849009247661670983898056223325554232552839895618542119366535989766411083564592864661633770061788394636911070244313598006855351192711572315770458659584492760763600350103887174863941737806234808598087350253509875556881097192692544791385889418017149858013108899222763734185712360760027513776813234715865706369238824951 3
# c = 26308018356739853895382240109968894175166731283702927002165268998773708335216338997058314157717147131083296551313334042509806229853341488461087009955203854253313827608275460592785607739091992591431080342664081962030557042784864074533380701014585315663218783130162376176094773010478159362434331787279303302718098735574605469803801873109982473258207444342330633191849040553550708886593340770753064322410889048135425025715982196600650740987076486540674090923181664281515197679745907830107684777248532278645343716263686014941081417914622724906314960249945105011301731247324601620886782967217339340393853616450077105125391982689986178342417223392217085276465471102737594719932347242482670320801063191869471318313514407997326350065187904154229557706351355052446027159972546737213451422978211055778164578782156428466626894026103053360431281644645515155471301826844754338802352846095293421718249819728205538534652212984831283642472071669494851823123552827380737798609829706225744376667082534026874483482483127491533474306552210039386256062116345785870668331513725792053302188276682550672663353937781055621860101624242216671635824311412793495965628876036344731733142759495348248970313655381407241457118743532311394697763283681852908564387282605279108
```

特点：

p和q是由小素数相乘后加1而得 利用Pollard p-1方法解出p 和 q

```
$ py -m primefac -vs -m=p-1 3284971819733758182300224371705765921850251900438699666088510059287220194883415554312592439561492896275057966734627945671063377450140729247300631253772389422171763805905879667968695356447199400928538479845049375690045922504036043084724097567845017155104878381864246750671142402784877836742733864728242866739324115715167541066101504463328206405680091328201636341520217192608929343101237926158507856630106017368932836369669981111235920902045780982767048774086885256187328488176238798962862930038579034436604664182550776770927662269283539321981128324430389985048374865172233699616472455336409706649395312715306697059463849195019960571303300468497038160590890969380237382651662287210082221364589984632502247631842588958009161332374764046729986618907078062029262704334961883912691969862580579994887507733838561768581933029077488033326056066378869170169389819542928899483936705521710423905128732013121538495096959944889076705471928490092476616709838980562233255542325528398956185421193665359897664110835645928646616337700617883946369110702443135980068553511927115723157704586595844927607636003501038871748639417378062348085980873502535098755568810971926925447913858894180171498580131088992227637341857123607600275137768132347158657063692388249513
# c = 26308018356739853895382240109968894175166731283702927002165268998773708335216338997058314157717147131083296551313334042509806229853341488461087009955203854253313827608275460592785607739091992591431080342664081962030557042784864074533380701014585315663218783130162376176094773010478159362434331787279303302718098735574605469803801873109982473258207444342330633191849040553550708886593340770753064322410889048135425025715982196600650740987076486540674090923181664281515197679745907830107684777248532278645343716263686014941081417914622724906314960249945105011301731247324601620886782967217339340393853616450077105125391982689986178342417223392217085276465471102737594719932347242482670320801063191869471318313514407997326350065187904154229557706351355052446027159972546737213451422978211055778164578782156428466626894026103053360431281644645515155471301826844754338802352846095293421718249819728205538534652212984831283642472071669494851823123552827380737798609829706225744376667082534026874483482483127491533474306552210039386256062116345785870668331513725792053302188276682550672663353937781055621860101624242216671635824311412793495965628876036344731733142759495348248970313655381407241457118743532311394697763283681852908564387282605279108
```

可以得到p和q. 接下来正常求解

0x3 [BJDCTF2020]rsa_output

共模攻击

题目

```
{21058339337354287847534107544613605305015441090508924094198816691219103399526800112802416383088995253908857460266726925615826895303377801614829364034624475195859997943146305588315939130777450485196290766249612340054354622516207681542973756257677388091926549655162490873849955783768663029138647079874278240867932127196686258800146911620730706734103611833179733264096475286491988063990431085380499075005629807702406676707841324660971173253100956362528346684752959937473852630145893796056675793646430793578265418255919376323796044588559726703858429311784705245069845938316802681575653653770883615525735690306674635167111,2767}

{21058339337354287847534107544613605305015441090508924094198816691219103399526800112802416383088995253908857460266726925615826895303377801614829364034624475195859997943146305588315939130777450485196290766249612340054354622516207681542973756257677388091926549655162490873849955783768663029138647079874278240867932127196686258800146911620730706734103611833179733264096475286491988063990431085380499075005629807702406676707841324660971173253100956362528346684752959937473852630145893796056675793646430793578265418255919376323796044588559726703858429311784705245069845938316802681575653653770883615525735690306674635167111,3659}

message1=20152490165522401747723193966902181151098731763998057421967155300933719378216342043730801302534978403741086887969040721959533190058342762057359432663717825826365444996915469039056428416166173920958243044831404924113442512617599426876141184212121677500371236937127571802891321706587610393639446868836987170301813018218408886968263882123084155607494076330256934285171370758586535415136162861138898728910585138378884530819857478609791126971308624318454905992919405355751492789110009313138417265126117273710813843923143381276204802515910527468883224274829962479636527422350190210717694762908096944600267033351813929448599

message2=11298697323140988812057735324285908480504721454145796535014418738959035245600679947297874517818928181509081545027056523790022598233918011261011973196386395689371526774785582326121959186195586069851592467637819366624044133661016373360885158956955263645614345881350494012328275215821306955212788282617812686548883151066866149060363482958708364726982908798340182288702101023393839781427386537230459436512613047311585875068008210818996941460156589314135010438362447522428206884944952639826677247819066812706835773107059567082822312300721049827013660418610265189288840247186598145741724084351633508492707755206886202876227
```

共模攻击求解 s 和 t 的代码

```
gcd ,s , t  = gmpy2.gcdext(e1,e2)
...
m = pow(c1,s,n)*pow(c2,t,n) % n
```

0x4 [BJDCTF2020]RSA

```
flag=open("flag","rb").read()

p=getPrime(1024)
q=getPrime(1024)
assert(e<100000)
n=p*q
m=bytes_to_long(flag)
c=pow(m,e,n)
print c,n
print pow(294,e,n)

p=getPrime(1024)
n=p*q
m=bytes_to_long("BJD"*32)
c=pow(m,e,n)
print c,n

'''
output:
12641635617803746150332232646354596292707861480200207537199141183624438303757120570096741248020236666965755798009656547738616399025300123043766255518596149348930444599820675230046423373053051631932557230849083426859490183732303751744004874183062594856870318614289991675980063548316499486908923209627563871554875612702079100567018698992935818206109087568166097392314105717555482926141030505639571708876213167112187962584484065321545727594135175369233925922507794999607323536976824183162923385005669930403448853465141405846835919842908469787547341752365471892495204307644586161393228776042015534147913888338316244169120  13508774104460209743306714034546704137247627344981133461801953479736017021401725818808462898375994767375627749494839671944543822403059978073813122441407612530658168942987820256786583006947001711749230193542370570950705530167921702835627122401475251039000775017381633900222474727396823708695063136246115652622259769634591309421761269548260984426148824641285010730983215377509255011298737827621611158032976420011662547854515610597955628898073569684158225678333474543920326532893446849808112837476684390030976472053905069855522297850688026960701186543428139843783907624317274796926248829543413464754127208843070331063037
38163126882580646951816637038735203547577567716361573075945434391356361597088196733240770990123563771893618419893022630376187651710120867710731100606572801422047796600062096405661605867699987897694331906383664908508537757727321479237154877520459409788707889589846389244014157797454493926824781893793660701310080816975867504226456854776403162843141472792216858099849469580040304331240664352763766746631847366954232616921866536642304357900338848663416764266349589660728215580833190235118850019796090567220704657964705276457941181430568913751986088091646727205677864144275894013501640080874038714450815635806795521501897915337055253515349847745972087732981120468820838754382612258213240421484845495472248708665806140879522380502220299761352201473698345212107386005485130234351775673270102666706276590627762687921545793633079969881275597305755762093017277885911653857120710042499083850825512761663733449968005864541178692530236879041476824861180935816019755436925545867545010945798769874958463055117757749204340365641996828516353682381981757353135649723615434268991452532167380792545865185476851239635538974086327014877536274444811558163962932636234216054850003500015609721544688125105550545671385417391314204097638250043518544252172112806210903061368369054309575159360374022344774547459345216907128193957592938071815865954073287532545947370671838372144806539753829484356064919357285623305209600680570975224639214396805124350862772159272362778768036844634760917612708721787320159318432456050806227784435091161119982613987303255995543165395426658059462110056431392517548717447898084915167661172362984251201688639469652283452307712821398857016487590794996544468826705600332208535201443322267298747117528882985955375246424812616478327182399461709978893464093245135530135430007842223389360212803439850867615121148050034887767584693608776323252233254261047
'''
```

两个n求出公因数解出p 和 q ， 对e爆破 ， 求出私钥d解密rsa

0x5 [GWCTF 2019]BabyRSA

```
flag = 'GWHT{******}'
secret = '******'

assert(len(flag) == 38)

half = len(flag) / 2

flag1 = flag[:half]
flag2 = flag[half:]

secret_num = getPrime(1024) * bytes_to_long(secret)

p = sympy.nextprime(secret_num)
q = sympy.nextprime(p)

N = p * q

e = 0x10001

F1 = bytes_to_long(flag1)
F2 = bytes_to_long(flag2)

c1 = F1 + F2
c2 = pow(F1, 3) + pow(F2, 3)
assert(c2 < N)

m1 = pow(c1, e, N)
m2 = pow(c2, e, N)

output = open('secret', 'w')
output.write('N=' + str(N) + '\n')
output.write('m1=' + str(m1) + '\n')
output.write('m2=' + str(m2) + '\n')
output.close()
```

p 和 q相差不大，可以分解
解出c1 和 c2 ，求解方程组即可。

0x6 坏蛋是雷宾

rabin密码
rabin解题代码

```
from gmpy2 import *
import hashlib
n=523798549
p=10663
q=49123
e=2
c=162853095
inv_p = invert(p, q)
inv_q = invert(q, p)


mp = pow(c, (p + 1) / 4, p)
mq = pow(c, (q + 1) / 4, q)


a = (inv_p * p * mq + inv_q * q * mp) % n
b = n - int(a)
c = (inv_p * p * mq - inv_q * q * mp) % n
d = n - int(c)

for i in (a, b, c, d):
    print(bin(i)[2:])
```

0x7 nctf2019-babyrsa

```
def nextPrime(n):
    n += 2 if n & 1 else 1
    while not isPrime(n):
        n += 2
    return n

p = getPrime(1024)
q = nextPrime(p)
n = p * q
e = 0x10001
d = inverse(e, (p-1) * (q-1))
c = pow(bytes_to_long(flag.encode()), e, n)

# d = 19275778946037899718035455438175509175723911466127462154506916564101519923603308900331427601983476886255849200332374081996442976307058597390881168155862238533018621944733299208108185814179466844504468163200369996564265921022888670062554504758512453217434777820468049494313818291727050400752551716550403647148197148884402646868466938421183872177535169634497538098603540476192567878694002978585681397003965675194698253985751038854876244634244299130177295856208771681716034441114646928413796611120751233993432706102722878652008803981935732608482686334619834350150312270702178527282408473980844146871463973031107092149132
# c = 53827231680738281106961685582942066817579911490227778211275633014134832238745272333007211808392986170767056850411742474158261570965830550693373939878922627642112252270358807544174570567239091355252449579359069026656797771011301113927802375029286562257052624314319530035200939329243759021112800772552051182174367441120640694296786329232598986279971458038927539892556152731403000210406545059014427878106536265243057063166631693417972057529387555900565689867382278034874672741143982571879621407965511362205328096876068673856393677437055275116807199553807463776311564686898441508783814605609907556528994493400453135218042
```

已知e d，需要求出p和q

```
d = 19275778946037899718035455438175509175723911466127462154506916564101519923603308900331427601983476886255849200332374081996442976307058597390881168155862238533018621944733299208108185814179466844504468163200369996564265921022888670062554504758512453217434777820468049494313818291727050400752551716550403647148197148884408264686846693842118387217753516963449753809860354047619256787869400297858568139700396567519469825398575103885487624463424429913017729585620877168171603444111464692841379661112075123399343270610272287865200880398193573260848268633461983435015031227070217852728240847398084414687146397303110709214913
c = 53827231680738281106961685582942066817579911490227778211275633014134832238745272333007211808392986170767056850411742474158261570965830550693373939878922627642112252270358807544174570567239091355252449579359069026656797771011301113927802375029286562257052624314319530035200939329243759021112800772552051182174367441120640694296786329232598986279971458038927539892556152731403000210406545059014427878106536265243057063166631693417972057529387555900565689867382278034874672741143982571879621407965511362205328096876068673856393677437055275116807199553807463776311564686898441508783814605609907556528994493400453135218041
e=0x10001
src=d*e-1
i=2**15
while True:
    if(src%i==0):
        if((src//i)>=2**2046 and (src//i)<=2**2048):
            phi=src//i
            q_1=iroot(phi,2)[0]      #开平方
            q=nextprime(q_1)     #下一个素数
            if(phi%(q-1)==0):     #验证是否成立
                p_1=phi//(q-1)
                p=p_1+1
                if(isprime(p)):
                    print(p,q)
                    #(14319361159175221091877047640238478335174002884176322323610288522183996663707318846280819597454857983336831390408309578690647941634768192373110026035965242644159310775589248594480941918934831195630845645952343745996971306065340329098739865960424826996704517162967437275255864372484624323274233610808112250758391, 14319361159175221091877047640238478335174002884176322323610288522183996663707318846280819597454857983336831390408309578690647941634768192373110026035965242644159310775589248594480941918934831195630845645952343745996971306065340329098739865960424826996704517162967437275255864372484624323274233610808112250764971)
                    break
    i+=1
    if((src//i)<2**2046):
        print 0
        break
n=p*q
m=pow(c,d,n)
```

0x8 [AFCTF2018]Single

替换密码

直接用quipqiup网站解密即可

0x9 [De1CTF2019]xorz

用到了汉明距离、频率分析

代码(转自官方wp)

```
def getKeyPool(cipher, stepSet, plainSet, keySet):
    ''' 传入的密文串、明文字符集、密钥字符集、密钥长度范围均作为数字列表处理.形如[0x11,0x22,0x33]
        返回一个字典，以可能的密钥长度为键，以对应的每一字节的密钥字符集构成的列表为值，密钥字符集为数字列表。
            形如{
                    1:[[0x11]],
                    3:[
                        [0x11,0x33,0x46],
                        [0x22,0x58],
                        [0x33]
```

```
                    ]
                }
    '''
    keyPool = dict()
    for step in stepSet:
        maybe = [None] * step
        for pos in xrange(step):
            maybe[pos] = []
            for k in keySet:
                flag = 1
                for c in cipher[pos::step]:
                    if c ^ k not in plainSet:
                        flag = 0
                if flag:
                    maybe[pos].append(k)
        for posPool in maybe:
            if len(posPool) == 0:
                maybe = []
                break
        if len(maybe) != 0:
            keyPool[step] = maybe
    return keyPool


def calCorrelation(cpool):
    '''传入字典，形如{'e':2,'p':3}
        返回可能性，0~1,值越大可能性越大
        (correlation between the decrypted column letter frequencies and
        the relative letter frequencies for normal English text)
    '''
    frequencies = {"e": 0.12702, "t": 0.09056, "a": 0.08167, "o": 0.07507, "i": 0.06966,
                   "n": 0.06749, "s": 0.06327, "h": 0.06094, "r": 0.05987, "d": 0.04253,
                   "l": 0.04025, "c": 0.02782, "u": 0.02758, "m": 0.02406, "w": 0.02360,
                   "f": 0.02228, "g": 0.02015, "y": 0.01974, "p": 0.01929, "b": 0.01492,
                   "v": 0.00978, "k": 0.00772, "j": 0.00153, "x": 0.00150, "q": 0.00095,
                   "z": 0.00074}
    relative = 0.0
    total = 0
    fpool = 'etaoinshrdlcumwfgypbvkjxqz'
    total = sum(cpool.values())  # 总和应包括字母和其他可见字符
    for i in cpool.keys():
        if i in fpool:
            relative += frequencies[i] * cpool[i] / total
    return relative


def analyseFrequency(cfreq):
    key = []
    for posFreq in cfreq:
        mostRelative = 0
        for keyChr in posFreq.keys():
            r = calCorrelation(posFreq[keyChr])
            if r > mostRelative:
                mostRelative = r
                keychar = keyChr
        key.append(keychar)

    return key
```

```
def getFrequency(cipher, keyPoolList):
    ''' 传入的密文作为数字列表处理
        传入密钥的字符集应为列表，依次包含各字节字符集。
            形如[[0x11,0x12],[0x22]]
        返回字频列表，依次为各字节字符集中每一字符作为密钥组成部分时对应的明文字频
            形如[{
                    0x11:{'a':2,'b':3},
                    0x12:{'e':6}
                },
                {
                    0x22:{'g':1}
                }]
    '''
    freqList = []
    keyLen = len(keyPoolList)
    for i in xrange(keyLen):
        posFreq = dict()
        for k in keyPoolList[i]:
            posFreq[k] = dict()
            for c in cipher[i::keyLen]:
                p = chr(k ^ c)
                posFreq[k][p] = posFreq[k][p] + 1 if p in posFreq[k] else 1
        freqList.append(posFreq)
    return freqList


def vigenereDecrypt(cipher, key):
    plain = ''
    cur = 0
    ll = len(key)
    for c in cipher:
        plain += chr(c ^ key[cur])
        cur = (cur + 1) % ll
    return plain


def main():
    ps = []
    ks = []
    ss = []
    ps.extend(xrange(32, 127))
    ks.extend(xrange(0xff + 1))
    ss.extend(xrange(38))
    cipher = getCipher(c)

    keyPool = getKeyPool(cipher=cipher, stepSet=ss, plainSet=ps, keySet=ks)
    for i in keyPool:
        freq = getFrequency(cipher, keyPool[i])
        key = analyseFrequency(freq)
        plain = vigenereDecrypt(cipher, key)
        print plain,"\n"
        print ''.join(map(chr,key))


if __name__ == '__main__':
    main()
```

来自天枢战队的代码，直接对flag逐位爆破，对应位置密文异或结果若有意义则保留，否则排除。

```
import string
length = 30
flag_dic = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ_'
plain_dic = '0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ,. '
cipher = '1e5d4c055104471c6f234f5501555b5a014e5d001c2a54470555064c443e235b4c0e590356542a130a4242335a47551a590a13
6f1d5d4d440b0956773613180b5f184015210e4f541c075a47064e5f001e2a4f711844430c473e2413011a100556153d1e4f450614411519
01470a196f035b0c4443185b322e130806431d5a072a46385901555c5b550a541c1a2600564d5f054c453e32444c0a434d43182a0b1c540a
55415a550a5e1b0f613a5c1f10021e56773a5a0206100852063c4a18581a1d15411d17111b052113460850104c472239564c0755015a1327
1e0a55553b5a47551a54010e2a06130b5506005a393013180c100f52072a4a1b5e1b165d50064e411d0521111f235f114c47362447094f10
035c066f19025402191915110b4206182a544702100109133e394505175509671b6f0b01484e06505b061b50034a2911521e44431b5a233f
13180b55081315230501544037404155034 84f0c2602564d470a18407b775d031110004a54290319544e06505b060b424f092e1a77044310
1952333213030d554d551b2006064206555d50141c454f0c3d1b5e4d43061e453e39544c1758085658180200110210544310 1d111a043c03
521455074c473f3213000a5b085d113c194f5e08555415180f5f433e270d131d420c1957773f560d11440d40543c060e470b55545b114e47
0e193c155f4d47110947343f13180c100f565a000403484e184c15050250081f2a54470545104c5536251325435302461a3b4a02484e1254
5c1b4265070b3b5440055543185b36231301025b084054220f4f42071b1554020f430b196f19564d4002055d79'.decode('hex')
list_flag = []
for i in range(0,length):
   list_i = []
   for flag_byte in flag_dic:
       count = 0
       for j in range(i,600,length):
           if chr(ord(flag_byte)^ord(cipher[j])) in plain_dic:
               count += 1
       if count>=600/length-1:
           list_i.append(flag_byte)
   list_flag.append(list_i)
print list_flag
```

0x10 afctf2018-magicnum

浮点数转ascii字符

```
72065910510177138000000000000000.000000
71863209670811371000000.000000
184896826254127600000000000000000.000000
72723257588050687000000.000000
46746591674697662000000000.000000
19061698837499292000000000000000000000.000000
```

c代码（来自博客）

```
#include <stdio.h>

int main()
{   float i = 72065910510177138000000000000000.000000;
    char *p = &i;
    printf("%c\n",*(p));
    printf("%c\n",*(p+1));
    printf("%c\n",*(p+2));
    printf("%c\n",*(p+3));
    return 0;
}
```