MTCTF_Crypto

原创

sh1kaku_ 于 2021-05-25 21:05:32 发布 195 收藏

分类专栏： Crypto 文章标签： 密码学

本文链接：https://blog.csdn.net/shikaku_/article/details/117263592

MTCTF

一共有四道，做出了前两道，bob师傅ak了，记录一下

easy_RSA

第一步给了这些信息：

```
n:0x9371c61a2b760109781f229d43c6f05b58de65aa2a674ff92334cb5219132448d72c1293c145eb6f35e58791669f2d8d3b6ce506f4b3
543beb947cf119f463a00bd33a33c4d566c4fd3f4c73c697fa5f3bf65976284b9cc96ec817241385d480003cdda9649fa0995b013e66f583
c9a9710f7e18396fbf461cb31720f94a0f79L
e:0x3
encrypt(m):0x5f4e03f28702208b215f39f1c8598b77074bfa238dfb9ce424af7cc8a61f7ea48ffbbd5a5e1a10f686c3f240e85d011f6c8
b968d1d607b2e1d5a78ad6947b7d3ec8f33ad32489befab601fe745164e4ff4aed7630da89af7f902f6a1bf7266c9c95b29f2c69c33b93a7
09f282d43b10c61b1a1fe76f5fee970780d7512389fd1L
encrypt(m+1):0x5f4e03f28702208b215f39f1c8598b77074bfa238dfb9ce424af7cc8a61f7ea48ffc5c26b0c12bcff9f697f274f59f0e5
5a147768332fc1f1bac5bbc8f9bb508104f232bdd20091d26adc52e36feda4a156eae7dce4650f83fabc828fdcfb01d25efb98db8b94811c
a855a6aa77caff991e7b986db844ff7a140218449aaa7e8L
```

显然是个Franklin-Reiter攻击，解密算法是coppersmith，于是用sagemath解：

```
def short_pad_attack(c1, c2, e, n):

    PRxy.<x,y> = PolynomialRing(Zmod(n))

    PRx.<xn> = PolynomialRing(Zmod(n))

    PRZZ.<xz,yz> = PolynomialRing(Zmod(n))


    g1 = x^e - c1

    g2 = (x+y)^e - c2
```

```
    q1 = g1.change_ring(PRZZ)

    q2 = g2.change_ring(PRZZ)


    h = q2.resultant(q1)

    h = h.univariate_polynomial()

    h = h.change_ring(PRx).subs(y=xn)

    h = h.monic()


    kbits = n.nbits()//(2*e*e)

    diff = h.small_roots(X=2^kbits, beta=0.5)[0]  # find root < 2^kbits with factor >= n^0.5


    return diff
def related_message_attack(c1, c2, diff, e, n):

    PRx.<x> = PolynomialRing(Zmod(n))

    g1 = x^e - c1

    g2 = (x+diff)^e - c2


    def gcd(g1, g2):

        while g2:

            g1, g2 = g2, g1 % g2

        return g1.monic()

    return -gcd(g1, g2)[0]

n=10353899917070788041551949508274639962458843278997742140636898094680491883866131581849099072023843019985373583
5314279220543849187855536479273346661124943692747785561204014667595291985197472099558155942787887737269419606791
1774764190648612615374968667478818574083660849255312841423626267190965472050615068957 37
e=3
c1=6692526981165098182122795985958678449311032433186136466514873516858015588505530670006373853080299107294526303
5966847007672831546078947147437704784378852465264721011186196139469374189536612608862442812168338813841666743718
0859603204528185243391788937640830398219152776631383169782517449699922312004985354075 69
c2=6692526981165098182122795985958678449311032433186136466514873516858015588505530671475848761459173712878944067
8469431513168060638418391926079830877473690259031930508456232414891405790482123922059095778216110701924539262214
1770705238595600372306312738982734162230965405890430228731189485695596989414108526448 40

nbits = n.nbits()
kbits = nbits//(2*e*e)
print ("padding lenght %d bytes " % (kbits/8))
print ("upper %d bits (of %d bits) is same" % (nbits-kbits, nbits))
diff = 1

m = related_message_attack(c1, c2, diff, e, n)
print(m)
```

参考链接

需要注意的是，这里已知m和m+1的加密结果，所以diff=1，并不需要用到short_pad_attack函数

然后得到压缩包的密码：**everything_is_easy_in_this_question**

里面的信息如下：`280316470206017f5f163a3460100b111b2c254e103715600f13,`
`091b0f471d05153811122c70340c0111053a394e0b39500f0a18, 4638080a1e49243e55531a3e23161d411a362e4044111f374409,`
`0e0d15470206017f59122935601405421d3a244e10371560140f, 031a08080e1a540d62327f242517101d4e2b2807177f13280511,`
`0a090f001e491d2c111d3024601405431a36231b083e022c1d, 16000406080c543854077f24280144451c2a254e093a0333051a,`
`02050701120a01334553393f32441d5e1b716027107f19334417, 131f15470800192f5d167f352e0716481e2b29010a7139600c12,`
`1609411e141c543c501d7f232f0812544e2b2807177f00320b1f, 0a090c470a1c1d3c5a1f2670210a0011093a344e103715600712,`
`141e04040f49153142043a22601711520d3a331d0826`

标题就告诉你是one time cipher，脚本很乱，就不放了

Random

连接端口后得到两个信息：

```
e*d+n=35633297540489769466037294664262360520001411667008399033232552682031857090204944501733698062146668509430761881757786675089462704927083974479505217323240591483902327440110000659828659741949867267396380975663031355730721144486150952620665547518589520423953754171515936766218259390697837678651386577685537677170349700e*d-n=35631217189172345887237864632755558268752323806911659190337189249584063538108134801847442190467178380784970904037510072545451877201076029593818817158758982434745049997602081331925728121109671424746193666505049486196379096537233769171744560913962205768412597987920787691983690729820637162066905895546049924707877520
```

也就是给了e*d和n，RSA已知edn分解pq，数学原理大致就是利用一个很小的数对幂进行分解一直到不能再分解为止，爆破出p，q

```
from gmpy2 import next_prime, gcd


def Factorize(n, ed):
    g = 2
    while True:
        k = ed-1
        while not k & 1:
            k //= 2
            p = int(gcd(pow(g, k, n) - 1, n)) % n
            if p > 1:
                print(g)
                return (p, n // p)
        g = int(next_prime(g))


if __name__ == "__main__":
    n = 104017565871178939971501575340112562454393004836992144768171622389677604840484994312793583974506432289548886013830127200541386300397244284320008224080452457863872125395966395146581042009792132509365457899093476717102397445859172446049330231365732777057809179757453711728433043860025829224034106974387623123609

    ed=35632257364831057676637579648508959394376867736960029111784870965807960314156539651790570126306923445107866392897648373817457291064080002036662017240999786959323688718856040995877193931529769346071287321084040420963550120511692360896205054216275863096183176079718362229100975060259237420359146236615793800939113610

    print(Factorize(n, ed))
```

得到：

```
p=109804055081742712599253331663435795537190613169419451903239390836654899022861688612296645893652100263882981734824967572646979964047946850646746682724797 71
q=947301680195179777126784644545973847397342158805814069525303151170040753393587239726473163190117466515927887865803509423122806387848014555608820664104277 9
```

之后爆破e，d，在十万以内有三组解：

```
e=[45687, 65553, 75247]
```

然后挨个试，在e=65553时成功了，下一阶段就是LCG，已知10个连续生成值得到seed，脚本如下：

```
from gmpy2 import gcd,invert
from functools import reduce
from Crypto.Util.number import *
lcg=[373207461671623820087376019958358658538005041346424780658116499432866936280568583158930409651925975131678849650551 2, 889020410002643234774595552531028821910539847878753728765026701587339597931898875369329439855209813852612984936474 8 , 344307231541519820980708360837797317710170991115581498688336855116257288936928879875547609259319636164476825729631 8 , 450527808990863331989796465516481052624098240650279022924700809960037666147571037658720380909689911378702988757735 5 , 905964627329109917595537196941355559193431828915680231496713219575269254926353240795269786795905404552747026966107 3 , 308502406338164832678867729416859167542330228602627144184885636903258204951291546508242872918734151073800822687090 0 , 829602898428855915492844262234161637629320583471650776650077048226197342404411106116336982895181513548685386292916 6 , 225875025995436317142641556114557913551112733614262630602186897206443474209239264495364761121070078774999646676702 6 , 438212313003494454265515657500071085107884229536735394319951287851463943477016160232611591591353141705854795493 6492 , 109829335982234278520054727485433799136018963986478116809645791613391289089765111733828965491042960314832439009439 25 ]

def crack_unknown_modulus(states):
    diffs = [s1 - s0 for s0, s1 in zip(states, states[1:])]
    zeroes = [t2*t0 - t1*t1 for t0, t1, t2 in zip(diffs, diffs[1:], diffs[2:])]
    modulus = abs(reduce(gcd, zeroes))
    return crack_unknown_multiplier(states, modulus)
def crack_unknown_multiplier(states, modulus):
    multiplier = (states[2] - states[1]) * invert(states[1] - states[0], modulus) % modulus
    return crack_unknown_increment(states, modulus, multiplier)
def crack_unknown_increment(states, modulus, multiplier):
    increment = (states[1] - states[0]*multiplier) % modulus
    return modulus, multiplier, increment
modulus, multiplier, increment=crack_unknown_modulus(lcg)
remultiplier=invert(multiplier,modulus)
seed=(lcg[0]-increment)*remultiplier% modulus
print(long_to_bytes(seed))
```

然后就得到了flag

后记

还是太菜了呀，前两道题很多细节都把握不住，在bob师傅耐心的指导下做出来了。
之前太懈怠了了，要开始多做题了。