

Crypto

第一题

RSA，但是n特别大。通过看代码发现p和q都符合 2^n-1 这一形式。我们知道指数的增长是非常快的，那么可以通过 `libnum.gcd()` 函数得出n的一个因子。如图。

```
task.py
task.py > No Selection
21760037126354322632135047624976370716030032762004102300007072300710327179001202210000037004737007307102200074007113310230010742013480
8863281563685419483112294522066621545654209819630729109439974566550367158285111558035205190151293969910755432001132582172707112666592558867
1493882019145648956271615654316440321905380221231276576190033940370313372783531071424333120440758646896149373089526503999343561782359884923
3877636140606312163910986575786388157410869614071872525731928122559067993512108785682221702332733281574520542008914729209077239374901359663
637027537059976976910914303558245352109535926999855355079787997346383426083213011038589359094814741908976449748350484578664207026761891970
2391567971318631325667087916415365077503294375455877566480225663882990763437943998316415848326509883587228489963734787854881959683851324638
8220853419026850512993051755008601013045733990586682993129275747017410024439062767485427837573126538958985464467115474913616346907624022015
6880259297040239010487640861968021534152049267804780642988330520556274512536479460210687058150668175789327707131915895384517507807866017960
8196467657260838076736379838608564972663185371190006636806148457208669645139104654396131314831047747941920305523064097118253264943243151448
02337453958784590818095393764480271054649775782188744901307027663871874871746470590469320230424450955567169537
28 e =
29 c =
65537
2747230465686934364492940871515324868392868975353130223784614958619322655965852472626685704693541599170802117023089169653475239630815894826
7883448005519437899038324466800260210112385442078460903737669349655257050264668557967019304568164476003371712074047631859356381314220275797
3702239047641509560473188385514368034765804147555539511736976425888570287326879454013818994639479102268406402310797771276301050540373899230
9757209162010361585600400859235665258055453040868030445784823963084812467846920422018102448364940681809087475322264064084500019436931058077
1229814383522301051424863596590780874214227418269499366538896115646215263529622610945094596111978549423398090993404775464170357473041015202
1685991701533369283007476252159633517630105125287688431216490129139805115766972930538219806867707287139630153217683188003090776192007793015
1057094846403428175672447021775760982351719578572401855257562249133948084612906093680161051555407672984218440844966768906294766977545875699
5253120785957525187827300384774112337418322486945805846155099612540019035400289672095933344790066690507378154508782025366734942533152175403
2976433513550163674681305591065426732567741788842596758878948187820038338135003782249432441455642626306229931029785429788605715169816881100
2608704427494270398810285740113422081879305121465497284624729384486924042269774783148988004427702322075980549456826529670334936684481553746
29677160486873168967742551866186111255422134879287890339768485586086380464815353593938860099819880811839831965922365779297765474981110180878
1936527757357239555683829958047450274021835014495062065908776012644688434564200851049885463697273124612569331498876656220381633200293350324
3437889863070276727879267986089985706383225156589922164389585151874961615612334049404427210575996338071547332195506960774004491011707124920
6896299622697682570796009979231639024949641531925642671892612175509176257755874660881922500484568695515777221221301172884965160819874524331
1729526588001196426268910052305558512663028217941976930796771184504406989715748422158412810397418240755388517215486585714147306511749680100
0582559794780041356513213292042115672524100778125761067035645619886377867254247369267789140221428188031625458572695412941813385401518593038
7538559001852054899430659831877897869634304089804042890940508613367691810105478309187935398788782894721886082746696672680602366985718301192
199044918403911149757344481946344762772558236351226158716656971440746603633587421806448241396824296258018282131665158321569398245274315943
3370039743897072216855095688844123698745297458491116059452828366633503448889148896657676191573646532434154
```

得出其中一个因子后我们就用同样的方法得到另一个因子。然后就是算逆元，得到d。再 `pow(c,d,n)` 就是明文，用一下 `libnum.n2s` 得 flag。

第三题

通过“有几个朋友发送的内容还是相同的！”和一堆的n，e，c(并且每个e都是3)得出考点是中国剩余定理，公式代入并不复杂，但是这有几个是那几个呢？通过如图程序(注释内容中的)得到对应的那几个，分别是1.3.5和2.4.6，有一组数据是没用的。然后就是常规的操作了。

task-6.py

No Selection

```
55 N=[n1,n2,n3,n4,n5,n6,n7]
56 C=[c1,c2,c3,c4,c5,c6,c7]
57 '''m=n1*n2*n3*n4*n5
58 m1=n2*n3*n4*n5
59 m2=n1*n3*n4*n5
60 m3=n2*n1*n4*n5
61 m4=n2*n3*n1*n5
62 m5=n2*n3*n4*n1
63
64 t1=gmpy2.invert(m1,n1)
65 t2=gmpy2.invert(m2,n2)
66 t3=gmpy2.invert(m3,n3)
67 t4=gmpy2.invert(m4,n4)
68 t5=gmpy2.invert(m5,n5)
69
70 x=(c1*t1*m1+c2*t2*m2+c3*t3*m3+c4*t4*m4+c5*t5*m5)%m'''
71
72 for i in range(7):
73     for j in range(7):
74         if(j<=i):
75             continue
76         for k in range(7):
77             if(k<=j):
78                 continue
79             m=N[i]*N[j]*N[k]
80             m1=N[j]*N[k]
81             m2=N[i]*N[k]
82             m3=N[i]*N[j]
83             t1=gmpy2.invert(m1,N[i])
84             t2=gmpy2.invert(m2,N[j])
85             t3=gmpy2.invert(m3,N[k])
86             x=(C[i]*t1*m1+C[j]*t2*m2+C[k]*t3*m3)%m
87             if(gmpy2.iroot(x,3)[1]==True):
88                 print(gmpy2.iroot(x,3))'''
89 x=568732055145476975591470226574180134836236313767966458457771393585064887002171250821082394523653678193601034640294327744277986338401795144492
1287998468535322627581987600520791246790801774613389776111756063858341810137533343711764384080712638213804098446979696348450514241120554085
3143792576015685532078872844216797138414524536907439515992368589530044695860898023979837106986464535204065351399843115034554597382151865676
68150718389315356984306066841639282959775474727315485170232447343000188235096262607720177650796746319777284125771252063723080510750668669
89 y=185621001424160505736839429992438186350760920739904026017598343467161341032417110105862754116391090086768820731921486434550316104819370411759
0783831649790976119044134604441471303785279732373156855887068354665450765996037183830774646291030696384414382038890615830751742900755717699
7981100832381005241166245313840924340700889103328791356006596260398018870616432028969867769367476898927333093436690154570749992900186513222
256367699832103567881359455935790857756039619588610627989749300374686342861485344010971873179013269852993512813
90 print(libnum.n2s(y))
91
```