buuctf crypto page 2

原创

~VAS~ 已于 2022-03-02 10:22:37 修改 97 收藏

分类专栏： 笔记 buuctf ctf 文章标签： 密码学

于 2022-02-23 14:01:40 首次发布

本文链接：https://blog.csdn.net/zip471642048/article/details/123087621

笔记 同时被 3 个专栏收录

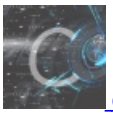
53 篇文章 0 订阅

订阅专栏

buuctf

5 篇文章 0 订阅

订阅专栏



ctf

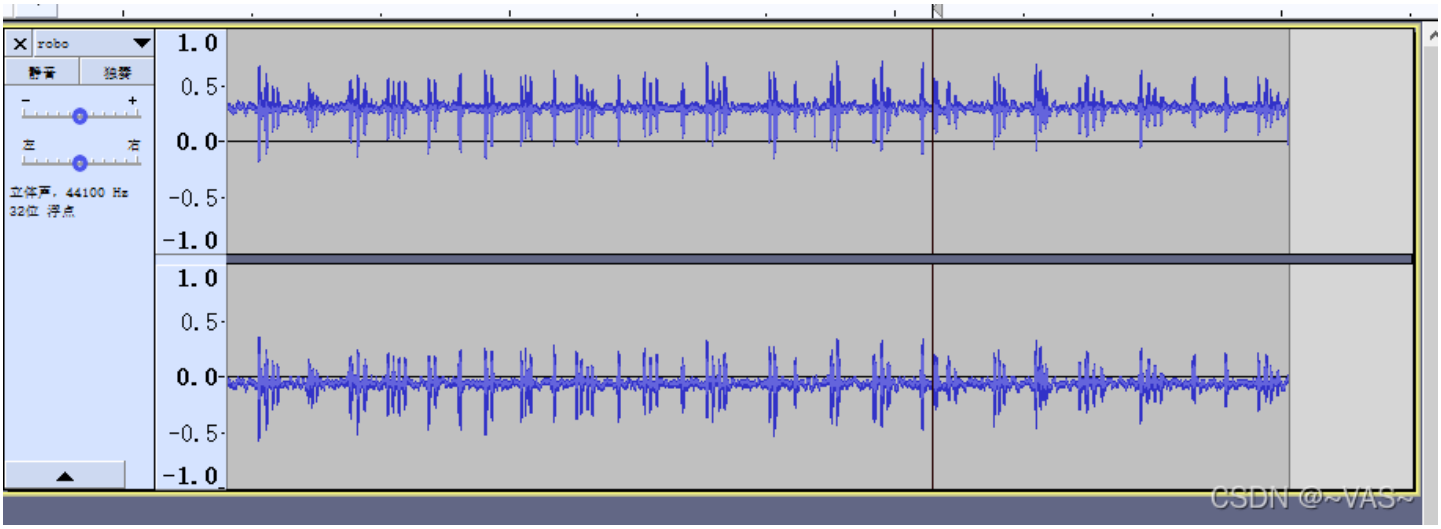
50 篇文章 1 订阅

订阅专栏

目录

robomunication

听着就是莫斯密码，读的有点搞笑hah



hellowhatisthekeyitisboopbeep
flag为大写的boopbeep

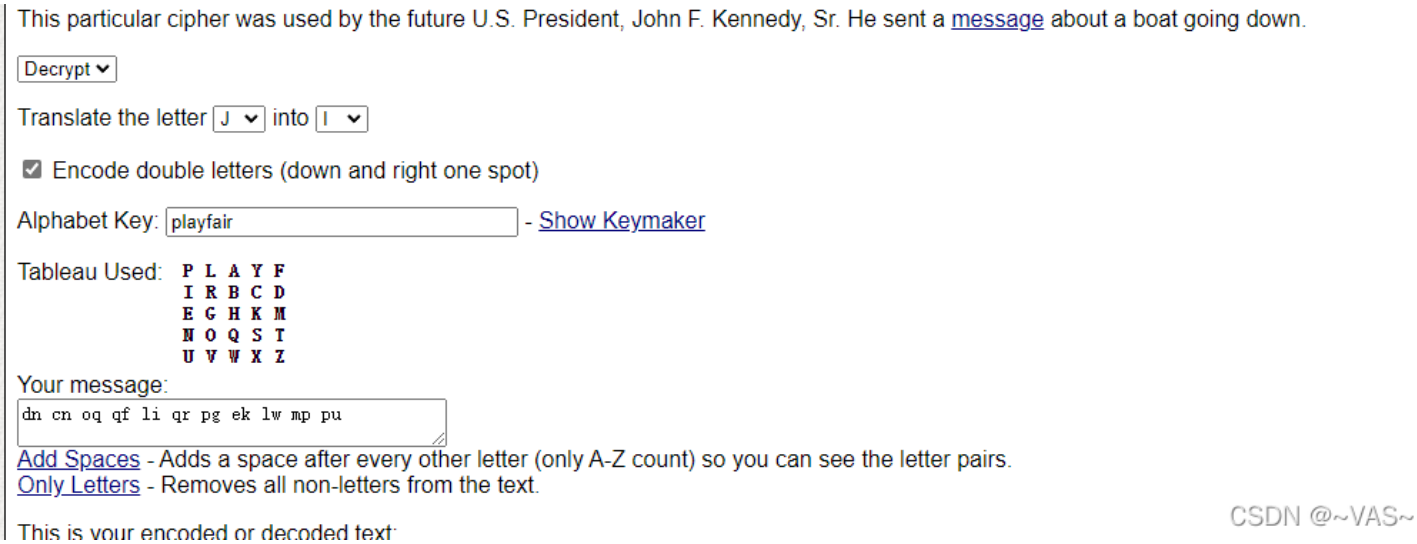
Dangerous RSA

低加密指数攻击

```
n=0x52d483c27cd806550fbe0e37a61af2e7cf5e0efb723dfc81174c918a27627779b21fa3c851e9e94188eaee3d5cd6f752406a43fbecb53e80836ff1e185d3ccd7782ea846c2e91a7b0808986666e0bdadbfb7bdd65670a589a4d2478e9adcafe97c6ee23614bcb2ecc23580f4d2e3cc1ecfec25c50da4bc754dde6c8bfd8d1fc16956c74d8e9196046a01dc9f3024e11461c294f29d7421140732fedacac97b8fe50999117d27943c953f18c4ff4f8c258d839764078d4b6ef6e8591e0ff5563b31a39e6374d0d41c8c46921c25e5904a817ef8e39e5c9b71225a83269693e0b7e3218fc5e5a1e8412ba16e588b3d6ac536dce39fcdfce81eec79979ea6872793
e=0x3
c=0x10652cdfaa6b63f6d7bd1109da08181e500e5643f5b240a9024bfa84d5f2cac9310562978347bb232d63e7289283871efab83d84ff5a7b64a94a79d34cfbd4ef121723ba1f663e514f83f6f01492b4e13e1bb4296d96ea5a353d3bf2edd2f449c03c4a3e995237985a596908adc741f32365
import gmpy2
import libnum
k = 0
while 1:
    res = gmpy2.iroot(c+k*n,e)

    if res[1] == True:
        print(libnum.n2s(int(res[0])))
        print(k)
    k += 1
```

Cipher

还能提示什么呢？公平的玩吧（密钥自己找） Dncnoqqfliqrpgeklwmppu 注意：得到的 flag 请包上 flag{} 提交, flag{小写字母}
密钥是playfair
解密网站http://rumkin.com/tools/cipher/playfair.php

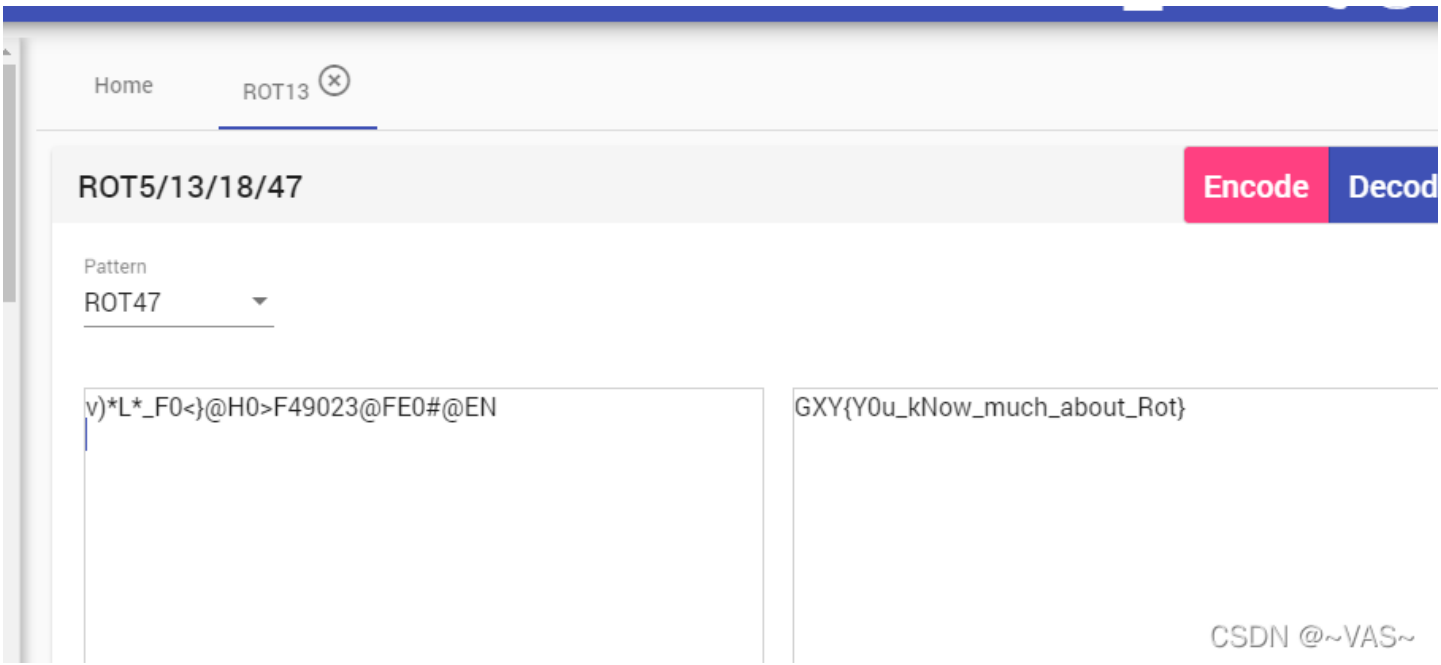


[HDCTF2019]basic rsa

```
import gmpy2
from Crypto.Util.number import *
from binascii import a2b_hex,b2a_hex

flag = "******************"

p = 26224880018227704065019205543990658047 9
q = 26285499423932282854792559548751991555 1

e = 65533
n = p*q
phi_n = (p-1)*(q-1)
c = 27565231154623519221597938803435789010285480123476977081867877272451638645710
d = gmpy2.invert(e,phi_n)
m = pow(c,d,n)
print(long_to_bytes(m))
# 27565231154623519221597938803435789010285480123476977081867877272451638645710
```

[GXYCTF2019]CheckIn

base64之后rot47



[GUET-CTF2019]BabyRSA

```
pq1= 0x1232fecb92adead91613e7d9ae5e36fe6bb765317d6ed38ad890b4073539a6231a6620584cea5730b5af83a3e80cf30141282c97b
e4400e33307573af6b25e2ea
pq2 = 0x5248becef1d925d45705a7302700d6a0ffe5877fddf9451a9c1181c4d82365806085fd86fbaab08b6fc66a967b2566d743c62654
7203b34ea3fdb1bc06dd3bb765fd8b919e3bd2cb15bc175c9498f9d9a0e216c2dde64d81255fa4c05a1ee619fc1fc505285a239e7bc655ec
6605d9693078b800ee80931a7a0c84f33c851740
e = 0xe6b1bee47bd63f615c7d0a43c529d219
d = 0x2dde7fbaed477f6d62838d55b0d0964868cf6efb2c282a5f13e6008ce7317a24cb57aec49ef0d738919f47cdcd9677cd52ac2293ec
5938aa198f962678b5cd0da344453f521a69b2ac03647cdd8339f4e38cec452d54e60698833d67f9315c02ddaa4c79ebaa902c605d7bda32
ce970541b2d9a17d62b52df813b2fb0c5ab1a5
enc_flag = 0x50ae00623211ba6089ddfae21e204ab616f6c9d294e913550af3d66e85d0c0693ed53ed55c46d8cca1d7c2ad44839030df2
6b70f22a8567171a759b76fe5f07b3c5a6ec89117ed0a36c0950956b9cde880c575737f779143f921d745ac3bb0e379c05d9a3cc6bf0bea8
aa91e4d5e752c7eb46b2e023edbc07d24a7c460a34a9a
import gmpy2
from Crypto.Util.number import *
n = int(pq2) - int(pq1) -1
m = pow(int(enc_flag),int(d),int(n))
print(long_to_bytes(m))
```

达芬奇密码

按照斐波那契数列和数列1的位置来排序数列2

```
a = '1 233 3 2584 1346269 144 5 196418 21 1597 610 377 10946 89 514229 987 8 55 6765 2178309 121393 317811 46368
 4181 1 832040 2 28657 75025 34 13 17711'.split(' ')
# def fib_recur(n):
#     assert n >= 0, "n > 0"
#     if n <= 1:
#         return n
#     return fib_recur(n-1) + fib_recur(n-2)
#
# for i in range(1, 40):
#     print(fib_recur(i), end=' ')
flag = [''for i in range(32)]
print(flag)
b = '1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196
418 317811 514229 832040 1346269 2178309'.split(' ')
c = '369688538821167255473421769522 86'
for indexa,i in enumerate(a):
    for indexb,j in enumerate(b):
        if i == j:
            flag[indexb] = c[indexa]
            print(indexa,indexb,flag)
flag[0] = '3'
for i in flag:
    print(i,end='')
```

密码学的心声

8进制的ASCII码

```
s = '111 114 157 166 145 123 145 143 165 162 151 164 171 126 145 162 171 115 165 143 150'
tmp = [s.split(' ')[i] for i in range(len(s.split(' ')))]
for i in tmp:
    print(chr(int(i,8)),end='')
```

rsa2

```
#python2
import hashlib
import RSAwienerHacker
N = 10199180977755325347027675139926474013115768232925267350179215450700615843443200914199536724196252570595004625340018888465826249653470643879151507188586089755273665689956691573129722581725063987364337631010399217064690655724283289391490205358108750251278730332274778042021088485216658671763655905815254497947 1
e = 46731919563265721307105180410302518676676135509737992912625092976849075262192092549323082367518264378630543338219025744820916471913696072050291990620486581719410354385121760761374229374847695148230596005409978383369740305816082770283909611956355972181848077519920922059268376958811713365106925235218265173085
d =  RSAwienerHacker.hack_RSA(e,N)
print(d)
flag = "flag{" + hashlib.md5(hex(d)).hexdigest() + "}"
print(flag)
```

[BJDCTF2020]这是base??

base64换表

```
import base64
import string
dict={0: 'J', 1: 'K', 2: 'L', 3: 'M', 4: 'N', 5: 'O', 6: 'x', 7: 'y', 8: 'U', 9: 'V', 10: 'z', 11: 'A', 12: 'B',
 13: 'C', 14: 'D', 15: 'E', 16: 'F', 17: 'G', 18: 'H', 19: '7', 20: '8', 21: '9', 22: 'P', 23: 'Q', 24: 'I', 25:
 'a', 26: 'b', 27: 'c', 28: 'd', 29: 'e', 30: 'f', 31: 'g', 32: 'h', 33: 'i', 34: 'j', 35: 'k', 36: 'l', 37: 'm'
, 38: 'W', 39: 'X', 40: 'Y', 41: 'Z', 42: '0', 43: '1', 44: '2', 45: '3', 46: '4', 47: '5', 48: '6', 49: 'R', 50
: 'S', 51: 'T', 52: 'n', 53: 'o', 54: 'p', 55: 'q', 56: 'r', 57: 's', 58: 't', 59: 'u', 60: 'v', 61: 'w', 62: '+
', 63: '/', 64: '='}
str1 = "FlZNfnF6Qol6e9w17WwQQoGYBQCgIkGTa9w3IQKw"

string1 = "JKLMNOxyUVzABCDEFGH789PQIabcdefghijklmWXYZ0123456RSTnopqrstuvw+/"
string2 = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/"

print (base64.b64decode(str1.translate(str.maketrans(string1,string2))))
for l,v in dict.items():
    print(v,end='')
```

RSA5

```
from gmpy2 import *

n1 = 20474918894051778533305262345601880928088284471121823754049725354072477155873778848055073843345820697886641
08684261248654125018396596600159134203156295356179333234164133430284799610841746636068813986650517968951658930563690213721018562465085490678003720441220630994919908000557692277577372243886376211775042932758579209344742398000240120061330294383421282090926971387668346581736915858582229467505697897061220288542643607195021453826292107740907616041743669983613880116262131484560879687020683470411670776316984738722330782890857094498441697301942752979002908976626494907803866952346524383767526385806285473908363420 7
c1 = 97446390824333086572897876921359540078205339859689774131627572259641501891292950863739385091922496927176638871002519503989696195606289557006214694773634034292797499261667889337274426195417287349087880548324119634588172116407865115606711995781642276852444202568807946265675560598210417400163534587402213304540234401004596111172015199041203447775585180276906930906901873854185413018369220475876142712127998200299393974534369567190001529679063746488033737551153642479689099652668120063308684103632039584772593574475799301335280465057506813612929559130656921330015633365091079594680082006749414336488584289629112613732 0
c2 = 15819636201971185538694880505120469332582151856714070824521803121848292387556864177196229718923770810072104155432038682511434979353089791861087415144087855679134383396897817458726543883093567600325204596156649305930352575274039425470836355002691145864435755333821133969266951545158052745938252574301327696822347115053614052423028835532509220641378760800693351542633860702225772638930501021571415907348128269681224178300248272689705308911282208685459668200507057183420662959113956077584781737983254788703048275698921427029884282557468334399677849962342196140864403989162117738206246183665814938783122909930082802031855

n3 = 25033254625906757272369609119214202033162128625171246436639570615263949157363273213121556825878737923265290579551873824374870957467163989542063489416636713654642486717219231225074115269684119428086352535471683359486248203644461465935500517901513233739152882943010177276545128308412934555830087776128355125932914846459470221102007666912211992310538890654396487111705385730502843589727289829692152177134753098649781412247065660637826282055169991824099110916576856188876975621376606634258927784025787142263367152947108720757222446686415627479703666031871635656314282727051189190889008763055811680040315277078928068816491
c3 = 41853085294168740058312307810140924071984513859556773996685018339026234783956692794048839907251843327091524433725837010761987866352917393567708572867021071567300200043589556225110614106610589826220551997368208082038414467963052843946517144309186903894869205608346723161581464531837894121409390290293247560353580817544266451600332629243302486752161082709801570497054886202634851294809528147640028652800191851276624493183242793832777664162581422751439235321687984130110282715430852490290489974522125031117423023020654010514580665853953604684474606586729528 5
```

```
164354719382277521838785362345363802549238912220450755908862

n4 = 21206968097314131007183427944486801953583151151443627943113736996776787181111063957960698092696800555044199
15676567793537314959822118479228681221329461774983460769630211613674566281665811705542780331523004270069512571840
01646810484873064775005221089174056824724922160855810527236751389605017579545235876864998419873065217294820244730
07851205251265658155602290018876228375491181680816851833710923951285981250047302689102760248068085658020813668980
90403250992045378599705615049764523492552888387941964218910649009132381586673390027614766605038951015853086721168
68018787523459264932165046816881682774229243688581614306480751
c4 = 45210380110447584418911284684672330884938857508505889857085199111547780905971361261502890418934541266744681
41393472662337350361712212694867311622970440707727941113263832357173141775855227973742571088974593476302084111770
06257642228383662775595608870429488598921385514726806545178149166092797483655806107122598566777405184770865315920
23310717547006829190360750579943293198966370747701790461142621377023839700574373038608003195569415846655847559970
51940245039167629126576784024482348452868313417471542956778285567779435940267140679906686531862467627238401003450
91016371912972094224703881218025365697614144576182583435506130
n5 = 22822039733049388110936778173014765663663303811791283234361230649775805923902173438553927805407463106104699
77399415837570403309347176138779985216833789852698052175361430789966901593138781992742187531630459152190159282380
14417756447695701045846773508629371397013053684553042185725059996791532391626429712416994990889693732805181947970
07142930959961497377273655629940424642479166067925388494002172884690634419885477919195173971934290876133066191040
47711993342855077424291042095249692960568615479948783992342433635374744215357167806452076314979329436078782175170
03543288696726923909670396821551053048035619499706391118145067
c5 = 15406498580761780108625891878008526815145372096234083936681442225155097299264808624358826686906535594853622
68737926896946843307238814978660739539642410431882087944374311235870654675393521575607834595937529965071855575969
98887852318017597503074317356745122514481807843745626429797861463012940172797612589031686718185390345389295851075
52792785161470766022701785406901478083141727989874972593300378103285234648518956218518590278236816559341047136890
53984804716308866689647366550015817904619653821077889773020957270843006765841175595986603353170046055155638099398
27061718489704602243049964556005039822234489048782128494123 57

n6 = 21574139855341432908474064784318462018475296809327285532337706940126942575349507668289214078026102682252713
75770308155309310882321406379151848228984678019732982113950797476378026029030960088492081195984292554058396708568
70848765317877441480914852329276375776405689784571404635852204097622600656222714808541872252335877037561388406258
71817152787666528247863762622492749604671939619566909748536797952491587510784222965803675062197197387621599659588
87780618746107068907129094818194956125414431077694333485977512165018624584603172050794498783848972312789722341680
24360212786712372279936867919447114223450004797511877044263 69
c6 = 20366856150710305124583065375297661819795242238376485264951185336996083744604593418983336285185491197426018
59503144465212328846149187902109602820369413668320344169298706956351302600186143572211798555990969267090734756359
94578265880806540396777223906955491026286843168637367593400342814725694366078337030937104035993569672959361347288
78941430271868468567729830583289197167029822221428488481177684999966175883053014830854285472673370709987674125402
25911508196842253134355901263861121500650240296746702967594224401650220168780537141654489215019142122284308116284
84129004257364769474080721001708734051264841350424152506027932

n7 = 25360227412666612490102161131174584819240931803196448481224305250583841439581008528535930814167338381983764
99129657563723191654764797057375826941116821930237054168478912511250502114850680964308195023762370318102569658599
80446956913220121836604246364968970730455574007687459437873425482673865646254621431501761136562644502100239255719
45961405709276631990731602198104287528528055650050486159837612279600415259486306154947514005408907590083747758
95311548612486548672063382055913506344094252803140295195855763083350377511201071560427811432552899377108123353524
71184817658522732524049634307928989482195394733124629798491 37
c7 = 19892772524651452341027595619482734356243435671592398172680379981502759695784087900669089919987705675899945
65864862380009027259915459012308218964502180095807686151839732543952113999565202637713236823250210862003340005134
61277576986238861426217934232257492402865116665560917878516839780175069833100735243982872797376800917873335475382
39920606776108098824363954757081836378867324958278301547568210998471529316313732443986283857446010879371417260367
24777668313564113044468819986747795011881636006644880329436396948286989847394922006996844627489228835500026529135
18229322945040819064133350314536378694523704793396169065179

n8 = 22726855244632356029159691753451822163331519237547639938779517751496498713174588935566576167329576494790219
36072787716607413649612992729629699697004808287048880445656498666712938813655613701334622811898193689951068758958
52865171513230482931502570368474754240443781091681794122878893405963947552577049380061626776565815093754711025462
61355748251869048003600520034656264521931808651038524134185732929570384705918563982065684145766427962502261522
48199419198982011057598190699843155310752554200118765570353468323177798841926833824954764133571839331229580004473
45347616927994034694979540628978562990312574547359458674911 91
```

```
c8 = 604011979517585640754108236002353220461472385868863672482271271757275979396024634180030814973980987123431304962973293479756978105300068618566637483397840329052507259877400173135024474459077279570106512956189811657649998418592066127112366535613271919366547423559688423910803060588277786885612237822268114057051918032128697694715404227262241130398101130258622563085989273172464057465812547828711519840625384736797988376800081260539548295269868960447771947894759544218592148065263786833567323320066210062102506150089572960530566586469312295255736187152316530020607032566035309559277803776739536032923133132282361006000 6

n9 = 232973337914430532973630007868353360952522908184619500545426583274845074065946327857127674599589179430955225942282054234282073451288997458009273191472576697738126695427828392377443051800982765788419294963459639975122442193767017876160462353971393818948374355626625910607684769973335387480652940331416105022523252928018168122689341713619343999515486272677914010897039373890125865810802233130601594562388570807406995286664113030299348070112149539841697858447141596277920169264909552826978771416146388063976893067953283447784786920847542167534258425578188994679451026467763426551676553842248605040860831478412522327609416794510264677634265516765538422486050408608314784125223276094 1

c9 = 541812030120837871311588946557996425787181411451504609609096015973785907682925851692036157785390392595419840684375730368755784830230220022929591690243020573784360180670073823475669857570861242492848044086873912007588868167206220652915656642127661110780291741899362502969062719681383032636987424977761923960330060587686596751571907979711591057865356278789901931013994590495802488241783373630489476543348947623457535675527514725657738702287334890690014963494074710451385015411810699113707264330862028466310828305224575094522899538780343212884215225154929269894740766364389585343265002935209201837283445705427110281693 4

c10 = 991988046378683668498795797909152747747144499639237524407552784186550916018166654301631763496351243751032419870241632284137748941702957238847445007580146299682524465753028610742818635417283671650281760907059092976926193232427535328993930253644031062869834924487206400570064452022372767095078792429600429688303297894120088336265399335163854586020717902247249267125663042722846185266811803531702142867595487494701519774591691819772512112223636938274153398302346225591392469280624938744901662986582331640236601765784416691984668349785184238805828385621990053556742710360386995506619342550138525532209790153140210388 3869

n11 = 223246859475396537224999324694096075330654191573478139619580756890476904652664043841994836839085947873124455281596355278339044758018903814556538072655012173287578713527312930003034382053158167926639175790666748423077438452617710323639285688446698957680925156583287562292458370252617442606148607469979315035487885099838680383497202253057309855762936752690737090223507008365100540676417537132129999543070225244958855833617073785137421625663390101343549078637332059218450389182244639037898418814008140745872617202838797601220709014665171182654228634203769215367348455021002514608724991222366868321895496980207373176683019 c11 = 14915270502032949898828292485603951848049772777471261431039572191646241875284410478373512635804406864747673804640055402646279101264831299306683440958145475921150610578434701314980750604203951110086190271990370199257012366601665630682456839757877628043595201647016916909164825910261385827055582468694961627597808784371379608230000439882273030038764105031213701633037116033594307645393375978668625084515281582851032518100587418796878752183841602825061727066133594776572154207348160493933395937554892185887966070602618979052334532686714116106310473404594879374795119334503694622137957389330190014718031576077917385384 67

n12 = 27646746423759020111007828653264027999257847645666129907789026054594393648800236117046769112762641778865620892443423100189619327585811384883515424918752749559627553637785037359639801125213256163008431942593727931931898199727552768626775618479833029101249692573716030706695702510982283555740851047022672485743432464647772882314215176114732257497240284164016914018689044557218920300262234652840632406067273375269301008409860193180822366735877288205783314326102263756503786736122321348320031950012144905869556204017430593656052867939493633163499580242224763404338807022510136217187779084917996171602737036564991036724299 c12 = 219915241289572605360437712848549203931058081267001282221258567755068857219711931093613159611291908146746471364648870878939906608949616128382050864010188854576674889118986542702355619801111746033237212809111974882865852693568495792630434563163194764958886962193442198665168611876541805092478812512512789193462671299047392773862892403943845751243311356559435138310099340233974570821846997377343888237633068053264303958499357702138175333872354863070088924109206116699326930181655694174458858108257496093886272312358409126446546858196209316633462975963348344986617890164503717692036501099947718724041857702301729340139 71
```

```
n13 = 2054548740581692873173898837447501268682793370978978439185570683513627027093340120301932913693765087838611
7187776530639342572123237188053978622697282521473917978282830432161153221216194169879669541998840691383025487220
8508720754360643084999249585179797279544029656121960814043416515173263640415192501250364248226343542687738954656
9892088343922299658122635859587399397660469983061393232072055413001167129794443351504718056548449519100388759989
1289037982010216357831078328159028953222056918189365840711588671093333013117454034313622855082795813122338562446
2230412111922770892250783246821080338430239035501728919596735 51
c13 = 1422743918819102946125047669279053965461919988848731942911441455797537630868890802814081715720557980405978
3807641305577385724758530138514972962209062230576107406142402603484375626077345190883094097636019771377866339531
5119651366505674123638891831596161884492637524753286632453110599883379960473592632888374363055888480445729377594
2446658687028051242433680706472989451584055240475687959069879704633333644546512044508758762174390662427962177963
4772378802959109714400516183718323267273824736540168545946444437586299214110424738159957388350785999348535171553
5693730882515527123912883652952676656913577196160116136287721 75
n14 = 2735972771158427723489715772405585279401921684522979893865581426946004638435356813859856775539255965346094
9444557879120040796798142218939251844762461270251672399546774067275348291003962551964648742053215424620256999345
4483988052785927770496682815583128717739799313430978068787011140560300415066904769542540065925552753425795296252
3119432135790466851212153951488070404696997489841209567508258531545826759101673492464629435766692429390841834550
8902112711075232047998775303603175363964055048589769318562104883659754974955561725694779754279606726358588624791
9881599927683923495214201721059388737195064541841735591256798 7
c14 = 3788529784248255027081674540877016372807848222776887920453488878247137930578296797437647922494510483767651
1504929333560932889659437415702689438619870242766107127174091399464095139630431144639331460884300042377471634228
0295925029660257064936301615158136400679589422659958470807258269699674051888760678546077585102981428035938576309
1078902301957226484620428513604630585131511167015763190591225884202772840456563643159507805711004113901417503751
1810508236382078035331114295109116161608513917547544347648195680548508238109011598212978497900056461021293540357
3535012447683878666154208904550965691034867674284495700885745 7
n15 = 2754593760375173724878522089173579646897332973807620914407992144996729257234942453901050228756403011683126
1268197384650511043068738911429169730640135947800885987171539267214611907687570587001933829208655100828045651391
6180896032884565703345005331786952384076847022512526715793710186516750543686062825246733699830346823305783087698
8645633581873382723729457047685367355268536168914426155289575826652239300411601784939734625911922106382166328093
5820440671825601452417487330105280889520007917979115568067161590058277418371493228631232457972494285014767469893
6478928886814339658574969161107049447580702686268970450147828 37
c15= 1406911297060889573241703997754273266579660189376240150087878687168064579875478331569351126174005972517134 2
4041865710669725463328136677111356611766594246199361010389034391442948863793225916357666826451798880586175775724
0930748470817114448870841054346297200817999459408747393563802661267938975975681149052412719562874126287130442790
8481214992471182859308828778119005750928935764927967212343526503410515793717201360360437981322576798056276657140
3633327007147322248483468089639923024090377060945889641702395211935894700708397904045972529908185837178691402298
1171229500571054047635674337890664226704572363387401164925984 2
n16 = 2574616207569791156026318179121643306257417857242460033685627817611273305443146325390343312823270905414160
7100891177804285813783247735063753406524678030561284491481221681954564804141454666928657549670266775659862814924
3865841487854536473168649359427729191405635063056662078168976018627130928092344290965847532637078288997809792231
1818100929365556314652679238891346255730643366429696633146990642866512743882939970300286780026994785586926203671
4256550075520193125987011945192273531732276641728008406855871598678936585324782438668746810516660152018244253008
0924700665556872771389372987479519295762310362513162706025134 51
c16 = 1734428486027548947749152581992285532679227512871970940129254560812285982982746208839004461223496755168287
9954301458425842831995513832410355328065562098763660326163262033200347338773439095709944202252494552172589503915
9659315243265236632897775831526647222419208005378673310306239066740818522962323063362715428327284108036311702296
4271752494233239084246703514363150440114072708327073246423744391526386588058030877611121971896174637884292464414
2127243573824972533819479079381023103585862099063382129757560124074676150622288706094110075567706403442920696472
6277976076979628730261122405274983085359032326639390285870367 24
n17 = 2328848693411712031503691941858813622702848549413793019632371533620884932783396569389467056721797172792124
3839129969128783853015760155446770590696037582684845937132790047363216362087277861336964760890214059732779383020
3492048032057258702254299859395701415082200412868578100481646967070186637584168077089106714774073660988834308118
6193301497340939017994857771257974935229944031054368903565146539986790842888554123777614340437633344294939706324
9223702355051571790555151203866821867908531733788784978667478707672984539512431549558672467752712004519300318999
2081020767325014125891049047349837898953587536640774868945294 99
c17 = 1073825441811407654807144884496404646814162174060321438498635418910523697707100142927156063642807597045989
0958274941762528116445171161040040833357876134689749846940052619392750394683504816081193432350669452446113285638
```

```
98255176258665632910900721401994497581643482776888270463046000120945223916289657619187632466233315383553395660029525515837702519842695094404064323543021101106358603246772432973578594737205175904213817105416585484247299058380089998489323254909276640051030008358551301417122042310345229289149614180695630039654068238166836756456942781309206405399310353763599431114301070881485186723970649257720389902 4

n18 = 19591441383958529435598729113936346657001352578357909347657257239777540424811749817783061233235817916560689138344041497732749011519736303038986277394036718790971374656832741054547056417771501234494768509780369075443550907847298246275717420562375114406055733620258777905222169702036494045086017381084272496162770259955811174440490126514747876661317750649488774992348005044389081101686016446219264069971370646319546429782904810063020324704138495608761532563310699753322444871060383693044481932265801505819646998535192083036872551683405766123968487907648980900712118052346174533513978009131757167547595857552370586353973
c18 = 38349170988872029319819687046591193416244322947593619195539375510534996074403332340181891419702463022993857425482785898960332828949812003532706371272134831721825298904959034256491167559016311016658763017998656127177503600890851791427506646034541936420530163847145158558683687235089222717671902855211377856880756228329248292483627744764562328268858010469693845195493854282595915667168908446046962587836393908541530393294807262051471992471836215351724508259790471324954396038408065012549971670511424271573817998907253237655588038080301094680486822520287202413574786147046100891208103671924143520341774846885023640228874 

n19 = 19254242571588430171308191757871261075358521158624745702744057556054652332495961196795369630484782930292003238730267396462491733557715379956969694238267908985251699834707734400775311452868924330866502429576951934279223234676654749272932769107390976321208605516299532560054081301829440688796904635446986081691156842271268059970762004259219036753174909942343204432795076377432107630203621754552804124408792358220071862369443201584155711893388877350138023238624566616551246804054720492816226651467017802504094070614892556444425915920269485861799532473383304622064493223627552558344088839860178294589481899206318863310603
c19 = 67905535339912972058045619912254931053123988251876822507801975107847652264296632842204004805630393419385997833467240510762112656634686438264301090132450140358111782950819399586870874773128677202899645060978197620952444791293599988676718118197381966878846966804634586613743109946107600094742641157502049208755274344864375366235896845194115191001702914233674249385668203154865074442020224080038791184657612739167552908981129915255461141910640229913297243700646325699038561892361778940077666907826302474438953588939837358228242434871818510987872712702567808910944051219476310887299173983176523204977651017901326791718890 

n20 = 26809700251171279102974962949184411136459372267620535198421449833298448092580497485301953796619185339316064387798092220298630428207556482805739803420279056191194360049651767412572609187680508073074653291350998253938793269214230457117194434853888765303403385824786231859450351212449404870776320297419712486574804794325602760347306432927281716160368830187944940128907971027838510079519466846176106565164730963988892400240063089397720414921398936399927948235195085202171264728816184532651138221862240969655185596628285814057082448321749567943946273776184657698104465062749244327092588237927996419620170254423837876806659
c20 = 38621355660843401376986472712387941204199127152899052854850745121069261898665287042463221942460167752426501104314674830977406789498506928806795254613941681940403968845475604486278463088283349609082256858057285902980646671301748901528132153712913301179254879877441322285914544974519727307311002330350534857867516466612474769753577858660075830592891403551867246057397839688329172530177187042229028685862036140779065771061933528137423019407311473581832405899089709251747002788032002094495379614686544672969073249309703482556386024622814731015767810042969813752548617464974915714425595351940266077021672409858645427346

n=[n1,n2,n3,n4,n5,n6,n7,n8,n9,n10,n11,n12,n13,n14,n15,n16,n17,n18,n19,n20]
c=[c1,c2,c3,c4,c5,c6,c7,c8,c9,c10,c11,c12,c13,c14,c15,c16,c17,c18,c19,c20]
for i in range(len(n)):
 for j in range(len(n)):
  if(i!=j):
   if(gcd(n[i],n[j])!=1):
    print(i+1,j+1)
    print(gcd(n[i],n[j]))
p = mpz(132585806383798600305426957307612567604223562626764190211333136246643723811046149337852966828729052476725552361132437370521548707664977123165279305052971868012755509160408641100548744046621516877981864180076497524093201404558036301820216274968638825245150755772559259575544101918590311068466601618472464832499)
e = 65537
q = (n5//p)
phi = (q-1)*(p-1)
d = invert(e,phi)
```

```
m = pow(c5,d,n5)
from Crypto.Util.number import *
print(long_to_bytes(m))
```

[NCTF2019]childRSA

p和q用yafu分解

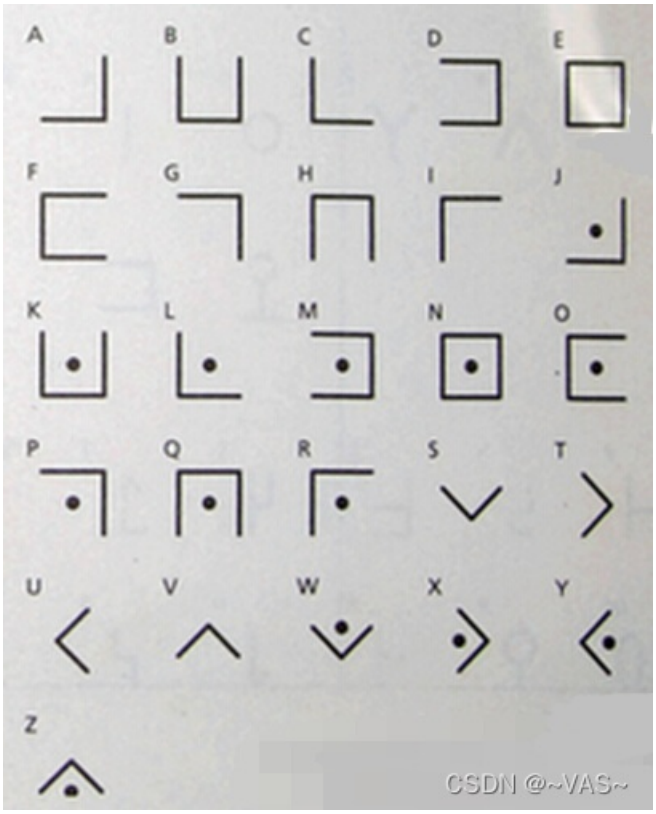
```
n = 32849718197337581823002243717057659218502519004386996660885100592872201948834155543125924395614928962750579667346279456710633774501407292473006312537723894221717638059058796679686953564471994009285384798450493756900459225040360430847240975678450171551048783818642467506711424027848778367427338647282428667393241157151675410661015044633282064056800913282016363415202171926089293431012379261585078566301060173689328363696699811123592090204578098276704877408688525618732848817623879899628629300385790344366046641825507767709276622692835393219811283244303899850483748651722336996164724553364097066493953127153066970594638491950199605713033004684970381605908909693802373826516622872100822213645899846325022476318425889580091613323747640467299866189070780620292627043349618839126919699862580579994887507733838561768581933029077488033326056066378869170169389819542928899483936705521710423905128732013121538495096959944889076705471928490092476616709838980562233255542325528398956185421193665359897664110835645928646616337700617883946369110702443135980068553511927115723157704586595844927607636003501038871748639417378062348085980873502535098755568810971926925447913858894180171498580131088992227637341857123607600275137768132347158657063692388249513c = 26308018356739853895382240109968894175166731283702927002165268998773708335216338997058314157717147131083296551313334042509806229853341488461087009955203854253313827608275460592785607739091992591431080342664081962030557042784864074533380701014585315663218783130162376176094773010478159362434331787279303302718098735574605469803801873109982473258207444342330633191849040553550708886593340770753064322410889048135425025715982196600650740987076486540674090923181664281515197679745907830107684777248532278645343716263686014941081417914622724906314960249945105011301731247324601620886782967217339340393853616450077105125391982689986178342417223392217085276465471102737594719932347242482670320801063191869471318313514407997326350065187904154229557706351355052446027159972546737213451422978211055778164578782156428466626894026103053360431281644645515155471301826844754338802352846095293421718249819728205538534652212984831283642472071669494851823123552827380737798609829706225744376667082534026874483482483127491533474306552210039386256062116345785870668331513725792053302188276682550672663353937781055621860101624242216671635824311412793495965628876036344731733142759495348248970313655381407241457118743532311394697763283681852908564387282605279108import gmpy2e = 65537p = 17844949321269420574233207858325620505867229060365261624022734063873081194522494782612177264220462933510887383278192139030850176366115463869693573270972401654695597752908813599583849747635074962144271969072222691363577241088051663965136362682144245677900969933345261695319379932864744696870704530470254791579973443181880037436037729230924836154886890906689547451833308944658176342575538983707216697068487701166323497863186970385954187604913271349009072040835110838797157743895172733796236847805929544604796251068769504749448060547337717302146776449554154159039473268514082915276153203579018726972470344438683865619367425313 9q = 18408412154011530759716136701101414289882352602767435455503778587848171160225730750898502257780178278876978680001598441044371779999464223619484068455753891784942096736012150967534829620388634026438522415096464295896543880186430618750379010028109913086397771020466054679912875541852132729071963507522158582421748738622700467352729228153622195896176068103229334009939586319403178843514229608521959486663519246435336503408959241480933218388242346153612397287387147775594908222383004959456132945734953770392632515294958212341904907301314432568963205543328335499926519311728825291851530876701688567880221736670037665436550286 7phi = (p-1)*(q-1)d = gmpy2.invert(e,phi)m = pow(c,d,n)print(m)from Crypto.Util.number import *print(long_to_bytes(m))
```

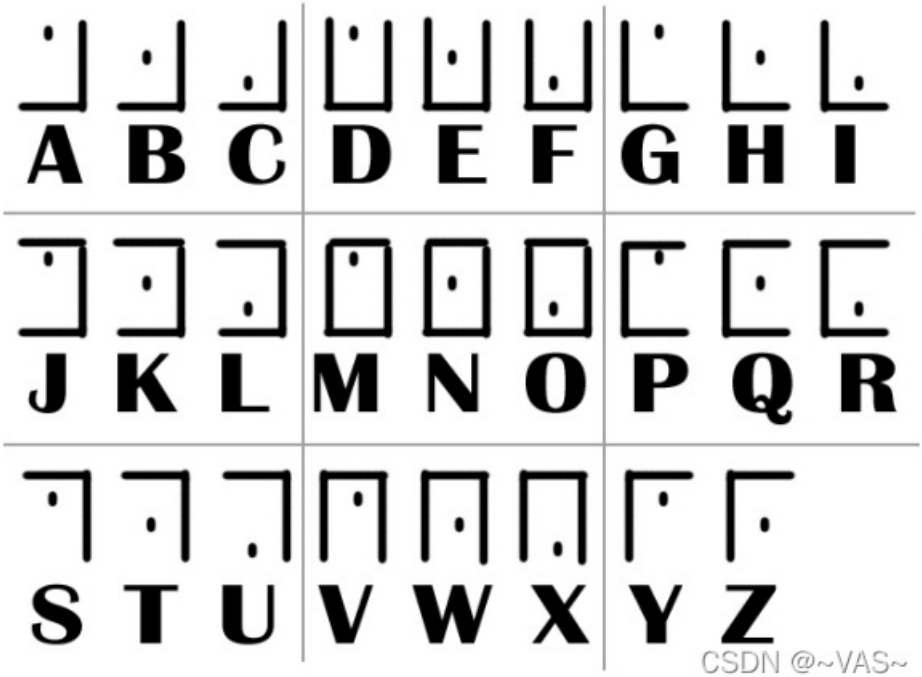
[NCTF2019]Keyboard

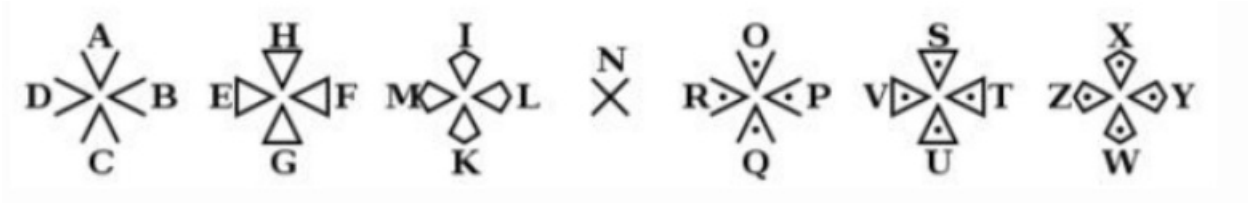
```
cipher = 'OOO YYY II W UUU EE UUUU YYY UUUU Y W UUU I I RR W I I RR RRR UUUU RRR UUUU T II UUUU I W U RRR EE WWW
 EE YYY EEE WWW W TT EE'
base = 'QWERTYUIOP'
a=[', 。? ! ','ABC','DEF','GHI','JKL','MNO','PQRS','TUV','WXYZ']
for part in cipher.split(" "):
    s=base.index(part[0])
    count=len(part)
    print(a[s][count-1],end="")
```

[MRCTF2020]古典密码知多少

标准银河字母+圣堂武士密码+猪圈密码组合







FGCPFLIRTUASYON

然后栅栏

```
Bytes解码:      FGCPFLIRTUASYON
Escape解码:     FGCPFLIRTUASYON
------------------------
栅栏fence解码:
因数[3, 5]:
分为3栏时，解密结果为: FLAGISCRYPTOFUN
分为5栏时，解密结果为: FPIUYGFRAOCLTSN

------------------------
凯撒Caesar解码:
key #0: FGCPFLIRTUASYON
key #1: EFBOEKHQSTZRXNM
key #2: DEANDJGPRSYQWML
key #3: CDZMCIFOQRXPVLK
```

[HDCTF2019]bbbbbbrsa

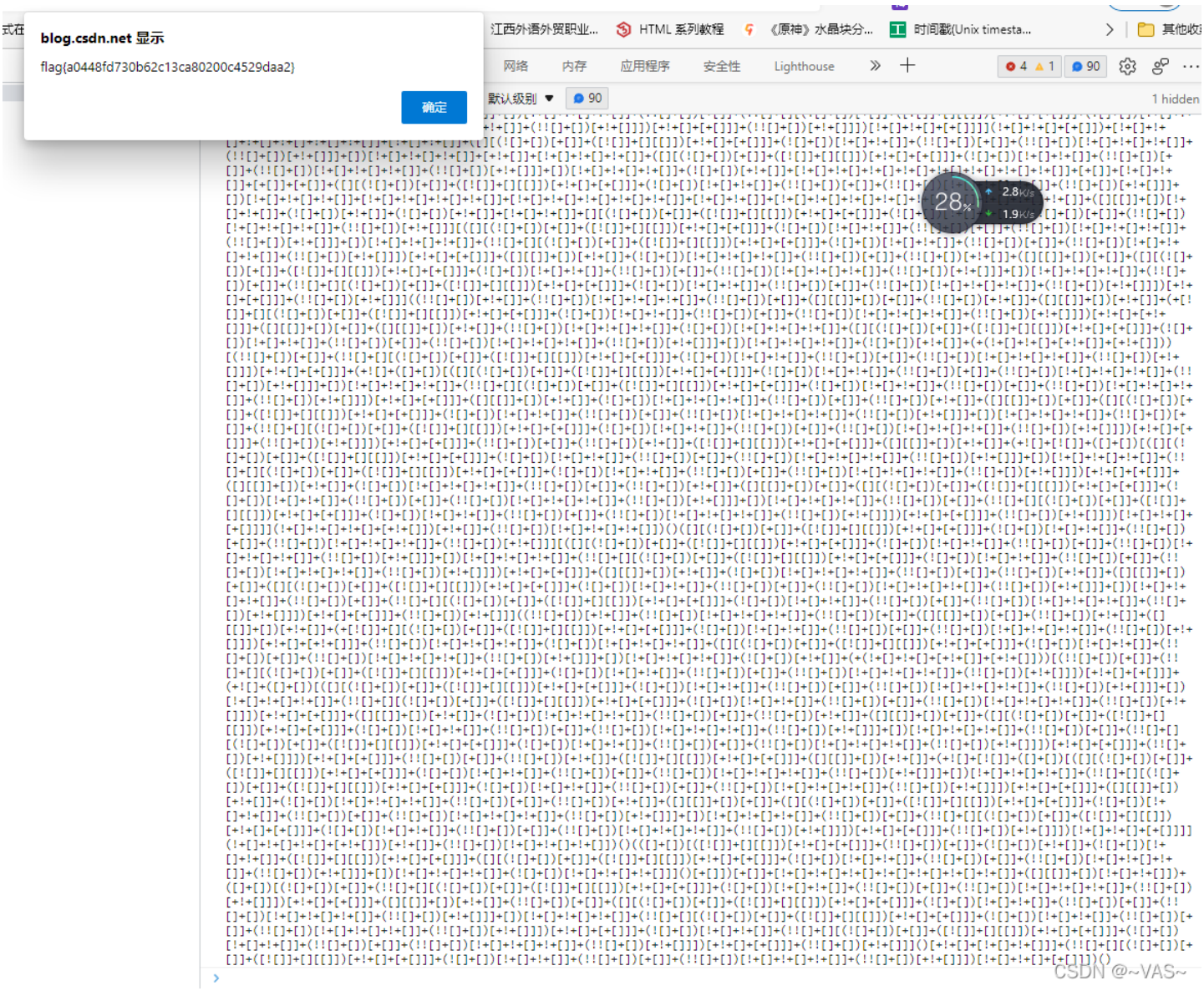
爆破e

```
p = 17707738967525769504250799816500646084 9
n = 37421829509887796274897162249367329400988647145613325367337968063341372726061
c = 23737406995293649917635893242000934662067855618361018403816222372255122346 32
q = n//p
import gmpy2
from Crypto.Util.number import *
phi_n = (p-1)*(q-1)
for e in range(50000,70000):
    try:
        d = gmpy2.invert(e,phi_n)
        m = pow(c,d,n)
        print(long_to_bytes(m))

    except Exception:
        pass
```

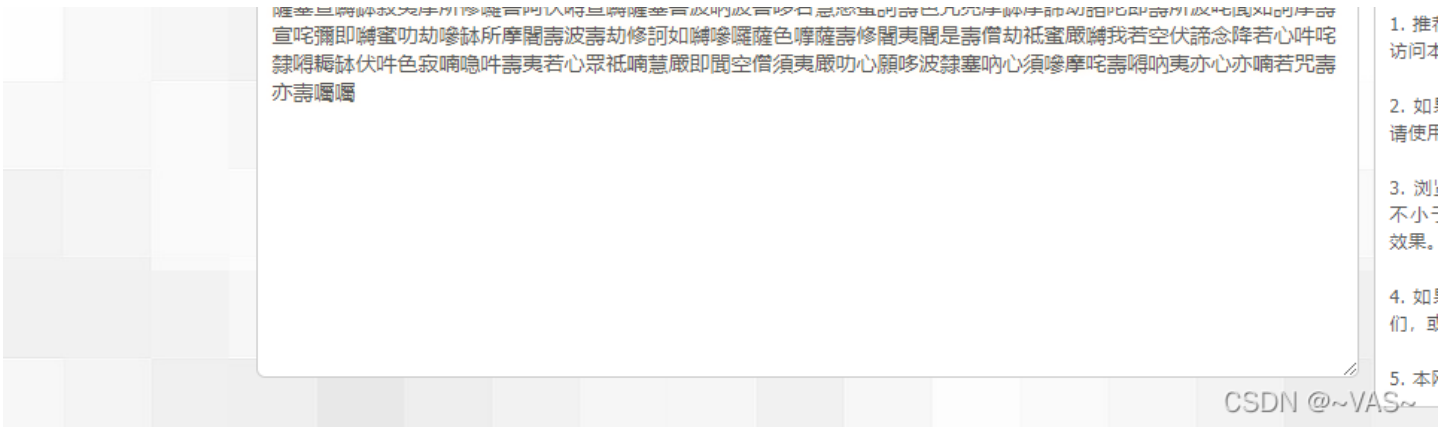
```
b'\x1f\xceB\x8d\x90\xd5\xd6\x83A\xc6;Z\xcc\x89\x07x\xea\xdb1\xdfo\xe2b\x89\xe8\xb1\xfb!\x0e\x90A\xec'
b'\x1a\xd6Q\xbdC\xe8DA\x8e\x95-Gp\xfdOJ-q\xd3\xd0_5\xbc\xa9f9\x838\xf36\x94\xe1'
b'\n!\xcea\xba\x15\xca;2\xa6\xac\xcf\xef\xc8>\xc1g\xd9\x15\xe9w"\xc6\x1d\xc2\xbd\rD6R\xcdH'
b'\x14a_c+C\xe3\xb9\x87\xb7\xd1S\xd4\n\xe3P\xbag-\x1cSM\x87\x04<\xa4\\\xb1\x00"\x80\xd4'
b'flag{rs4_1s_s1mpl3!#}'
b',\x8e\x98\xa9\x1c.9t\x07+g7\xc6(\xee)\x1f\n]H\xc5G\x17I/N\xc6\x9b\x11"\x0b\xd4'
b'\x12\x1bDv*\xef\xc2\x96\x97\xa0"6[j\x18\x81\xba7\xc5=\xd0\x93\x9d\x02]?\xc2\xba\x87HK\xf1'
b'\x07{\xde\x8c\xf8\x98`*\x90u\xa8\xc0}P\xcc\x80\xbd\xd2\x0fn\x8e\x9d\xb1f\x8dp\xaa\x82\t\x1d7\xc8'
b'2}\x92\xfbj\xdf \x0b\x88/\xcbJ\xa1\x9f6\xca^\x86\xd79u\xb6z\xb3\xe8,u\xd8\xa0+\xc8['
b'M\xfe\xea\x04\xe1o\xc5X0\xd4\x9b\x85$\xb4\xe1\x1c\xea\xbeY\n\xed\x12\xd4y\xdb\xc7\xf6\xd0\xf8.t\xeb'
b'A\x84\x11\x16.Za\x8dQ\xbe\x1bW?\x88\x17\xf6;\xe5\x00X\x8d\xb2|\xc4\x10A\xe8N\xdf\xa0\x946'
b'A\xbeYR\x8cV!a\x975"(\x8d\x8225\xc6\xbc\xb2R\xf3\xc7y\x1bW\xad\xfa\x1bHT\xdb\x93'
```


这是什么

解压压缩包在WordDocument中找到一段jsfuck



[WUSTCTF2020]佛说：只能四天



核心价值观编码

社会主义核心价值观：富强、民主、文明、和谐；自由、平等、公正、法治；爱国、敬业、诚信、友善

RLJDQTOVPTQ606duws5CD6IB5B52CC57okCaUUC3SO4OSOWG3LynarAVGRZSJRAEYEZ_ooe_doyouknowfence

编 码　　解 码

平等文明自由友善公正自由诚信富强自由自由平等民主平等自由自由友善敬业平等公正平等富强平等自由平等民主和谐公正自由诚信平等和谐公正公正自由法治平等法治法治法治和谐和谐平等自由和谐自由自由和谐公正自由敬业自由文明和谐平等自由文明和谐平等和谐文明自由和谐自由和谐和谐平等和谐法治公正诚信平等公正诚信民主自由和谐公正民主平等平等平等平等自由和谐和谐和谐平等和谐自由诚信平等和谐自由自由友善敬业平等和谐自由友善敬业平等法治自由法治和谐和谐自由友善公正法治敬业公正友善爱国公正民主法治文明自由民主平等公正自由法治平等文明平等友善自由平等和谐自由友善自由平等文明自由民主自由平等平等敬业自由平等平等诚信富强平等友善敬业公正诚信平等公正友善敬业公正平等平等诚信平等公正自由公正诚信平等法治敬业公正诚信平等法治平等公正友善平等公正诚信自由公正友善敬业法治法治公正公正公正平等公正诚信自由公正和谐公正平等

工具介绍

核心价值观编码（Core Values Encoder），经爱党爱国青年sym同意移植本站，旨在通过编程学习党的十八大提出的"社会主义核心价值观"。
联系作者：github地址
工具地址：https://sym233.github.io/core-values-encoder/

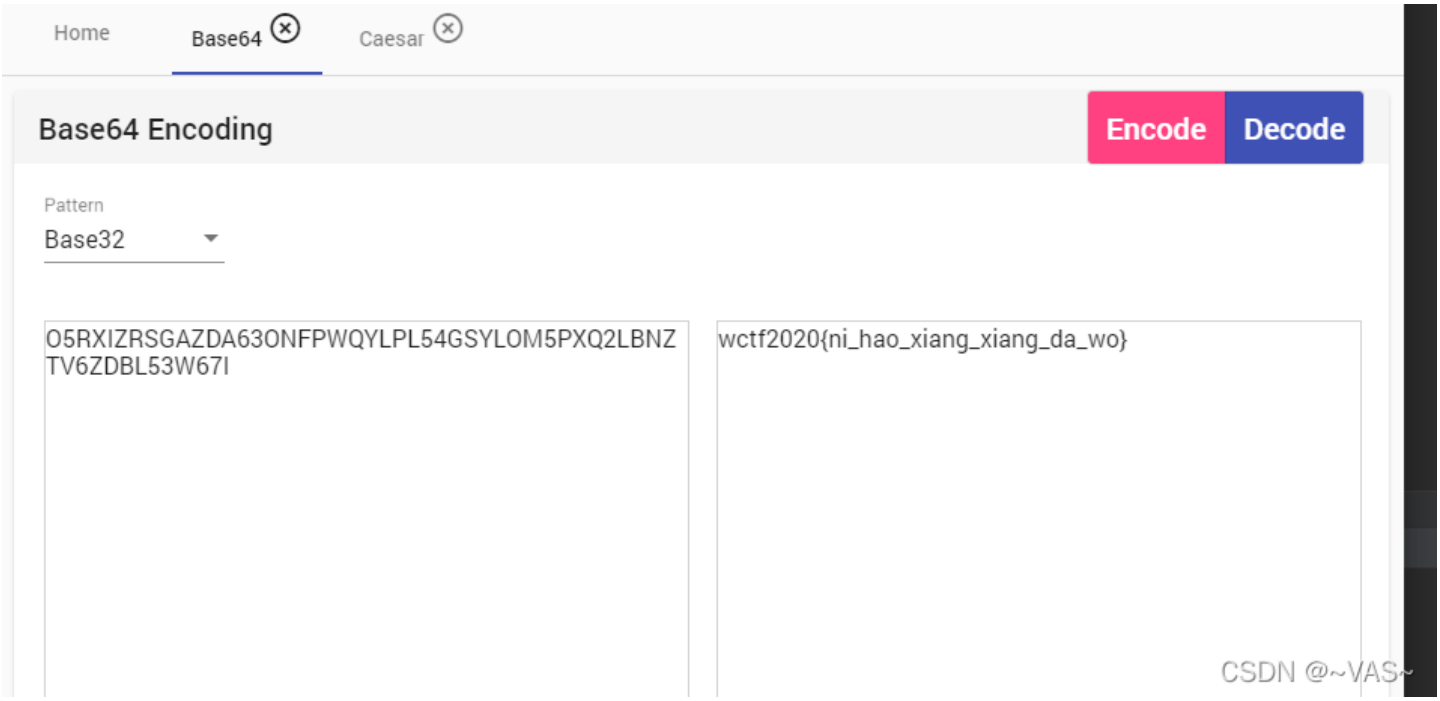
RLJDQTOVPTQ606duws5CD6IB5B52CC57okCaUUC3SO4OSOWG3LynarAVGRZSJRAEYEZ_ooe_

解密结果↓　　复制内容　　↑解密结果转至文本框↑

Quoted解码:　RLJDQTOVPTQ606duws5CD6IB5B52CC57okCaUUC3SO4OSOWG3LynarAVGRZSJRAEYEZ_ooe_
Atbash解码:　IOQWJGLEKGJLWFDHXWRYYXXLPXZFFXHLLHLDTOBMZIZETIAHQIZVBVALLV
JSFuck解码:　RLJDQTOVPTQ606duws5CD6IB5B52CC57okCaUUC3SO4OSOWG3LynarAVGRZSJRAEYEZ_ooe_
JJEncode解码:
BrainFuck解码:
URL解码:　RLJDQTOVPTQ606duws5CD6IB5B52CC57okCaUUC3SO4OSOWG3LynarAVGRZSJRAEYEZ_ooe_
Unicode-str解码:　RLJDQTOVPTQ606duws5CD6IB5B52CC57okCaUUC3SO4OSOWG3LynarAVGRZSJRAEYEZ_ooe_
Unicode-Ascii解码:
Bytes解码:　RLJDQTOVPTQ606duws5CD6IB5B52CC57okCaUUC3SO4OSOWG3LynarAVGRZSJRAEYEZ_ooe_
Escape解码:　RLJDQTOVPTQ606duws5CD6IB5B52CC57okCaUUC3SO4OSOWG3LynarAVGRZSJRAEYEZ_ooe_

```
------------------------
栅栏fence解码:
因数[2, 3, 4, 6, 8, 9, 12, 18, 24, 36]:
分为2栏时，解密结果为: RULUJCD3QSTOO4WOPSTOQW6GO36Ldyunwasr5ACVDG6RIZBS5JBR5A2ECYCE5Z7_ookoCea_
分为3栏时，解密结果为: R53LBLJ5yD2nQCaTCrO5AV7VPoGTkRQCZ6aSOUJ6URdCAu3EwSYsOE54ZCO_DSo6OoIWeBG_
分为4栏时，解密结果为: R5UALCUVJDCGD63RQISZTBOSO54TVBORP5SAT2OEQCWY6CGEO53Z67L_doyouknowCaesar_
分为6栏时，解密结果为: RO5U3JL6BULRJd5CyADu23nEQwCSaYTsCOrEO554AZVC7OV_PDoSGoT6kORoQICWZe6BaGS_
分为8栏时，解密结果为: RT52UOAELQCCUWVYJ6DCCGGED06533RZQ6I7SLZ_TdBoOySoOu5k4nJoVwBCOaRePs5aSrA_
分为9栏时，解密结果为: RPw5oS3GYLTsBkOLREJQ55C4yZZD6C2aOnS_QODCUSaJoT66CUOrRoOdI5CWAAeVuB73GVE_
分为12栏时，解密结果为: ROO555U43AJZLV6CB7UOLVR_JPdD5oCSyGAoDTu62k3OnREoQQwICCSWaZYeT6sBCaOGrSE_
分为18栏时，解密结果为: RQPOwD5CoUSS3aGJYoLTT6s6BCkUOOLrRREoJOQd5I55CC4WyAZAZeDV6uCB27a3OGnVSE__
分为24栏时，解密结果为: RDOTOu56525kU34O3nARJEZoLQVQ6wCIBC7CUSOWLaVZRY_eJTP6dsDB5CoaCOSGyrGSAEo_
分为36栏时，解密结果为: RJQOPQOdw5DI55C5oCUCS4SW3yaAGZJAYZoeLDTVT66usC6BB2C7kaU3OOOGLnrVRSREE_o_

------------------------
凯撒Caesar解码:
key #0: RLTDQTOVPTQ6O6duws5CD6IB5B52CC57okCaUUC3SO4OSOWG3LynarAVGRZSJRAEYEZ ooe
```


Caeser 3解密的base32可解



[BJDCTF2020]RSA

```
c1 = 12641635617803746150332232646354596292707861480200207537199141183624438303757120570096741248020236666965755
79800965654773861639902530012304376625551859614934893044459982067523004642337305305163193255723084908342685949O1
83732303751744004874183062594856870318614289991675980063548316499486908923209627563871554875612702079100567018G9
89929358182061090875681660973923141057175554829261410305056395717088762131671121879625844840653215457275941351T5
36923392592250779499960732353697682418316292338500566993040344885346514140584683591984290846978754734175236547I8
92495204307644586161393228776042015534147913888338316244169120
n1 = 13508774104460209743306714034546704137247627344981133461801953479736017021401725818808462898375994767375627
74949483967194454382240305997807381312244140761253065816894298782025678658300694700171174923019354237057095070S5
30167921702835627122401475251039000775017381633900222474727396823708695063136246115652622259769634591309421761Z6
95482609844261488246412850107309832153775092550112987378276216111580329764200116625478545156105979556288980735G9
68415822567833347454392032653289344684980811283747668439003097647205390506985552229785068802696070118654342813N8
43783907624317274796926248829543413464754127208843070331063037
c2 = 97915337055253515349847745972087732981120468820838754382612258213240421484845495472248708665806140879522380
50222029976135220147369834521210738600548513023435177567327010266670627659062776268792154579363307996988127559I3
05755762093017277885911653857120710042499083850825512761663733449968005864541178692530236879041476824861180935B1
60197554369255458675450109457987698749584630551177577492043403656419968285163536823819817573531356497236154342G8
99145253216738079254586518547685123963553897408632701487753627444481155816396293263623421605485000350001560972I5
44688125105550546571385417391314204097638250043518544252172I
n2 = 12806210903061368369054309575159360374022344774547459345216907128193957592938071815865954073287532545947370
67183837214480653975382948435606491935728562330520960068057097522463921439680512435086277215927236277876803684A6
34760917612708721787320159318432456050806227784435091161119982613987303255995543165395426658059462110056431392S1
75487174478980849151676611723629842512016886394696522834523077128213988570164875907949965444688267056003322085I5
20144332226729874711752888298595537524642481261647832718239946170997889346409324513553013543000784222338936021Z8
03439850867615121148050034887767584693608776323252233254261047
```

```python
import gmpy2
from Crypto.Util.number import *
q = (gmpy2.gcd(n1,n2))
p = n1//q

# output=38163126882580646951816637038735203547577567716361573075945434391356361597088196733240770990123563771893618419893022630376187651710120867710731100606572801422047796600062096405661605867699987897694331906383664908508537757727321479237154877520459409788707889859846389244014157797454493926824781893793660701310080816975867504226456854776403162843141472792216858099849469580040304331240664352763766746631847366954232616921866536642304357900338848663416764266349589660728215580833190235118850019796090567220704657964705276457941181430568913751986088091646727205677864144275894013501640080874038714450815635806795521501 8
# for i in range(100000):
#  res=pow(294,i,n1)
#  if (res==output):
#   #print(i)
#   #52361
#   e=i
#   break
e = 52361
phi_n = (p-1)*(q-1)
d = gmpy2.invert(e,phi_n)
m = pow(c1,d,n1)

print(long_to_bytes(m))
```

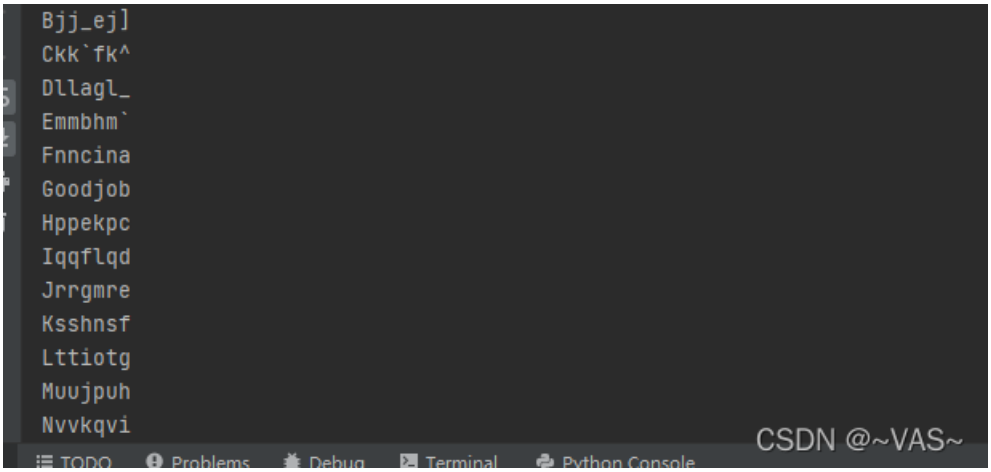
[MRCTF2020]天干地支+甲子

60年甲子

```
六十年甲子（干支表）
1       2       3       4       5       6       7       8       9       10
甲子 乙丑 丙寅 丁卯 戊辰 己巳 庚午 辛未 壬申 癸酉
11      12      13      14      15      16      17      18      19      20
甲戌 乙亥 丙子 丁丑 戊寅 己卯 庚辰 辛巳 壬午 癸未
21      22      23      24      25      26      27      28      29      30
甲申 乙酉 丙戌 丁亥 戊子 己丑 庚寅 辛卯 壬辰 癸巳
31      32      33      34      35      36      37      38      39      40
甲午 乙未 丙申 丁酉 戊戌 己亥 庚子 辛丑 壬寅 癸卯
41      42      43      44      45      46      47      48      49      50
甲辰 乙巳 丙午 丁未 戊申 己酉 庚戌 辛亥 壬子 癸丑
51      52      53      54      55      56      57      58      59      60
甲寅 乙卯 丙辰 丁巳 戊午 己未 庚申 辛酉 壬戌 癸亥
```

爆破一下位移了多少，flag是Goodjob

```
a = '11  51  51  40  46  51  38'.split()
for i in range(128):
    for j in a:
        print(chr(int(j)+i),end='')
    print( )
```

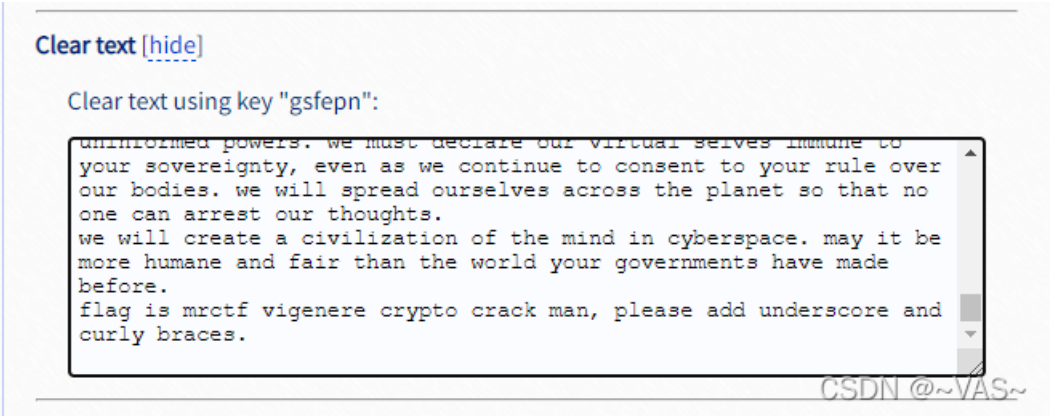


[BJDCTF2020]rsa_output

```
n1,e1 = 21058339337354287847534107544613605305015441090508924094198816691219103399526800112802416383088995253908
85746026672692561582689530337780161482936403462447519585999794314630558831593913077745048519629076624961234005435
54622516207681542973756257677388091926549655162490873849955783768663029138647079874278240867932127196686258800145
69116207307067341036118331797332640964752864919880639904310853804990750056298077024066767078413246609711732531000
95636252834668475295993747385263014589379605667579364643079357826541825591937632379604458855972670385842931178470
52450698459383168026815756536537708836155257356903066746351671111,2767

n2,e2 = 21058339337354287847534107544613605305015441090508924094198816691219103399526800112802416383088995253908
85746026672692561582689530337780161482936403462447519585999794314630558831593913077745048519629076624961234005435
54622516207681542973756257677388091926549655162490873849955783768663029138647079874278240867932127196686258800145
69116207307067341036118331797332640964752864919880639904310853804990750056298077024066767078413246609711732531000
95636252834668475295993747385263014589379605667579364643079357826541825591937632379604458855972670385842931178470
52450698459383168026815756536537708836155257356903066746351671111, 3659

c1=20152490165522401747723193966902181151098731763998057421967155300933719378216342043730801302534978403741086887
79690407219595331900583427620573594326637178258263654449969154690390564284161661739209582430448314049241134425126
61759942687614118421212167750037123693712757180289132170658761039363944686883698717030181301821840888696826382123
23084155607494076330256934285171370758586535415136162861138898728910585138378884530819857478609791126971308624318
45459059929194053557514927891100093131384172651261172737108138439231433812762048025159105274688832242748299624796
36527422350190210717694762908096944600267033351813929448599

c2=11298697323140988812057735324285908480504721454145796535014418738959035245600679947297874517818928181509081545
02705652379002259823391801126101197319638639568937152677478558232612195918619558606985159246763781936662404413366
01016373360885158956955263645614345881350494012328275215821306955212788282617812686548883151066866149060363482958
70836472698290879834018228870210102339383978142738653723045943651261304731158587506800821081899694146015658931413
50104383624475224282068849449526398266772478190668127068357731070595670828223123007210498270136604186102651892888
40247186598145741724084351633508492707755206886202876227
from gmpy2 import invert
def gm(n, c1, c2, e1, e2):
    def egcd(a, b):
        if b == 0:
            return a, 0
        else:
            x, y = egcd(b, a % b)
            return y, x - (a // b) * y
    s = egcd(e1, e2)
    s1 = s[0]
    s2 = s[1]

    if s1 < 0:
        s1 = - s1
        c1 = invert(c1, n)
    elif s2 < 0:
        s2 = - s2
        c2 = invert(c2, n)
    m = pow(c1, s1, n) * pow(c2, s2, n) % n
    return m
m = gm(n1,c1,c2,e1,e2)
from Crypto.Util.number import *
print(long_to_bytes(m))
```

[MRCTF2020]vigenere

维吉尼亚解密



[ACTF新生赛2020]crypto-rsa0

```
e=65537
p=90185880664342063772402771624767392713862401730886765262953151639909683470229228412991282745514829264909083992
37153883494964743436193853978459947060210411
q=75470056738777382578357297600377652133400366963507663242291436131799321451221306857785040624101370436359582088
05698698169847293520149572605026492751740223
c=50996206925961019415256003394743594106061473865032792073035954925875056079762626648452348856255575840166640519
33486269006394931651575025654593749821347628663745580345289078126444603073236987104487035983856861817658620604l0
55000297981733272816089806014400846392307742065559331874972274844992047849472203390350
import gmpy2
phi = (p-1)*(q-1)
d = gmpy2.invert(e,phi)
m = pow(c,d,p*q)
import libnum
from Crypto.Util.number import *
print(long_to_bytes(m))
```

[BJDCTF2020]signin

[MRCTF2020]keyboard

正常解密是mobilephond提交不对，正确flag是mobilephone

RSA4

```
import gmpy2
```

```python
import  binascii

#利用中国剩余定理求解同余方程，aList：余数，mList：模数
def CRT(aList, mList):
    M = 1
    for i in mList:
        M = M * i    #计算M = ∏ mi
    #print(M)
    x = 0
    for i in range(len(mList)):
        Mi = M // mList[i]    #计算Mi
        Mi_inverse = gmpy2.invert(Mi, mList[i]) #计算Mi的逆元
        x += aList[i] * Mi * Mi_inverse #构造x各项
    x = x % M
    return x

if __name__ == "__main__":
    #========== n c ==========
    n1 = "331310324212000030020214312244232222400142410423413104441140203003243002104333214202031202212403400220
03120214232243410414310424424121420444444332300024413012202242231020110441104403011330232301410133121430322331240
02430402404413033243132101010422240133122211400434023222214231402403403200012221023341333340042343122302113410210
01102212332413030244313300013034040201044424431201300003341100424320102034014404040100034420012230422114420014130
004"
    c1 = "310020004234033304244200421414413320341301002123030311202340222410301423440312412440240244110200112141
140201224032402232131204213012303204422003300004011434102141321223311243242010014140422411342304322201241112402
322031011312212230040220031200021102300233411432014043113403111342301402314122013333331424024231343321130210241
311111142443003244012334003404431422340040122411132300024223442044124041102102310022200312321434303012203230104
243"
    n2 = "302240000040421410144422133334143140011011044322223144412002220243001141141114123223331331304421113021
231204322233120121444434210041232214144413244434424302311222143224402302432102242132244032010020113224011121043
321432212034242431340443140222120243431000423420024323311443002142124140334141200043442113302240203012230333343
424403120424012230124223201130321122004422241113440301213242031111030244234402112210122441123000220334414014304
114"
    c2 = "112200203404013430330214124004404423210041321043000303233141423344144222343401042200334033203124030011
440014210112103234440312134032123400444344144233020130110134042102220302002413321102022414130443041144240310121
020100310104334204234412411424420321211112232031121330310333414423433343322024400121200333330432223421433344122
301244001304140142322021012402443104001341431312112343342411311341442204333042200231414411113414204433340411224
344"
    n3 = "332200324410041111434222123043121331442103233332422341041340412034230003314420311333101344231212130200
312041044324431141033004333110021013020140020011222012300020041342040004002220210223122111314112124333211132230
321240224231412140313031444441344030244201114232444240300300033402130321213032133430204013042433300013140230301
210341133344044404212422401131032030133412313300043320403024400113240041303240343234301431024014401302423214240
323"
    c3 = "100134441201411303224332041240022422243323340111242100124402414023421004103311314413032420110021013230
404033111204213044222222003244022442433224224444140433421301111113300222132030303244221011330322120420422431014
343422032041210421132121042124233303311343113111141432000112400021113121222343400034033120404010430214331120313
324322123304112340014030132021432101130211241134422413442312013042141212003102211300321404043012124332013240431
242"

    cList = [int(c1,5), int(c2,5), int(c3,5)]
    nList = [int(n1,5), int(n2,5), int(n3,5)]
    m_e = CRT(cList, nList) #计算m^e
    for e in range(1, 10):  #遍历e求解
        m, f = gmpy2.iroot(m_e, e) #m_e开e次根
        print("加密指数e = %d: "%e)
        m = hex(m)[2:]
        if len(m)%2 == 1:
            m = m + '0' #binascii.unhexlify()参数长度必须为偶数，因此做一下处理
        flag = binascii.unhexlify(m)
        print(flag)
```

```
        print(flag)
```

[WUSTCTF2020]babyrsa

```
c = 28767758880940662779934612526152562406674613203406706867456395986985664083182
n = 73069886771625642807435783661014062604264768481735145873508846925735521695159
e = 65537
p = 189239861511125143212536989589123569301
q = 38612312537192365119121986981129358645 9
phi = (p-1)*(q-1)
import gmpy2
d = gmpy2.invert(e,phi)
from Crypto.Util.number import *
m = pow(c,d,n)
print(long_to_bytes(m))
```

[GWCTF 2019]BabyRSA

```
N=6365851495945747469090301601826908662229092564648472917830006518372279213372378996512879435977732709443840348589252957448807271016068414136400065276148731106514101558937765487378231529437978847291301497582791274300447392540004266109228345730949570825895394456108282794288145243134912620619305128290744662326331305991044908935720939438327403018096308475415925489212002882224327892086509499376383034294564688891001926138590737529238124542122399089489301783553313909335367710657918176439787630450308337123261628838106381200293783370929386621741197476878994846036283440794935566014224984053607319581627192961605840426710571602412848525229136762645962019061636m1=900099743414522432169869380283712575286049432089411765187174635547749678781526945864693777652961131656594987260127122886704588843739714198427509292876586402662196866469569298721157821730939797429587451216719285687094685260987159271898296004972831180516411073051288526970320533681151812160696266061655034651257252048755787012377892929662118240027614818152766662368690051291388627824768591030867260918604976148832829499550232224143332431932685647816216998704125578224043812138040266858312214307282907555978192593396166501586747132488416543385151994055320031737325204578139011702647130851070770014780833413390020698705853782570511502175117557614910215532394m2=48744398575740517342662818837565711760423550793696752299325797210887228369830523845446572321422687141427678891205818619703982124291273674282408062768097180251120691439467215924020691073585065199931610001469106729570813863936320359624469399556278028663711639473825077412975902108019732372480541466804231880601065281440507876973854891367546618155100552706530951536495061013720639325714835765966668709166274984856022545382636227170429269284759633953322908803882053208610942115857584107760126871317509787408353624900601894878941323878392284563349402360886525607196285658122989004389693902561360056428339132933145219906285893037456599163419149513793957453954696m = pow(c,d,N)
```

SameMod

```
N=6365851495945747469090301601826908662229092564648472917830006518372279213372378996512879435977732709443840348589252957448807271016068414136400065276148731106514101558937765487378231529437978847291301497582791274300447392540004266109228345730949570825895394456108282794288145243134912620619305128290744662326331305991044908935720939438327403018096308475415925489212002882224327892086509499376383034294564688891001926138590737529238124542122399089489301783553313909335367710657918176439787630450308337123261628838106381200293783370929386621741197476878994846036283440794935566014224984053607319581627192961605840426710571602412848525229136762645962019061636m1=900099743414522432169869380283712575286049432089411765187174635547749678781526945864693777652961131656594987260127122886704588843739714198427509292876586402662196866469569298721157821730939797429587451216719285687094685260987159271898296004972831180516411073051288526970320533681151812160696266061655034651257252048755787012377892929662118240027614818152766662368690051291388627824768591030867260918604976148832829499550232224143332431932685647816216998704125578224043812138040266858312214307282907555978192593396166501586747132488416543385151994055320031737325204578139011702647130851070770014780833413390020698705853782570511502175117557614910215532394m2=48744398575740517342662818837565711760423550793696752299325797210887228369830523845446572321422687141427678891205818619703982124291273674282408062768097180251120691439467215924020691073585065199931610001469106729570813863936320359624469399556278028663711639473825077412975902108019732372480541466804231880601065281440507876973854891367546618155100552706530951536495061013720639325714835765966668709166274984856022545382636227170429269284759633953322908803882053208610942115857584107760126871317509787408353624900601894878941323878392284563349402360886525607196285658122989004389693902561360056428339132933145219906285893037456599163419149513793957453954696p = 797862863902421984951231350430312260517773269684958456342860983236184129602390919026048496119757187702076499551310794177917920137646835888862706126924088411570997141257159563952725882214181185531209186972351469946269508511312863779123205322378452194261217016552527754513215520329499967108196968833163329724620251096080377748737q = 797862863902421984951231350430312260517773269684958456342860983236184129602390919026048496119757187702076499551310794177917920137646835888862706126924088411570997141257159563952725882214181185531209186972351469946269508511312863779123205322378452194261217016552527754513215520329499967108196968833163329724620251096080377747699e = 65537
import gmpy2
import sympy
from Crypto.Util.number import *
phi = (p-1)*(q-1)
d = gmpy2.invert(e,phi)
#m = pow(c,d,N)

F1 = sympy.Symbol('F1')
F2 = sympy.Symbol('F2')
c1 = pow(m1,d,N)
c2 = pow(m2,d,N)
f1 = F1 + F2 - c1
f2 = pow(F1, 3) + pow(F2, 3) -c2
res = sympy.solve([f1,f2],[F1,F2])
flag1 = (long_to_bytes(res[0][1]))
flag2 = (long_to_bytes(res[1][1]))
print(flag1+flag2)
```

```
n1,e1 = 626656572072690726599724135833158541709572614634198975553801712298136074281349840153359475708879653634194165969125932306563124 9,773
n2,e2 = 626656572072690726599724135833158541709572614634198975553801712298136074281349840153359475708879653634194165969125932306563124 9,839

c1=345352059272344393545115154524502586423238887172168232640891502434980406204197670236472866068291239690396819398113155311153734 9
c2=567281802681629334407011933253662961945716357003630529686905353229310537969079338601906575446529286776952173641417080323830953 5
from gmpy2 import invert
def gm(n, c1, c2, e1, e2):
    def egcd(a, b):
        if b == 0:
            return a, 0
        else:
            x, y = egcd(b, a % b)
            return y, x - (a // b) * y
    s = egcd(e1, e2)
    s1 = s[0]
    s2 = s[1]

    if s1 < 0:
        s1 = - s1
        c1 = invert(c1, n)
    elif s2 < 0:
        s2 = - s2
        c2 = invert(c2, n)
    m = pow(c1, s1, n) * pow(c2, s2, n) % n
    return m
m = gm(n1,c1,c2,e1,e2)
m = str(m)
flag=""
i=0
while i < len(m):
    if m[i]=='1':
        c=chr(int(m[i:i+3]))
        i+=3
    else:
        c=chr(int(m[i:i+2]))
        i+=2
    flag+=c
print(flag)
```

一张谍报

```
str1 = '今天上午，朝歌区梆子公司决定，在每天三更天不亮免费在各大小区门口设卡为全城提供二次震耳欲聋的敲更提醒，呼吁大家早睡早起，不要因为贪睡断送大好人生，时代的符号是前进。为此，全区老人都蹲在该公司东边树丛合力抵制，不给公司人员放行，场面混乱。李罗鹰住进朝歌区五十年了，人称老鹰头，几年孙子李虎南刚从东北当猎户回来，每月还寄回来几块鼹鼠干。李罗鹰当年遇到的老婆是朝歌一枝花，所以李南虎是长得非常秀气的一个汉子。李罗鹰表示：无论梆子公司做的对错，反正不能打扰他孙子睡觉，子曰：‘睡觉乃人之常情’。梆子公司这是连菩萨睡觉都不放过啊。李南虎表示：梆子公司智商捉急，小心居民猴急跳墙！这三伏天都不给睡觉，这不扯淡么！到了中午人群仍未离散，更有人提议要烧掉这个公司，公司高层似乎恨不得找个洞钻进去。直到治安人员出现才疏散人群归家，但是李南虎仍旧表示爷爷年纪大了，睡不好对身体不好。'
str2 = "喵天上午，汪歌区哗叽公司决定，在每天八哇天不全免费在各大小区门脑设卡为全城提供双次震耳欲聋的敲哇提醒，呼吁大家早睡早起，不要因为贪睡断送大好人生，时代的编号是前进。为此，全区眠人都足在该公司流边草丛合力抵制，不给公司人员放行，场面混乱。李罗鸟住进汪歌区五十年了，人称眠鸟顶，几年孙叽李熬值刚从流北当屁户回来，每月还寄回来几块报信干。李罗鸟当年遇到的眠婆是汪歌一枝花，所以李值熬是长得非常秀气的一个汉叽。李罗鸟表示：无论哗叽公司做的对错，反正不能打扰他孙叽睡觉，叽叶：‘睡觉乃人之常情’。哗叽公司这是连衣服睡觉都不放过啊。李值熬表示：哗叽公司智商捉急，小心居民猴急跳墙！这八伏天都不给睡觉，这不扯淡么！到了中午人群仍未离散，哇有人提议要烧掉这个公司，公司高层似乎恨不得找个洞钻进去。直到治安人员出现才疏散人群归家，但是李值熬仍旧表示爷爷年纪大了，睡不好对身体不好。"
str3 = '喵汪哗叽双哇顶，眠鸟足屁流脑，八哇报信断流脑全叽，眠鸟进北脑上草，八枝遇孙叽，孙叽对熬编叶：值天衣服放鸟捉猴顶。鸟对：北汪罗汉伏熬乱天门。合编放行，卡编扯呼。人离烧草，报信归洞，孙叽找爷爷。'



res = ""
for i in range(len(str3)):
    for j in range(len(str2)):
        if str3[i] == str2[j]:
            res += str1[j]
            break
print(res)
## 今朝梆子二更头，老鹰蹲猎东口，三更鼹鼠断东口亮子，老鹰进北口上树，三枝遇孙子，孙子对虎符曰：南天菩萨放鹰捉猴头。鹰对：北朝罗汉伏虎乱天门。合符放行，卡符扯呼。人离烧树，鼹鼠归洞，孙子找爷爷。
# flag{南天菩萨放鹰捉猴头}
```

[BJDCTF2020]easyrsa

```
c = 792254786685776145980749150265421628301277617778951154935067295810181028134840228409831014779654943068925380
351099487742013553726854941065265447962085869132411036718202564878840704159994309138622754318215774620294709957 2
389676084392706406084307657000104665696654409155006313203957292885743791715198781974205578654792123191584957665 2
932083904537483691823331528098823124533597061478081989229167627737217266815889771038774541190437448891645293831 8
807749919493290964391869664687690732736475138095318251788313459181080084897171918480871369434298545810300667601 3
451912221080252735948993692674899399826084848622145815461035
z = 321157486776232096674716228721852750702579247660150200728052673598390593932843165958829333722897321272740764
345875193333001424730103446948038851685575488012024959332262154377633292802421135565244984575595628729008116020 5
694442396740377762330696188075761324632872961664303262896407293127208586692804597379937471184682515778105696516 4
178505232524245809179235607571567174228822561697888645968559343608375331988097157145264357626738141646556353500 9
949241158757481983180362968986040970009382721959030567335658801505402753692396377939759233295987160033503082593 2
143675257929100035556043154222969975995514115291470836249448 2
n = 153107451613368954134066900093247662007891792488969519420472354489016123511284593091458255475692984798211012
490941618672076865376070474479687087589909501363809247473590525705495940985699706328543518259507297525635022848 4
926373012758638252270395989339232933376092763735305225027419582146902340144384139509641023184359210142659188257 3
405934188675124326997277775238287928403743324297705151732524641213516306585297722190780088180705070359469719869 3
439391065292047982859575168607743840018927775259161677432724199585720553322320560959794481550824659777814825983 7
199479887191751476750839473044797477032996768176762549539444 1
e=65537
import gmpy2
from Crypto.Util.number import *
pq1 = gmpy2.iroot(z+2*n,2)[0]
pq2 = gmpy2.iroot(z-2*n,2)[0]
q = (pq1-pq2)//2
p = (pq1+pq2)//2
phi = (p-1)*(q-1)
d = gmpy2.invert(e,phi)
m = pow(c,d,n)
print(long_to_bytes(m))
```