# 上海市大学生网络安全大赛2020——baby_dsa

原创

te_mgl 于 2020-11-17 19:52:11 发布 

分类专栏： 密码学 文章标签： 密码学



本文链接：https://blog.csdn.net/weixin_44617902/article/details/109750766





## DSA加密

task.py

```
#!/usr/bin/env python
from Crypto.Util.number import *
from hashlib import sha512,md5
from os import urandom
import random

def hash(message):
 return int(sha512(message).hexdigest(), 16)

def key_gen():
 q = getPrime(256)
 while True:
  p = random.getrandbits(2816)*q + 1
  if isPrime(p):
   print(p.bit_length())
   break
 while True:
  g = pow(random.randrange(1, p-1), (p-1)/q, p)
  if g != 1:
   break
 x = random.randrange(1, q)
 y = pow(g, x, p)
 pubkey = (p, q, g, y)
 privkey = x
 return pubkey, privkey

def sign(message, pubkey, privkey):
 p, q, g, y = pubkey
 x = privkey
 k = pow(y, x, g) * random.randrange(1, 512) % q
 r = pow(g, k, p) % q
 s = inverse(k, q) * (hash(message) + x * r) % q
 return r, s

def verify(message, signature, pubkey):
```

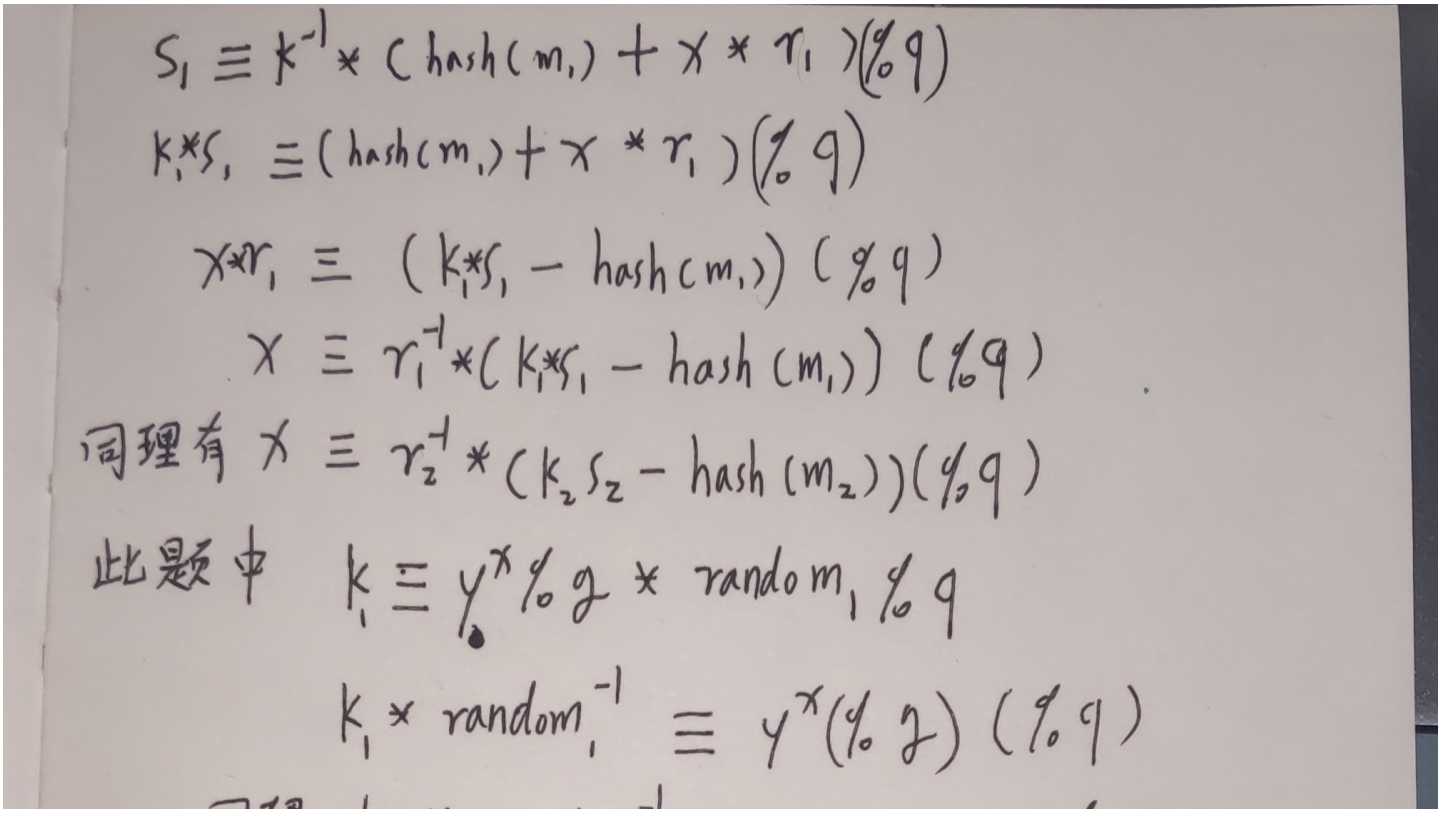
```
 p, q, g, y = pubkey
 r, s = signature
 if not (0 < r < q) or not (0 < s < q):
  return False
 w = inverse(s, q)
 u1 = (hash(message) * w) % q
 u2 = (r * w) % q
 v = ((pow(g, u1, p) * pow(y, u2, p)) % p) % q
 return v == r

pubkey, privkey = key_gen()
print(pubkey)

message1 = urandom(16).encode('hex')
signature1 = sign(message1, pubkey, privkey)
print(message1,signature1)
print(verify(message1, signature1, pubkey))

message2 = urandom(16).encode('hex')
signature2 = sign(message2, pubkey, privkey)
print(message2,signature2)
print(verify(message2, signature2, pubkey))

flag = 'flag{'+md5(long_to_bytes(privkey)).hexdigest()+'}'
```

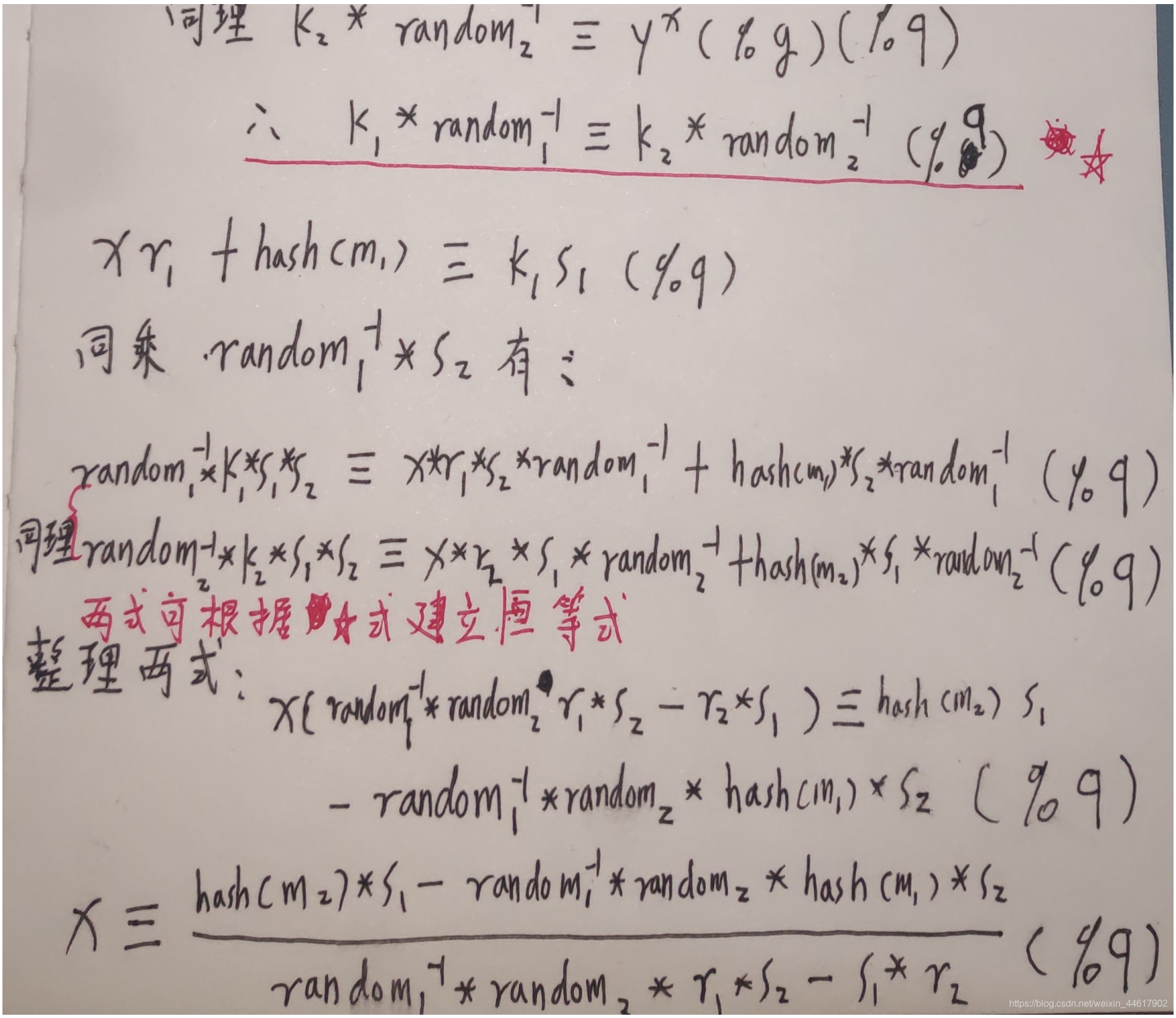
data

```
(32972264630373244580088372844989633726490388893906850518496801750165056460017612201098589216242660440351331341354025612356358334282068868883080277723530307674009210783468683772984012138120532503160020339416922721926446132522965798845167315604365010732539244576465586988554847817470293977551116332975872159765796334519336582353853865395180065700696535751460600168119111406146064719303273413685829798360425854068113522363260652926364845508072137564821530844277145496942646856959775315374256822121555535688486660885769328882346593552136649097537817539174011619777626636580975044119149080816770339809150390795177661597605222612791153473858130094375101568989697695636878694957219772654447995856340198809515320802179604569017889184392657881699100628228895801993664174551865954899730003517702004850950084942288293001450396959369463795856250514025530349712074747624631477444673601588475933560307451941432762549494636506982105155695331, 82302835442112137125891403368151249910268706824854786126600390413622302196443L, 1156233264299340971106498371495495695225880592354374034142195518472540521911699506391311324676590685365234887170345722135060009885002334748836477169806166169806180231794918961214520698361874839110454610266388341977984902756569838594616255112661600466818870137432772800368859461445854700956291885576855069405183771903076277144927769029433730710613058788277691211698675287829143272152835171859480781061918556840079857761203012054552142555673071865310355331986288606422711525790877591376770834180618492794265362178603111236615495225612101250344421932588038244804199229449738675082560512062564365473035097263889257937140778993389305893378514344032352806521972367991027459721160744835688761657797398841523104074451793557924512992305640697344011520550723893828185707635141404445213445935586965289450282024222064488965878769991566367115153619761583843561579531705057955933288008556165952066173304891391375100346312776539530448611005L, 2909996237877318126977196918520612902466194134636363123821469699005463845147107828431539627048519160916016790288308661763323315195151563014015371730699081815090284643226473522566324246848093491210242625970069137074838111176441974819590537854750834064725830991405065050713001933560024430077502205249322191919329692022703433239550352913968084726866847876105591147020547846993654908603927370610562331603089432964785407983531348789370883366729281628943329617622775593458604799162480868211178119903910251871251930740590010864413059771332527746989966531222971234478374491686573473080162700308813956740664211440029597519368391669357262008337851327363288597103518713525675115161421709560918853521785793022996343222548183839785857731366925889229760436173379045453961467556092841637434762977726865484751701976054128476895871715224532290559327127141548699894548085614588520317691194892355984020669240827783760814946322584484340485620053L)
('0234e7971889def7e60348f77db94b7a', (108592362699597657352363937799363052173055743318392345021902267089299915823861, 13707557323895695260471053137828523837745895683218331343360027380310980108819L))
True
('16c5ac270b72f70319657b4410d985d4', (419606422463790676405247094160015360582928173191097643173697772244262187465181, 74676725322515593502346275468843411563746982149373670021082686341369076719088L))
True
```

这题比赛时没做出来卡住了，赛后看了大佬文章豁然开朗。

我也是第一次遇到dsa加密的题，看了几篇博客大概过程弄懂。推导出了x的两个关系式，k与x有关，也想着建立联系，但是想半天也没想出来hhhh还是太菜了（继续加油吧）。看了大佬的解题过程后一下就明白了，自己也写了下过程记录下这个题。






代码：

```
from Crypto.Util.number import *
from hashlib import sha512,md5

def hash(message):
    return int(sha512(message).hexdigest(), 16)

def sign(message, pubkey, privkey, random1):
    p, q, g, y = pubkey
    x = privkey
    k = pow(y, x, g) * random1 % q
    r = pow(g, k, p) % q
    s = inverse(k, q) * (hash(message) + x * r) % q
    return r, s
public = (32972264630373244580088372844989633726490388893906850518496801750165056460017612201098589216242660440351331341354025612356358334282068868883080277723530307674009210783468683772984012138120532503160020339416922721926446132522965798845167315604365010732539244576465586988554847817470293977551116332975872159765796334519336582353853865395180065700696535751460600168119111406146064719303273413685829798360425854068113522363260652926364845508072137564821530844277145496942646856959775315374256822121555535688486660885769328882346593552136649097537817539174011619777626636580975044119149080816770339809150390795177661597605222612791153473858130094375101568989697695636878694957219772654447995856340198809515320802179604569017889184392657881699100628228958019936641745518659548997300035177020048509500849422882930014503969593694637958562505140255303497120747476246314774446736015884759335603 0
```

```
74519414327625494946365069821051556953 3, 82302835442112137125891403368151249910268706824854786126600390413622302
196443, 11562332642993409711064983714954956952258805923543740341421955184725405219116995063913113246765906853652
34887170345722135060009885002334748836477169806166169806180231794918961214520698361874839110454610266388341977 98
49027565698385946162551126616004668188701374327728003688594614458547009562918855768550694051837719030762771449 27
76902943373071061305878827769121169867528782914327215283517185948078106191855684007985776120301205455214255567 30
71865310355331986288606422711525790877591376770834180618492794265362178603111236615495225612101250344421932588 03
82448041992294497386750825605120625643654730350972638892579371407789933893058933785143440323528065219723679910 27
45972116074483568876165779739884152310407445179355792451299230564069734401152055072389382818570763514140444521 34
45935586965289450282024222064488965878769991566367115153619761583843561579531705057955933288008556165952066173 30
48913913751003463127765395304486110 05, 29099962378773181269771969185206129024661941346363631238214696990054638 45
14710782843153962704851916091601679028830866176332331519515156301401537173069908181509028464322647352256632424 68
48093491210242625970069137074838111176441974819590537854750834064725830991405065050713001933560024430077502205 24
93221919193296920227034332395503529139680847268668478761055911470205478469936549086039273706105623316030894329 64
78540798353134878937088336672928162894332961762277559345860479916248086821117811990391025187125193074059001086 44
13059771332527746989966531222971234478374491686573473080162700308813956740664211440029597519368391669357262008 33
78513273632885971035187135256751151614217095609188535217857930229963432225481838397858577313669258892297604361 73
37904545396146755609284163743476297772686548475170197605412847689587171522453229055932712714154869989454808561 45
88520317691194892355984020669240827783760814946322584484340485620 53)
p, q, g, y = public
tmp1 = (b'0234e7971889def7e60348f77db94b7a', (108592362699597657352363937799363052173055743318392345021902267089
29991582386, 13707557323895695260471053137828523837745895683218331343360027380310980108819))
tmp2 = (b'16c5ac270b72f70319657b4410d985d4', (419606422463790676405247094160015360582928173191097643173697772244
26218746518, 74676725322515593502346275468843411563746982149373670021082686341369076719088))
message1, r1, s1 = tmp1[0], tmp1[1][0], tmp1[1][1]
message2, r2, s2 = tmp2[0], tmp2[1][0], tmp2[1][1]
hm1, hm2 = hash(message1), hash(message2)

for random1 in range(1, 512):
    for random2 in range(1, 512):
        random1_inv = inverse(random1, q)
        random_mul = random1_inv * random2
        x = ((s1 * hm2 - s2 * hm1 * random_mul) * inverse(s2 * r1 * random_mul - s1 * r2, q))%q

        if sign(message1, public, x, random1) == tmp1[1]:
            print(x)
            flag = 'flag{'+md5(long_to_bytes(x)).hexdigest()+'}'
            print(flag)
            print(random2)
            exit()
    print(random1)


# 成功拿到flag后可以知道random1 = 36, random2 = 58 ^_^
```