西湖论剑_unknown_dsa_复现

原创

M3ng@L 于 2022-01-03 19:44:09 发布 2919 收藏 1

分类专栏： CTF比赛复现 文章标签： 密码学 python

本文链接：https://blog.csdn.net/qq_51999772/article/details/122291477

unknown_dsa

题目描述

```
from Crypto.Util.number import *
from Crypto.PublicKey import DSA
from Crypto.Hash import SHA
from gmpy2 import invert,powmod
import random
from secret import flag,m1,m2,ul,vl,wl

def encrypt():
    key = DSA.generate(int(1024))
    q = key.q
    p = key.p
    g = key.g
    x1 = bytes_to_long(flag[:len(flag)//2])
    x2 = bytes_to_long(flag[len(flag)//2:])
    assert x1<q and x2<q
    t = powmod(g, p*q-(p+q), p*q)
    hm1 = bytes_to_long(SHA.new(m1).digest())
    hm2 = bytes_to_long(SHA.new(m2).digest())
    k = random.randint(1, q-1)
    r1 = powmod(g, k, p) % q
    s1 = (hm1 + x1*r1) * invert(k, q) % q
    s2 = (hm2 + x1*r1) * invert(k, q) % q
    r2 = powmod(g, x1, p) % q
    s3 = (hm1 + x2*r2) * invert(k, q) % q
    print(p*q, (p-1)//q, t, sep=', ')
    print(r1, s1, s2, sep=', ')
    print(r2, s3, sep=', ')

def main():
```

```
    for i in range(len(ul)):
        assert ul[i]**2 - wl[i]* vl[i]**2==1
    e = 7
    cl1 = [int(powmod(bytes_to_long(m1), e, x)) for x in ul]
    cl2 = [int(powmod(bytes_to_long(m2), e, y)) for y in vl]
    print(wl, cl1, cl2, sep=', ')

    encrypt()

if __name__ == '__main__':
    main()

'''
[3912956711, 4013184893, 3260747771], [285258922377992879626654060042167879088906728491168257892421618605259039359564532216156338661551247525672638436509171103444968279126899462375893775287475091820096188899708247710081102572189872078366686862349824621967722110622766089551905863196505579070913020776070 4, 211158499061801396563106646074584256376705200819832482589841660262228987535050089041366888200757204110041582641386597621018735885836864733889517447339367697326172796497970851520578802337219 61, 30189917909218596478584770516695018125567727229437782304501120503531846349668278828965117763534189430853778744914819958349011705952697175980442697794795272126688075717705533508877769313469371334564020654067012387221017868030610086535505914621928112430346010542 4], [14805245002940976705662351036536660222877843156928840757713198043507452963271501497113345262602122694463228247931237866735379211713345206997233416938683722728592401118703567187475890102871950516388778938283577066421804574346522278885925827282621786987760731414 4, 16436318503180551519469383813896710397388249532728164023710951180471797588467030709318502386682626254448265648334522948071105444415378301997520500406974409481460927237136611253099942752 56, 10949587016016795940445976198460149258144635366996455598605244743540728764635947061037779912661207322820180541114179612916018317600403816027703391110922112311910900034442340387304006761589708943814396303183085858356961537279163175384848010568152485779372842]
851986153860756075670700209699817778276718736546312004720782419807378344388979001462488402791911391564165371083996828743706298882073345062370400178383135589112750739041484515402557058184775811828662694130182630798586802216473416807629890804180399727047590033436166524754381558068587359823529307712448809901903185269332674552489137822979916850411875651408 59, 10623995021320631630168390754576391633605524395570621094473647242596520010346142178180473167843011633370209977785527946913721916529372550088759028035597310758074521236893751407005999184894803171825380469462182173495760483812521095171152715126500073689660702919 8, 60132176395922896902518845244051065417143507550519860211077965501783315971109433544482411208238485135554065241864956361676878220342500208011089383751225437417049893725546176799417188875972677293680033005399883113531193705353404892141811493415079755456185858898014563869108922398697328052738792810946133296453262872057366145463111436355800514444465761045484988411946173276504454310516859641743992277393 76, 37659916692187611899413218566020315198350067089 6, 18770515984397310296359315120436113933504832924362082788141549313630907130298691484422077685628 2, 67473536025000431526798842443574113204760753502 9
'''
```

涉及的知识点按解题顺序有Pell方程， 中国剩余定理模数不互素的情况， DSA算法以及k共享情况

Pell方程

(10条消息) Pell方程_M3ng@L的博客-CSDN博客

首先通过求解Pell方程来算得vl，ul；也就是在之后RSA加密中使用的模数

中国剩余定理模数不互素的情况

(10条消息) 中国剩余定理模数不互素的情况_M3ng@L的博客-CSDN博客

这里使用低加密指数广播攻击，且模数不互素，推导过程在上面这篇文章中

可以算得消息$m_{}$,$m_{}$

这样就可以知道稍后在DSA算法中用到得$H(m_{})$,$H(m_{})$

DSA算法

(10条消息) DSA算法_M3ng@L的博客-CSDN博客

先求 q

由于已知 $p*$ 和 $(p-1)//q$

两者相乘得到 $p*($ ，在对这个一元二次方程求解即可得到

那么 q 也可以得到

有三组DSA算法

其中两组使用了相同的 $r1$, k, $x1$

使用 `k共享` 的攻击方式，通过 $k=(H(m_1)-H(m_2))*($ 求得

那么 k, $r1$, $s1$, $s2$ 已知，可以通过同余式 $x1\equiv r1 * (k*$ 求得

剩下的一组也是使用的相同的 k

由于 $r2$, $s3$ 已知，那么可以求得 $x2$

整道题的代码实现

```
from Crypto.Util.number import *
import gmpy2,sympy
from Crypto.Hash import SHA

wl = [3912956711, 4013184893, 3260747771]
cl1 = [2852589223779928796266540600421678790889067284911682578924216186052590393595645322161563386615512475256726384365091711034449682791268994623758937752874750918200961888997082477100811025721898720783666868623498246219677221106227660895519058631965055790709130207760704, 21115849906180139656310664607458425637670520081983248258984166026222898753505008904136688820075720411004158264138659762101873588583686473388951744733936769732617279649797085152057880233721961, 301899179092185964785847705166950181255677272294377823045011205035318463496682788289651177635341894308537787449148199583490117059526971759804426977947952721266880757177055335088777693134693713345640206540670123872210178680306100865355059146219281124303460105424]
cl2 = [148052450029409767056623510365366602228778431569288407577131980435074529632715014971133452626021226944632282479312378667353792117133452069972334169386837227285924011187035671874758901028719505163887789382835770664218045743465222788859258272826217869877607314144, 1643631850318055151946938381389671039738824953272816402371095118047179758846703070931850238668262625444826564833452294807110544441537830199752050040697440948146092723713661125309994275256, 10949587016016795940445976198460149258144635366996455598605244743540728764635947061037779912661207322820180541114179612916018317600403816027703391110922112311910900034442340387304006761589708943814396303183085858356961537279163175384848010568152485779372842]
pq1 = 85198615386075607567070020969981777827671873654631200472078241980737834438897900146248840279191139156416537108399682874370629888207334506237040017838313558911275073904148451540255705818477581182866269413018263079858680221647341680762989080418039972704759003343616652475438155806858735982352930771244880990190318526933267455248913782297991685041187565140859
pq2 = 106239950213206316301683907545763916336055243955706210944736472425965200103461421781804731678430116333702099777855279469137219165293725500887590280355973107580745212368937514070059991848948031718253804694621821734957604838125210951711527151265000736896607029198
t = 60132176395922896902518845244051065417143507550519860211077965501783315971109433544482411208238485135554065241864956361676878220342500208011089383751225437417049893725546176799417188875972677293680033005399883113531193705353404892141811493415079755456185858889801456386910892239869732805273879281094613329645326287205736614546311143635580051444446576104548
r1,s1,s2 = 498841194617327650445431051685964174399227739376, 376599166921876118994132185660203151983500670896, 187705159843973102963593151204361139335048329243
r2,s3 = 620827881415493136309071302986914844220776856282, 674735360250004315267988424435741132047607535029
```

```
# Pell方程求解部分
# sagemath
# def Pell(D,trynumber = 1000):
#     temp = continued_fraction(sqrt(D))
#     for i in range(trynumber):
#         denom = temp.denominator(i)
#         num = temp.numerator(i)
#         if num^2- D * denom^2 == 1:
#             return num,denom
#     return None,None
# for i in wl:
#     Pell(i)
# (1053719038397743281994860271744931381951301581046446334845066286043501100800113223885172926803288929660024822
622108642003526254073215709794979175642102601574147778599503344766303851524807174099126431147906613710297572104182
822067496462240009190564238288281272874966280,168450500310972930707208583777353845862723614274337696968629340838
437927919365973736431467737825931894403582133125917579196621697175572833671789075169621831768398654909584273631
435199401656488388500129435786860576254154212663214052759293877684501204658628574 7)
# (121723653124334943327337351369224143389428692536182586690052931548156177466437320964701609590004825981378294358
781446032392886186351422728173975231719924841105480990927174913175897972732532233,192145577664955207928130455866
581888726107094826100821214812182096944865270585580442342368184834160008486307853040151893126315088740920010178
019160080260110503080625399895592926388238200 4)
# (144017632483156253918361742519911736324442911438543723296525703932387325626989471622981748408863140707432849889
671096671391285764256535030625249875414525380273489340477349991866882957630489039799427756852550650142868784358
708347935642391730147703362434621133545 0,252206958168970759162170958566310090125041275900594363936921012504182
260973233311932227300915630320673148892860517454682634466493232953553501013181999429502235721940271891990460451560
462952746399770525857683655016403400233567567833599249351060740176050197 87)


# 中国剩余定理模数不互素部分求解
ml1 = [1053719038397743281994860271744931381951301581046446334845066286043501100800113223885172926803288929660024
822622108642003526254073215709794979175642102601574147778599503344766303851524807174099126431147906613710297572104
104182206749646224000919056423828828127287496628 0,121723653124334943327337351369224143389428692536182586690052931
548156177466437320964701609590004825981378294358781446032392886186351422728173975231719924841105480990927174913175
89797972732532233,144017632483156253918361742519911736324442911438543723296525703932387325626989471622981748408863
631407074328498896710966713912857642565350306252498754145253802734893404773499918668829576304890397994277568525506
501428687843547083479356423917301477033624346211335450]
ml2 = [168450500310972930707208583777353845862723614274337696968629340838437927919365973736431467737825931894403582
132312591757919662169717557283367178907516962183176839865490958427363614351994016564883885001294357868605762541542
126632140527595293877684501204658628574 7,192145577664955207928130455866581888726107094826100821214812182096948
652705855804423423681848341600084863078530401518931263150887409200101780191600802601105030806253998955929263882382
004,252206958168970759162170958566310090125041275900594363936921012504182260973233311932227300915630320673148892
860517454682634466493232955355350101318199942950223572194027189199046045156046295274639977052585768365501640340
023356756783359924935106074017605019787]
cl1 = [285258922377992879626654060042167879088906728491168257892421618605259039359564532216156338661551247525672638
436509171103444968279126899462375893775287475091820096188899708247710081102572189872078366686862349824621967722
110622766089551905863196505579079091302077607 04, 211158499061801396563106646074584256376705200819832482589841660262
228987535050089041366888200757204110041582641386597621018735885836864733889517447339367697326172796497970851520
57880233721961, 301899179092185964785847705166950181255677272294377823045011205035318463496682788289651177635341894
308537787449148199583490117059526971759804426977947952721266880757177055335088777693134693713345640206540670123
872210178680306100865355059146219281124303460105424]
cl2 = [148052450029409767056623510365366602228778431569288407577131980435074529632715014971133452626021226944632282
479312378667353792117133452069972334169386837227285924011187035671874758901028719505163887789382835770664218045
743465222788859258272826217869877607314144, 164363185031805515194693838138967103973882495327281640237109511804717
975884670307093185023866826262544482656483345229480711054444153783019975205004069744094814609272371366112530999427
5256, 109495870160167959404459761984601492581446353669964555986052447435407287646359470610377799126612073228201
805411141796129160183176004038160277033911109221123119109000344423403873040067615897089438143963031830858583569615
372791631753848480105681524857793728421
```

```
i = 0
def CRT(cl,ml):
```

```
    global i
    clist = cl
    moudlelist = ml
    if i == 0:
        m1,m2 = ml[i],ml[i+1]
        c1,c2 = cl[i],cl[i+1]
    else:
        m1,m2 = ml[i],ml[len(ml)-1]
        c1,c2 = cl[i],cl[len(cl)-1]
    d = gmpy2.gcd(m1,m2)
    print(d)
    c = c2 - c1
    assert c % d == 0
    # X = c // d * gmpy2.invert(m1 // d,m2 // d)
    _,k1,k2 = gmpy2.gcdext(m1,m2)
    X = c // d * k1 * m1
    if i == 0:
        clist.append((X + c1) % gmpy2.lcm(m1,m2))
        # moudlelist.append(m1*m2 // d)
        moudlelist.append(gmpy2.lcm(m1,m2))
        i += 2
    else:
        clist[len(clist)-1] = (X + c1) % gmpy2.lcm(m1,m2)
        # moudlelist[len(moudlelist)-1] = m1*m2 // d
        moudlelist[len(moudlelist)-1] = gmpy2.lcm(m1,m2)
        i += 1
    return clist,moudlelist


ii = 3
temp_cl,temp_ml = CRT(cl1,ml1)
while len(cl1) - ii > 0:
    temp_cl,temp_ml = CRT(temp_cl,temp_ml)
    ii += 1
final_c = temp_cl[len(temp_cl)-1]
m1 = long_to_bytes(gmpy2.iroot(final_c,7)[0])
print(long_to_bytes(gmpy2.iroot(final_c,7)[0]))
ii = 3
i = 0
temp_cl,temp_ml = CRT(cl2,ml2)
while len(cl2) - ii > 0:
    temp_cl,temp_ml = CRT(temp_cl,temp_ml)
    ii += 1
final_c = temp_cl[len(temp_cl)-1]
m2 = long_to_bytes(gmpy2.iroot(final_c,7)[0])
print(long_to_bytes(gmpy2.iroot(final_c,7)[0]))


# DES算法求解部分
hm1 = bytes_to_long(SHA.new(m1).digest())
hm2 = bytes_to_long(SHA.new(m2).digest())

# sagemath
# temp = pq1 * pq2
# p = var('p')
# solve([p*(p-1)==temp],p)
p = 95139353880772104939870618145448234251031105153406565833029787299040378395002190438381537974853777890692924407167823818980082672873538133127131356810153012924025270883966172420658777903337576027105954119811495411149092960422055445121097259802686960288258399754185484307350305454788837702363971523085335074839
```

```
q = pq1 // p
re_s12 = gmpy2.invert(s1-s2,q)
k = (hm1 - hm2) * re_s12 % q

re_r1 = gmpy2.invert(r1,q)
x1 = re_r1 * (k*s1 - hm1) % q
re_r2 = gmpy2.invert(r2,q)
x2 = re_r2 * (k*s3 - hm1) % q
print(bytes.decode(long_to_bytes(x1)),end="")
print(bytes.decode(long_to_bytes(x2)))
```

参考文章

2021年西湖论剑网络安全技能大赛部分WP - 安全客，安全资讯平台 (anquanke.com)