BUUCTF 每日打卡 2021-7-15

原创

Σ2333! 于 2021-07-15 17:04:34 发布 49 收藏

分类专栏： crypto 文章标签： 加密解密 密码学

本文链接：https://blog.csdn.net/weixin_52446095/article/details/118751209

引言

无

[MRCTF2020]Easy_RSA

加密代码如下：

```
import sympy
from gmpy2 import gcd, invert
from random import randint
from Crypto.Util.number import getPrime, isPrime, getRandomNBitInteger, bytes_to_long, long_to_bytes
import base64

from zlib import *
flag = b"MRCTF{XXXX}"
base = 65537

def gen_prime(N):
    A = 0
    while 1:
        A = getPrime(N)
        if A % 8 == 5:
            break
    return A

def gen_p():
    p = getPrime(1024)
    q = getPrime(1024)
    assert (p < q)
    n = p * q
    print("P_n = ", n)
    F_n = (p - 1) * (q - 1)
    print("P_F_n = ", F_n)
    factor2 = 2021 * p + 2020 * q
    if factor2 < 0:
        factor2 = (-1) * factor2
    return sympy.nextprime(factor2)



def gen_q():
```

```python
def gen_q():
    p = getPrime(1024)
    q = getPrime(1024)
    assert (p < q)
    n = p * q
    print("Q_n = ", n)
    e = getRandomNBitInteger(53)
    F_n = (p - 1) * (q - 1)
    while gcd(e, F_n) != 1:
        e = getRandomNBitInteger(53)
    d = invert(e, F_n)
    print("Q_E_D = ", e * d)
    factor2 = 2021 * p - 2020 * q
    if factor2 < 0:
        factor2 = (-1) * factor2
    return sympy.nextprime(factor2)


if __name__ == "__main__":
    _E = base
    _P = gen_p()
    _Q = gen_q()
    assert (gcd(_E, (_P - 1) * (_Q - 1)) == 1)
    _M = bytes_to_long(flag)
    _C = pow(_M, _E, _P * _Q)
    print("Ciphertext = ", _C)
'''
P_n =  14057332139537395701238463644827948204030576528558543283405966933509944444681257521108769303999679955371474546213196051386802936343092965202519504111238572269823072199039812208100301939365080328518578704076769147484922508482686658959347725753762078590928561862163337382463252361958145933210306431342748775024336556028267742021320891681762543660468484018686865891073110757394154024833552558863671537491089957038648328973790692356014778420333896705595252711514117478072828880198506187667924020260600124717243067420876363980538994101929437978668709128652587073901337310278665778299513763593234951137512120572797739181693
P_F_n =  14057332139537395701238463644827948204030576528558543283405966933509944444681257521108769303999679955371474546213196051386802936343092965202519504111238572269823072199039812208100301939365080328518578704076769147484922508482686658959347725753762078590928561862163337382463252361958145933210306431342748775024099427363967321110127562039879018616082926935567951378185280882426903064598376668106616694623540074057210432790309571018778281723710994930151635857933293394780142192586806292968028305922173313521186946635709194350912242693822450297748434301924950358561859804256788098033426537956252964976682327991427626735740
Q_n =  20714298338160449749545360743688018842877274054540852096459485283936802341271363766157976112525034004319938054034934880860956966585051684483662535780621673316774842614701726445870630109196016676725183412879870463432277629916669130494040403733295593655306104176367902352484367520262917943100467697540593925707162162616635533550262718808746254599456286578409187895171015796991910123804529825519519278388910483133813330902530160448972926096083990208243274548561238253002789474920730760001104048093295680593033327818821255300893423412192265814418546134015557579236219461780344469127987669565138930308525189944897421753947
Q_E_D =  10077207922229813458611615685074281785540812771696289192925986874667257260233391895807558267175249361825951828633612277270333018303722110505829865349079433788509849907358382183253279830951353838317523342953346734839038932322519880529495048480206814859090290722115096853906798043283131037636820277321226632011267069973750105483164628658514228141923757222271397564684355502473185568857383410871187440614954007825377434970815806305575493281267578612370076828804844532619988098371750453882549810378930487368219105305036680682580260265867426844084457795549936840401911491393447716042842866284701228951665531068011963860031522828429893520 1
Ciphertext =  40855937355228438525361161524441274634175356845950884889338630813182607485910094677909779126550263041947960009043847754950009434240703963344358101265361653325654173367970366117733827283446871752530810475866028386850274282926215579145146290243247942757725220131264649269906201404064129994857287503858768681150917354255775550273940334166430326447743396446540116867166397605123533557190657952222011672198317809613082257804784824672944108285434884122587644464948152387661857284544166918988594625320834372137931048237591473176136378814197875819207451514303945267127906084429601065375391218805142698306963417375077174489469620 21
'''
```

事实上，这种题目吓唬人的成分很大，只要仔细看就能解出来
比如给的第一个函数```gen_prime````完全没有用到
后面的P_n和Q_n也可以爆破出来，然后照着他的加密方法解出P和Q，P_F_n和Q_E_D完全没有用到
最后就是常规的RSA了
解密代码如下：

```
from Crypto.Util.number import *
import sympy

P_n =  14057332139537395701238463644827948204030576528558543283405966933509944444681257521108769303999679955371474546213196051386802936343092965202519504111238572269823072199039812208100301939365080328518578704076769147484925084826866589593477257537620785909285618621633373824632523619581459332103064313427487750243365560282677420213208916817625436604684840186868658910731107573941540248335525588636715374910899570386483289737906923560147784203338967055952527115141174780728288801985061876679240202606001247172430674208763639805389941019294379786687091286525870739013373102786657782995137635932349511375121205727977391816931
P_F_n =  14057332139537395701238463644827948204030576528558543283405966933509944444681257521108769303999679955371474546213196051386802936343092965202519504111238572269823072199039812208100301939365080328518578704076769147484922508482686658959347725753762078590928561862163337382463252361958145933210306431342748775024336556028267742021320275620398790186160829269355679513781852808824269030645983766681066166946235400740572104327903095710187782817237109949301516358579332933947801421925868062929680283059221733135211869466357091943509122426938224502977484343019249503585618598042567880980334265379562529649766823279914276267357401
Q_n =  20714298338160449749545360743688018842877274054540852096459485283936802341271363766157976112525034004319938054034934880860956966585051684483662535780621673316774842614701726445870630109196016676725183412879870463432277629916669130494040403733295593655306104176367902352484367520262917943100467697540593925707162162616635533550262718808746254599456286578409187895171015796991910123804529825519519278388910483133813330902530160448972926096083990208243274548561238253002789474920730760001104048093295680593033327818821255300893423412192265814418546134015557579236219461780344469127987669565138930308525189944897421753947
Q_E_D =  10077207922229813458611615685074281785540812771696289192925986874667257260233391895807558267175249361825951828633612277270333018303722110505829865349079433788509849907358382183253279830951353838317523342953346734839038932322519880529495048480206814859090290722115096853906798043283131037636820277321226632011267069973750105483164628658514228141923757222271397564684355502473185568857383410871187440614954007825377434970815806305575493281267578612370076828804844532619988098371750453882549810378930487368219105305036680682580260265867426844084457795549368404019114913934477160428428662847012289516655310680119638600315228284298935201
Ciphertext =  40855937355228438525361161524441274634175356845950884889338630813182607485910094677909779126550263041947960009043847754950009434240703963344358101265361653325654173367970366117733827283446871752530810475866028386850274282926215579145146290243247942757725220131264649269906201404064129994857287503858768681150917354255775550273940334166430326447743396446540116867166397605123533557190657952222011672198317809613082257804784824672944108285434884122587644464948152387661857284544166918988594625320834372137931048237591473176136378814197875819207451514303945267127906084429601065375391218805142698306963417375077174489469620215E = 65537

# P
P_p = 11815357834556225055076705773138578296306373458632111257986974765000144847363386030514228150486252192824652087630070740551514144472755083906683519590592728190388030786094263032249910616419173617420150645715727222080251560793961847671659388842883296237449414772357798099266162925471311692369006782715566888957 1
P_q = 11897508595485866064256258415213926142249334853259340030796012731724951176154203045191256136268736105319137530718041393172135525189535093637678165767489680138880637975075726437739660817423507502185461432800989740882423580016736920420368093829880375296498335829829969927342559638226886923713972475421444355638 3
P = sympy.nextprime(2021 * P_p + 2020 * P_q)
print(P)

# Q
Q_p = 12053884951466197015985585154757763771190036873246295377473848348075995086724486724040127386498498138580645373565596779732976925214312596696623676739199556341824374830268534833664287230604228640142758150160971357732994576093039513041174332259502628785307331015010353587307843689603594338506789306269885897629 1
Q_q = 17184748669465960870633692317378670807160368997294228976066969000261552526353448326147769954048261552022330078077817212022100841751859013375370114559194384055280207247429355660838967780641539238492491391167728812606624502573141639965685562583928875232626774197943685544126017730570752945671562506208089232701 7
Q = 2021 * Q_p - 2020 * Q_q
```

```
if Q < 0:
    Q = (-1) * Q
Q = sympy.nextprime(Q)
print(Q)

# flag
D = inverse(_E, (P-1)*(Q-1))
M = pow(Ciphertext, D, P*Q)
print(long_to_bytes(M))
```

结果为：

```
b'MRCTF{Ju3t_@_31mp13_que3t10n}'
```

[ACTF新生赛2020]crypto-aes

加密代码如下：

```
from Cryptodome.Cipher import AES
import os
import gmpy2
from flag import FLAG
from Cryptodome.Util.number import *

def main():
    key=os.urandom(2)*16
    iv=os.urandom(16)
    print(bytes_to_long(key)^bytes_to_long(iv))
    aes=AES.new(key,AES.MODE_CBC,iv)
    enc_flag = aes.encrypt(FLAG)
    print(enc_flag)
if __name__=="__main__":
    main()
```

AES基本没有了解过，以前尝试看了加密函数的[官方文档]，没怎么看明白
(https://pycryptodome.readthedocs.io/en/latest/src/cipher/aes.html)
只能找wp
如果只是解题的话，只需要利用

```
aes = AES.new(key,AES.MODE_CBC,iv)
flag = aes.decrypt(enc_flag)
```

两句代码即可，关键是解出**key**（密钥）和**iv**（初始化向量）
可以看到key是由8个重复的随机生成的2bytes,16bits的字符串组成32bytes,256bits字符串，iv则是随机生成16bytes,128bits的字符串
已知key与iv做异或运算得到的结果，所以key的前16bytes字符是不变的，由此可以推断出key
再用key后16bytes与异或的结果进行异或运算可以得到iv
解密代码如下：

```
from Crypto.Util.number import *
from Crypto.Cipher import AES

xor = 91144196586662942563895769614300232343026691029427747065707381728622849079757
enc_flag = b'\x8c-\xcd\xde\xa7\xe9\x7f.b\x8aKs\xf1\xba\xc75\xc4d\x13\x07\xac\xa4&\xd6\x91\xfe\xf3\x14\x10|\xf8p'

key = long_to_bytes(xor)[:16] * 2
print(key)
iv = bytes_to_long(long_to_bytes(xor)[16:]) ^ bytes_to_long(key[16:])
iv = long_to_bytes(iv)
aes = AES.new(key, AES.MODE_CBC, iv)
flag = aes.decrypt(enc_flag)
print(flag)
```

结果为：

```
b'actf{W0W_y0u_can_so1v3_AES_now!}'
```

结语：

希望继续坚持