Crypto日记之picoCTF 2022中的RSA题目求解

原创

Sm0ry 于 2022-03-24 15:11:03 发布 65 收藏 1

分类专栏： Crypto日记 文章标签： python rsa 加密解密 算法 安全

本文链接：https://blog.csdn.net/zgzhzywzd/article/details/123577222

前言

picoCTF 2022一共66道题，队友基本都解出来了，只能说师傅们tql！共有15道Cryptography题目，其中3道RSA题目，难点应该都在整数分解的算法，借Generate师傅的wp分析一波，靠谱！

题解

0x01 Very Smooth

Description：Forget safe primes… Here, we like to live life dangerously… >:)

看下代码：

```
#!/usr/bin/python

from binascii import hexlify
from gmpy2 import *
import math
import os
import sys

if sys.version_info < (3, 9):
    math.gcd = gcd
    math.lcm = lcm

_DEBUG = False

FLAG  = open('flag.txt').read().strip()
FLAG  = mpz(hexlify(FLAG.encode()), 16)
SEED  = mpz(hexlify(os.urandom(32)).decode(), 16)
STATE = random_state(SEED)

def get_prime(state, bits):
    return next_prime(mpz_urandomb(state, bits) | (1 << (bits - 1)))

def get_smooth_prime(state, bits, smoothness=16):
    p = mpz(2)
    p_factors = [p]
    while p.bit_length() < bits - 2 * smoothness:
        factor = get_prime(state, smoothness)
        p_factors.append(factor)
```

```
        p *= factor

    bitcnt = (bits - p.bit_length()) // 2
    while True:
        prime1 = get_prime(state, bitcnt)
        prime2 = get_prime(state, bitcnt)
        tmpp = p * prime1 * prime2
        if tmpp.bit_length() < bits:
            bitcnt += 1
            continue
        if tmpp.bit_length() > bits:
            bitcnt -= 1
            continue
        if is_prime(tmpp + 1):
            p_factors.append(prime1)
            p_factors.append(prime2)
            p = tmpp + 1
            break
    p_factors.sort()
    return (p, p_factors)

e = 0x10001

while True:
    p, p_factors = get_smooth_prime(STATE, 1024, 16)
    if len(p_factors) != len(set(p_factors)):
        continue
    # Smoothness should be different or some might encounter issues.
    q, q_factors = get_smooth_prime(STATE, 1024, 17)
    if len(q_factors) != len(set(q_factors)):
        continue
    factors = p_factors + q_factors
    if e not in factors:
        break

if _DEBUG:
    import sys
    sys.stderr.write(f'p = {p.digits(16)}\n\n')
    sys.stderr.write(f'p_factors = [\n')
    for factor in p_factors:
        sys.stderr.write(f'    {factor.digits(16)},\n')
    sys.stderr.write(f']\n\n')

    sys.stderr.write(f'q = {q.digits(16)}\n\n')
    sys.stderr.write(f'q_factors = [\n')
    for factor in q_factors:
        sys.stderr.write(f'    {factor.digits(16)},\n')
    sys.stderr.write(f']\n\n')

n = p * q

m = math.lcm(p - 1, q - 1)
d = pow(e, -1, m)
c = pow(FLAG, e, n)

print(f'n = {n.digits(16)}')
print(f'c = {c.digits(16)}')

#n = e77c4035292375af4c45536b3b35c201daa5db099f90af0e87fedc480450873715cffd53fc8fe5db9ac9960867bd9881e2f0931ffe0
```

```
cea4399b26107cc6d8d36ab1564c8b95775487100310f11c13c85234709644a1d8616768abe46a8909c932bc548e23c70ffc0091e2ed9a12
0fe549583b74d7263d94629346051154dad56f2693ad6e101be0e9644a84467121dab1b204dbf21fa39c9bd8583af4e5b7ebd9e02c862c43
a426e0750242c30547be70115337ce86990f891f2ad3228feea9e3dcd1266950fa8861411981ce2eebb2901e428cfe81e87e415758bf245f
66002c61060b2e1860382b2e6b5d7af0b4a350f0920e6d514eb9eac7f24a933c64a89
#c = 671028df2e2d255962dd8685d711e815cbea334115c30ea2005cf193a1b972e275c163de8cfb3d0145a453fec0b837802244ccde0fa
f832dc3422f56d6a384fbcb3bfd969188d6cd4e1ca5b8bc48beca966f309f52ff3fc3153cccaec90d8477fd24dfedc3d4ae492769a6afefb
bf50108594f18963ab06ba82e955cafc54a978dd08971c6bf735b347ac92e50fe8e209c65f946f96bd0f0c909f34e90d67a4d12ebe61743b
438ccdbcfdf3a566071ea495daf77e7650f73a7f4509b64b9af2dd8a9e33b6bd863b889a69f903ffef425ea52ba1a293645cbac48875c422
20ec0b37051ecc91daaf492abe0aaaf561ffb0c2b093dcdabd7863b1929f0411891f5
```

代码很长，输出结果给了n和c，已知e=65537，利用factordb无法直接分解，根据题目提示Don't look at me… Go ask Mr. Pollard if you need a hint!以及题目Very Smooth可以联想到Pollard's p−1光滑数。
即当p是N的因数，且p−1光滑的时候，可以使用 Pollard's p − 1 算法来分解 N。
NCTF2019-官方writeup中childRSA这题对这个算法讲的很清晰。
具体的python代码如下：

```python
def Pollard(n):
    a=2
    while True:
        for i in range(2,80000):
            a=pow(a,i,n)
        for j in range(80000,104729+1):
            a=pow(a,j,n)
            if j % 15 ==0:
                d=GCD(a-1,n)
                if(1<d<n):
                    return(d)
        a+=1
```

n成功分解后就是常规的RSA题了，一共跑了24秒，还是很快的，完整代码如下：

```
from Crypto.Util.number import *
import gmpy2
n = 'e77c4035292375af4c45536b3b35c201daa5db099f90af0e87fedc480450873715cffd53fc8fe5db9ac9960867bd9881e2f0931ffe0
cea4399b26107cc6d8d36ab1564c8b95775487100310f11c13c85234709644a1d8616768abe46a8909c932bc548e23c70ffc0091e2ed9a12
0fe549583b74d7263d94629346051154dad56f2693ad6e101be0e9644a84467121dab1b204dbf21fa39c9bd8583af4e5b7ebd9e02c862c43
a426e0750242c30547be70115337ce86990f891f2ad3228feea9e3dcd1266950fa8861411981ce2eebb2901e428cfe81e87e415758bf245f
66002c61060b2e1860382b2e6b5d7af0b4a350f0920e6d514eb9eac7f24a933c64a89'
c = '671028df2e2d255962dd8685d711e815cbea334115c30ea2005cf193a1b972e275c163de8cfb3d0145a453fec0b837802244ccde0fa
f832dc3422f56d6a384fbcb3bfd969188d6cd4e1ca5b8bc48beca966f309f52ff3fc3153cccaec90d8477fd24dfedc3d4ae492769a6afefb
bf50108594f18963ab06ba82e955cafc54a978dd08971c6bf735b347ac92e50fe8e209c65f946f96bd0f0c909f34e90d67a4d12ebe61743b
438ccdbcfdf3a566071ea495daf77e7650f73a7f4509b64b9af2dd8a9e33b6bd863b889a69f903ffef425ea52ba1a293645cbac48875c422
20ec0b37051ecc91daaf492abe0aaaf561ffb0c2b093dcdabd7863b1929f0411891f5'

def Pollard(n):
    a=2
    while True:
        for i in range(2,80000):
            a=pow(a,i,n)
        for j in range(80000,104729+1):
            a=pow(a,j,n)
            if j % 15 ==0:
                d=GCD(a-1,n)
                if(1<d<n):
                    return(d)
        a+=1
p=Pollard(int(n,16))
q=int(n,16)//p
d=gmpy2.invert(65537,(p-1)*(q-1))
m=pow(int(c,16),d,int(n,16))
print(long_to_bytes(m))
#picoCTF{94287e17}
```

0x02 Sum-O-Primes

Description：We have so much faith in RSA we give you not just the product of the primes, but their sum as well!

先看代码：

```python
#!/usr/bin/python

from binascii import hexlify
from gmpy2 import mpz_urandomb, next_prime, random_state
import math
import os
import sys

if sys.version_info < (3, 9):
    import gmpy2
    math.gcd = gmpy2.gcd
    math.lcm = gmpy2.lcm

FLAG  = open('flag.txt').read().strip()
FLAG  = int(hexlify(FLAG.encode()), 16)
SEED  = int(hexlify(os.urandom(32)).decode(), 16)
STATE = random_state(SEED)

def get_prime(bits):
    return next_prime(mpz_urandomb(STATE, bits) | (1 << (bits - 1)))

p = get_prime(1024)
q = get_prime(1024)

x = p + q
n = p * q

e = 65537
m = math.lcm(p - 1, q - 1)
d = pow(e, -1, m)
c = pow(FLAG, e, n)
print(f'x = {x:x}')
print(f'n = {n:x}')
print(f'c = {c:x}')
```

题目给了x = p + q和n = p * q，已知phi = (p-1) * (q-1) = p * q - (p + q) +1直接就可以直接得到phi，再解rsa就可以了，完整代码：

```python
from Crypto.Util.number import *
import gmpy2
x = '1b1fb4b96231fe1b723d008d0e7776169ee5d4a8e3573c12c37721cee5de1d882f040d1e3f543d36a574984ad95c1e79e02de14fa13
6b4be7f4468cbd62773f6a4fd06effc2b845ca07424100466bdfeee652d78b25a4273ba4e950e1a8ebfe256a2f8541fe2207c41f39c2363e
23064bc56bed5cf563b8dba873da3c1320256e'
n ='b6b2353316c7b0a6c0ecae3bd7d2191eee519551f4ed86054e6380663668e595f6f43f867caa8feda217905643d73453f3797f6096c9
89fd099852239e5d73c753f909d8efd172d211a4ed4a966dbcbf56b9cbadd416de0a3472a253571b4e4f1bab847a407a27eb37449488f63a
edb9f5ec72d9e331ab6154fe45c8cb4e2005d124d1ac8ecd588cd2280e215b078d8ea9da438bbcb1b155a339b91f39e3d17bab112436cdbb
6d104fdeb0dce1ac41a1fe8fda0490ef3124794e0383565c299df24ad8a915669469c0b0dc604ed359afb3636d5f633362d8ef9fce7a42f6
4d5f1f4e50911a15459f97c1b11ee44af4e8bb636895cf75da105a8d1564160ba091'
c = '49e426aba3431d9bb73bfc5dd18115dcea3c78a9915e9cf65e060560015c951327f20fe5dd74bfecd9a00659d4f740e42f707e47d8f
6b331d8ad1021de41e15f133cbe7c782f22168149df57a6c37095ba6877765a67d8478434a7a5eabb26097404ad464fa0388cacb97a26aaf
3b83b6eb0fa73e16bc1de49b33ee64920118f8483feff3634541df97dadad88302392095059cbe56e7148453f16464da8be2b6ca4a6fc005
2210f697975fd3c4f3f94bfa3bb2422124a6f0e9685f0440ed020294b6788d7ea3c002d86d86faced8e37b36673ea2b5c72726c66d1834d2
dcafdf40220c41dfb3d1f07c5c0d236ce7af86b937476c5aabe33cae8d535713627de'
phi=int(n,16)-int(x,16)+1
d=gmpy2.invert(65537,phi)
m=pow(int(c,16),d,int(n,16))
print(long_to_bytes(m))
#picoCTF{126a94ab}
```

0x03 NSA Backdoor

Description：I heard someone has been sneakily installing backdoors in open-source implementations of Diffie-Hellman… I wonder who it could be…

代码如下：

```
#!/usr/bin/python

from binascii import hexlify
from gmpy2 import *
import math
import os
import sys

if sys.version_info < (3, 9):
    math.gcd = gcd
    math.lcm = lcm

_DEBUG = False

FLAG  = open('flag.txt').read().strip()
FLAG  = mpz(hexlify(FLAG.encode()), 16)
SEED  = mpz(hexlify(os.urandom(32)).decode(), 16)
STATE = random_state(SEED)

def get_prime(state, bits):
    return next_prime(mpz_urandomb(state, bits) | (1 << (bits - 1)))

def get_smooth_prime(state, bits, smoothness=16):
    p = mpz(2)
    p_factors = [p]
    while p.bit_length() < bits - 2 * smoothness:
        factor = get_prime(state, smoothness)
        p_factors.append(factor)
        p *= factor

    bitcnt = (bits - p.bit_length()) // 2

    while True:
        prime1 = get_prime(state, bitcnt)
        prime2 = get_prime(state, bitcnt)
        tmpp = p * prime1 * prime2
        if tmpp.bit_length() < bits:
            bitcnt += 1
            continue
        if tmpp.bit_length() > bits:
            bitcnt -= 1
            continue
        if is_prime(tmpp + 1):
            p_factors.append(prime1)
            p_factors.append(prime2)
            p = tmpp + 1
            break

    p_factors.sort()

    return (p, p_factors)

while True:
    p, p_factors = get_smooth_prime(STATE, 1024, 16)
    if len(p_factors) != len(set(p_factors)):
```

```
        continue
    # Smoothness should be different or some might encounter issues.
    q, q_factors = get_smooth_prime(STATE, 1024, 17)
    if len(q_factors) == len(set(q_factors)):
        factors = p_factors + q_factors
        break

if _DEBUG:
    import sys
    sys.stderr.write(f'p = {p.digits(16)}\n\n')
    sys.stderr.write(f'p_factors = [\n')
    for factor in p_factors:
        sys.stderr.write(f'    {factor.digits(16)},\n')
    sys.stderr.write(f']\n\n')

    sys.stderr.write(f'q = {q.digits(16)}\n\n')
    sys.stderr.write(f'q_factors = [\n')
    for factor in q_factors:
        sys.stderr.write(f'    {factor.digits(16)},\n')
    sys.stderr.write(f']\n\n')

n = p * q
c = pow(3, FLAG, n)

print(f'n = {n.digits(16)}')
print(f'c = {c.digits(16)}')
```

观察代码里n的生成方式发现和Very Smooth的一样，使用相同的方法分解n：

```
from Crypto.Util.number import *
import gmpy2
n = '5bf9961e4bcfc88017e1a9a40958af5eae3b3ee3dcf25bce02e5d04858ba1754e13e86b78a098ea0025222336df6b692e14533dad7f
478005b421d3287676843f9f49ffd7ebec1e8e43b96cde7cd28bd6fdf5747a4a075b5afa7da7a4e9a2ccb26342799965f3fb6e65e0bb9557
c6f3a67568ccbfaaa7e3d6c5cb79dd2f9928111c3183bf58bd91412a0742bbfb3c5cebfb0b82825da0875c5ee3df208ce563f896d67287c8
b9aad9943dd76e5eae1fc8abd473ec9f9e4f2b49b7897954ca77b8f00ed51949c7e4f1f09bd54b830058bd7f4da04e5228250ba062ec0e1d
19fb48a05333aada60ecdfc8c62c15773ed7e077edba71621f6a6c10302cc9ed26ec9'
c = '2475123653f5a4b842e7ac76829e896450126f7175520929a35b6a4302788ceff1a605ed30f4d01c19226e09fc95d005c61320d3bbd
55cfebbc775332067ac6056c1969282091856eaa44ccaf5738ac6409e865bbd1186d69f718abd2b3a1dd3dc933a07ca687f0af9385406fd9
ee4fa5f701ad46f0852bf4370264c21f775f1e15283444b3bf45af29b84bb429ed5a17adc9af78aee8c5351434491d5daf9dd3ce3cf0cd44
b307eb403f0e9f482dd001b25ed284c4e6c1ba2864e5a2c4b1afe4161426cc67203f30553c88d7132aef1337eca00622b47cb7a28195f0e3
a2ab934e6163b2941a4631412e13b1a72fe34e6480fada9af4dae14f2608805d61ee'
def Pollard(n):
    a=2
    while True:
        for i in range(2,80000):
            a=pow(a,i,n)
        for j in range(80000,104729+1):
            a=pow(a,j,n)
            if j % 15 ==0:
                d=GCD(a-1,n)
                if(1<d<n):
                    return(d)
        a+=1
p=Pollard(int(n,16))
print(p)
```

后部分涉及到求解离散对数，我们可以拆分成modp，modq两部分从而简化，然后使用中国剩余定理将它们合并即可，sage脚本如下：

```
from Crypto.Util.number import *
p=112702077491326624035437448311528244416633038267184436467539953783623022543629307291975209668348933325006075339780165463077524233511267597550006727923822554354936896793276829740193900248027487979522143806746878229772394053610558597354876381637035909859704552979985236170415302488615161107293296362528480525723
n='5bf9961e4bcfc88017e1a9a40958af5eae3b3ee3dcf25bce02e5d04858ba1754e13e86b78a098ea0025222336df6b692e14533dad7f478005b421d3287676843f9f49ffd7ebec1e8e43b96cde7cd28bd6fdf5747a4a075b5afa7da7a4e9a2ccb26342799965f3fb6e65e0bb9557c6f3a67568ccbfaaa7e3d6c5cb79dd2f9928111c3183bf58bd91412a0742bbfb3c5cebfb0b82825da0875c5ee3df208ce563f896d67287c8b9aad9943dd76e5eae1fc8abd473ec9f9e4f2b49b7897954ca77b8f00ed51949c7e4f1f09bd54b830058bd7f4da04e5228250ba062ec0e1d19fb48a05333aada60ecdfc8c62c15773ed7e077edba71621f6a6c10302cc9ed26ec9'
q=int(n,16)//p
c='2475123653f5a4b842e7ac76829e896450126f7175520929a35b6a4302788ceff1a605ed30f4d01c19226e09fc95d005c61320d3bbd55cfebbc775332067ac6056c1969282091856eaa44ccaf5738ac6409e865bbd1186d69f718abd2b3a1dd3dc933a07ca687f0af9385406fd9ee4fa5f701ad46f0852bf4370264c21f775f1e15283444b3bf45af29b84bb429ed5a17adc9af78aee8c5351434491d5daf9dd3ce3cf0cd44b307eb403f0e9f482dd001b25ed284c4e6c1ba2864e5a2c4b1afe4161426cc67203f30553c88d7132aef1337eca00622b47cb7a28195f0e3a2ab934e6163b2941a4631412e13b1a72fe34e6480fada9af4dae14f2608805d61ee'
G1=GF(p)
G2=GF(q)
c1=G1(int(c,16))
c2=G2(int(c,16))
g1=G1(3)
g2=G2(3)
k1=discrete_log(c1,g1)
k2=discrete_log(c2,g2)
m=crt([k1,k2],[p-1,q-1])
print(long_to_bytes(m))
#picoCTF{cf58a7b8}
```