密码学原理

实验三：公钥加密

学号：2022113564　　　姓名：张哲恺

实验目的：本实验旨在让学生掌握运用密码学工具生成 RSA 密钥，进行非对称加解密，并分析相同因子 RSA 公钥理解 RSA 的安全性。

1、 公钥加密

(1) 使用密码学工具实现混合加密过程

要求：生成 RSA 密钥对（公钥 1 和私钥 1），用公钥 1 加密对称密钥并采用对称加密方法加密图片，用私钥 1 解密对称密钥，然后解密图片。（你可以使用实验二中的对称加密方案）

调用 rsa.generate_private（）生成私钥，设定指数为 65537，密钥长度为 2048，后端选择默认后端。再通过私钥 private_key.public_key()生成公钥，将私钥和公钥分别保存在 `rsa_private_key.pem` 和 `rsa_public_key.pem`.

Key_genenrate.py:

```
from cryptography.hazmat.primitives.ciphers.aead import AESGCM
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import padding
import numpy as np
from PIL import Image
import os
import hmac

# 生成 RSA 密钥对
private_key = rsa.generate_private_key(
    public_exponent=65537,
    key_size=2048,
```

```python
    backend=default_backend()
)

# 获取公钥
public_key = private_key.public_key()

# 将私钥保存为.pem 文件
pem_data_private = private_key.private_bytes(
    encoding=serialization.Encoding.PEM,
    format=serialization.PrivateFormat.PKCS8,
    encryption_algorithm=serialization.NoEncryption()
)

with open("C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB3\\rsa_private_key.pem", "wb") as pem_file_private:
    pem_file_private.write(pem_data_private)

print("Private key saved as rsa_private_key.pem")

# 将公钥保存为.pem 文件
pem_data_public = public_key.public_bytes(
    encoding=serialization.Encoding.PEM,
    format=serialization.PublicFormat.SubjectPublicKeyInfo
)

with open("C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB3\\rsa_public_key.pem", "wb") as pem_file_public:
    pem_file_public.write(pem_data_public)

print("Public key saved as rsa_public_key.pem")
```

图片加密过程和实验二一样，首先从文件中获取公钥和私钥，加密时使用 AES_GCM 加密方法，并且使用公钥加密对称密钥，解密时先用私钥解密对称密钥再解密图像。

```
key:  b'\xefi\xd7U\x8bp?\xe7\xaa\xcd\xb8\x84\xda\x15)\xd6'
encrypted_key:  b'\x93\xdav\xad\xa6 \xd42\xdbv\xd8\x87\t2\x91\xb1\x1e\x13gVb+=\xb7$\xa4\xed\x0faT:\xaa\xcb\xce,P\xdbV\xfc\xa1\x0
cB\xca\xaf\xd9\xd2\x84\xd1$l\x9b\xc6\xee\xf2\x10\xe8[H\xc2\x8c\xecfJ\xd0\xb1\xe9\x83~\xb9\xf3\xf3\xd1\x99s,e\x96\x9ch\x92+@\xf2\
xfbm\x94\xc5\xb3\xff\x9e\xd4\x8c\x9e\x1c\xe6X#\x1ah\xab\x173\x19\x8e\xf4\x9cN\xf1\xc4\x00\x86\xe7\xfb\xde]T\xde_\x7f\x19\xb6J\xc
7\t\x8e\x15\\\x86\x1c\xbd:\x9a\xef\x8eb x\xe0\x08\x99\xb1\xc0\xdd\xb5:\x1c\xef[\xa0~\xdc@\x87\xdd\xe3\x99i\xf9\x97\x15\xb6\x91\x
f0\xafo^\xe13$1\x94\xa0\x86\xf8R\x81a\xd1\x13S.\x1eG\xcb\xf1\xb6\x1e\x8c(\x98\tC\xa7\xa8E\x18r\x0b\x7fL]\x88\x86\x03\x0bd\xd6\xc
c\xe6\x19D\xa5\x9f\x1c\x8d\xe4\xa1\xf5\xd3(1\x06\x1b\x03\xf7\xcb\xb3\xd2F{^\xb9vVs%\xe8+M\\\xe4\xce\xa1\xd0t\xe1\xeeq\x0c\xa4\x0
6\x1b\x16\xdf\xddq'
decrypted_key:  b'\xefi\xd7U\x8bp?\xe7\xaa\xcd\xb8\x84\xda\x15)\xd6'
验证通过，解密成功
```

RSA.py:

```python
from cryptography.hazmat.primitives.ciphers.aead import AESGCM
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives import serialization
```

```python
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import hashes
from cryptography.hazmat.primitives.asymmetric import padding
import numpy as np
from PIL import Image
import os
import hmac

def encrypt(key) :
    with open("C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB2\\CCA\\input_image.bmp" , 'rb') as f:
        input_image = bytearray(f.read())

    input_image[10] = 98 #修改图片数据起始位置

    plaintext = np.array(input_image[55: ]) #记录图片数据
    #加密过程
    aesgcm = AESGCM(key)
    # print("key: " ,  key)
    nonce = os.urandom(12)
    # print("nonce: " , nonce.hex().upper())
    ciphertext = aesgcm.encrypt(nonce , plaintext.tobytes() , None)
    hmac_value = hmac.new(key, ciphertext, digestmod='sha256').digest()
    # print("hmac_value: " , hmac_value.hex().upper())

    with open("C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB3\\encrypted_image.bmp" , "wb") as f :
        f.write(input_image[:55]) #文件头
        f.write(nonce) #nonce
        f.write(hmac_value) #HMAC 消息
        f.write(ciphertext) #加密图像数据

def decrypt(encrypted_key) :
    key = private_key.decrypt(
        encrypted_key,
        padding=padding.PKCS1v15()
    )
    print("decrypted_key: " , key)
    aesgcm = AESGCM(key)
    with open("C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB3\\encrypted_image.bmp" , "rb") as f :
        bmphead = bytearray(f.read(55)) #获取文件头
        nonce = bytearray(f.read(12)) #nonce
        # print(nonce.hex().upper())
        hmac_v = bytearray(f.read(32)) #hmac
        # print(hmac_v.hex().upper())
        encrypted_image = bytearray(f.read())
    hmac_value = hmac.new(key , encrypted_image , digestmod='sha256').digest()
```

```python
    if (hmac_v == hmac_value) :
        bmphead[10] = 54 #修改图像数据起始位置
        ciphertext = np.array(encrypted_image)
        decrypted_img = aesgcm.decrypt(nonce , ciphertext.tobytes(), None) #解密
        with open("C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB3\\decrypted_image.bmp" , "wb") as f :
            f.write(bmphead)
            f.write(decrypted_img)
        print("验证通过，解密成功")
    else :
        with open("C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB3\\decrypted_image.bmp" , "wb") as f :
            f.write(bmphead)
            f.write(nonce)
            f.write(hmac_v)
            f.write(encrypted_image)
        print("验证失败，解密失败")

key = AESGCM.generate_key(bit_length=128)

#加载公钥
with open("C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB3\\rsa_public_key.pem", "rb") as pem_file_public:
    public_key = serialization.load_pem_public_key(
        pem_file_public.read(),
        backend=default_backend()
    )

# 加载私钥
with open("C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB3\\rsa_private_key.pem", "rb") as pem_file_private:
    private_key = serialization.load_pem_private_key(
        pem_file_private.read(),
        password=None,  # 如果私钥使用了密码，这里需要提供密码
        backend=default_backend()
    )

#加密 AES 密钥
encrypted_key = public_key.encrypt(
    key,
    padding=padding.PKCS1v15()
)
print("key: " , key)
print("encrypted_key: " , encrypted_key)
encrypt(key)
decrypt(encrypted_key)
```

2、 相同因子公钥分析

(1) 通过分析 RSA 公钥的因子得到私钥并破解加密信息

要求：分析附件中给出的两个公钥（公钥 1、公钥 2，均为 pem 格式）中大整数的公共因子，得到公钥 1 的私钥，并用私钥解密对称密钥，再解密对称加密的图片。

① 对称加密方法为密钥长度 128 位的 AES-CBC，对明文采用 PKCS #7 填充，128 位 IV 放在密文开头。

② 对称加密的明文为 RGBA 四通道图像中的所有像素，为使密文图片尺寸合法，对密文进行了填充，以四字节（一个像素）为单位，与 PKCS #7 类似，即如果密文图像最后一个像素转换为四字节整数的值为 k（大端序），说明密文图像的后 k 个像素是 padding。由此填充方法产生的密文图片比明文图片多一行像素。

③ 对称加密产生的密文图片为 enc1.png

④ 128 位对称密钥先进行 Base64 编码，再使用公钥 1 加密，加密方法为 RSA-OAEP，密文的 Base64 编码在下面给出。

公钥 1：


```
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAuz20BUTcqVDjzEOKiJF9
66LbQB/59lnXTj/SmiD07mV1XE03BLrWfi7jFh/iq5ZPzVXfbNPjHiojO9WRhWzr
wiQGZNVZ7qFoO/PzXOT8OyHyOMcrb6ogtCyFvDOeximr3M/ICmliU2JxbLSfteZj
AplHJVgs5bJ5LTW7eSy1x2Z5aOsHjesK3rkLi1yB2jM0MeaNIB/Enb82bBMKzAam
vN6tY8bQbEoRbTnlX6PUfkU9w7XsWLMa3QbpIH9mNam1Qz4ynCjWXcDo6KzYotUf
TgGlIIOOJKsAqgOgSHqTz83e8bBizPwJg+CxBzP4Ha8C9phc41i2GiEgDf4J1J0R
0BZDcJEgZIlI+B5tlvJTy/uQyvmEP+hyMD8d83RdzLYy9h8u0MNHjJygY/Kktftp
wPtZPThpMOWWbOMM72a8Y2usz5rKTBAe+bN5QyELCErc/aQB0ABUSsNf4XxaQWbz
gJdb3hEvUkas0PfHui8UB6Yuaa7RmEE6EPIELx2WF2BGw1AG8vg5mi3I+HYxpk9W
mxy2gj63UPqr1f0u7+fnig7ANlyyPYG3LLUfhBT/d9VH0W644lqF8eZo0INEHfQf
+g4qvVVSTWfuC84ky5gTnWMbzB0iqVsZD3xw4wfSrSKyK6QFNESNdOo+1E0nz83I
cQAFD+zSSMLgodHCgA9GlGECAwEAAQ==
-----END PUBLIC KEY-----
```


公钥 2：


```
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAymf92H5ljvvfTE8QjuOd
xv7YPOxXC05VceuSjtZN1aDb/4gqpWxDyMzRrPS8VRQTxkqWia4nd//zj+dheHNv
6+Emb3f00IyC2bcAFvDgQmnQB0sJZf2UI3mbMfLdnsIYW2YCbvxEiFYmUUOnh6xP
```

AnYFtZuvh9EDpyUwT95thQS23UEO2M1y5Q9SRUZo4EeQGb6/iqB6Q5FYabRqbsXe
Ckqxk1ENkPpuLkiQCtra++bICj4WbfVCOiiYpaN/faVud6qMHxsCxxkk+2p7kcs3
ZsCSEmLBzFNmzT32pMK9pq/rAXyXbGh4ECDuTdk1va/cCxIr5Ongven4oe4qGdnj
OCD9xPNfQZDSpYMaBcn1UveM9Rrv/GYaC9AgMnVvG5PaQOYKJzETU2gJm4rdPp/M
Hc9CvN30B6X9ewsLYIaA8ES/DRIqMG4GKAgMz0siROwLXMSkLXg1u4+mLeQzBQP5
TPJ5qwAwKJc6uPPoXo9ZmFnFW4THCoEJ+caax9M0Urg4+B6ids73C2u8A6xqVXld
ng0pAdt5exZqckhPWaWajFt4mmbUmlot7GU9PxV+NDhCn4YDmhBKQRin4lkuLilM
0/WmvnVxD7IhgXbDYrP7E1j/IO9VZQOGkntVT/BtvhJLQauF6J2bxyct9GD6Ahg4
BBKL1/FPLaDsmzWqbNiJKp8CAwEAAQ==
-----END PUBLIC KEY-----

使用公钥 1 加密的对称密钥：

MzhKNQx+U8ltsj5is29pSwu7yqdgoWPWIhgEwUTz3ywE84ue99Z7T/AISGOuyud6ET4E8xXFS/7wadzwY
j3yL6dQrw+F9KFPJRNkTDQll0Re+3kkGt2+M68HJRvmIcJaD1/0PNTv9gek5PdL59TNq/VerwqXusAIIO
dclwhb+U1EGJzJ0RS+8Wyp/+PU4J5P2mtFSak5SKNzDB8yg00uyhRBZGriQzw+QQRZanWJYs45UFYIP+9
ZMUK3lOkf3b8CT+qGW/HcDFwG59hn59PUvN8UFER3PcOTIRD/+RBSKoi1Sdr7uxvQ3XTBvFJKlDMp1es4
yzewmOgluBY2DtGV+aAbLzu5Sy6EfF7tJgid8V9T9ZQ8nqW9vtWkt6Y2okRhdkpX+E+y240gU1BEHOUNg
lM6oJ1b0nGiAL5cjUtX0IknEAsZR/U2ztsMQRzvy10xJpIgipKB52aNh6BnYzFH4DYndfehKh1NjVckcJ
OK+krTiUNwQMNhRYSZ8v1pZH6jR96TuDPib1KcJopjaGdf9zNa2bkdJ7NSWTe9j1jHMPJYjrP6XCefsix
RTWp5dEz3KgzWEgGBHmIhz2SYYWLcy0SKb3ljYFUrY6tDwVRC+Srkk4GOeS09OvxT3r9E/JdaiA9BXuRj
rV7LeCAW18AwbpZEaTHxjrVcoZ5sWpNasCI=

首先对于 RSA 加密算法，公钥（n,e），私钥(n,d)，只知道 n 和 e 要解出 d 是极其困难的，除非我们可以算出 phi(n) = (p-1)(q-1) ,由于给出的两个公钥的大整数有公因子，解析两个公钥得到以下两个大整数 n1 和 n2。

n1:

763876749766176106907097075499136932542506309462790134238145662864029448357672329384240786429825506651735471293192857800292074432517560147934296565646804636440319232118174807874707278193157951022791126432548954297872603841823471708730019040711843329411852093347261870244367947957553762079472966281313126343653099184604229852078759250659887230164618217053184654632257026197939177514038547022906343467564156731888905345558825896885445137274880047279398158012899019023727313182644527939451811489338720718307850784366603399048017626264140284123069450132380165349204708385658847188179122993601433823666052262671276550074507090466095270790435243944044613783720793684122472997119531680070699442004485228493923284784588609625299335494520546908568332978745943998585734768939661601368054137698386418517184469311920103617525768647224332348763739953622815888621102511666165138133372157328365286393142987641723981258561805886361917214660712608451187202188877282119338253380439361187091882619543835302625472246720924991620467674200185807782910046428532568783970685234593602888422305491686373915683830020610512240092345516269641178633217279587340585381147938669982611180322446105897291197929327110789282688669532090634700770073422 9

88590964225250401

n2：

8257453230352727987205996083611924927467876120458769532622892111
6934226834632237529963438818338196764490373355991241721449614542
7027254956721988308512463763281120693615792222840534071393074980
7620772385067911263399091312082641479782008387047030703693284684
9547416412806596658317269602065036109002837409514774790093162333
1337880180057566279781633562179594032007854587452253456011800679
6087636993279776867893270096507914629702419972077312389484742620
6507768527392466652937756530318950877958047549901365112440395211
0758192421559847639791002451557155022419866908364191649354978851
2507363016531527867928341850025492895780887322523488147867126470
5624673739981845168881979994568378470217480791816735406033498184
8371709009902941634174705067621057569867546563489955228609184225
5584156427766703029135474988031623176554245662143644293663677923
8879645609520907530815703192211384732408279212827245608683817944
8027864861814855945170296255542403540294916048693047962499625508
9636804414005895693082341362778985436590428958128339493235920405
2773575657647600522999551515352635095297894182442717707889131728
4336320446076086808680961181640941049041256254665150119759613810
1582616965789228461197684403645635919638430639346958157331305976
65757383432415903

e:65537

计算两个 n 的最大公约数得

gcd：

2662746604914278210118946157408502099623508915699113579952206764759427520652735
9658599180991364006790412470178983959891166450396562210736651239193396713286992
1361424615500700592172129369877382886759429688122276292232759808847508493056641
5408368531799565851707841315673741725230044012912126180730875177529970930728852
1049604030621405834040779873490914247498118977348086366949061671661649633292359
6597694116441723111638463831123458440237117355015386201086191301356691241805731
0360558425747795890330141492653690134887391146811240603150388499183736291482147
360755058914616509567628089785397455593184678644841141626656825

n/gcd：

2868754947828644730514630180882084287179835534780851179676384235662759300503032
8848480720213117750144513421186137860559649154083515009664247554827799501024133
0989584049090174100281703654123437961407768197272215050685216989404045602097831
1368890309302260961603526536264525270531543779940083615748600150099324367000379
4804588069401223165708260378509555227852581333329050238791040257668765437973719
1291135412321562472783246527050428331009155295982718909477961460425384175619320
9202867557398254794563072356857180971372678346648829265282843987747595563553002
38891782609689550563644493694127000591995589436464269344076825593n

由于 n 的特点，猜测其为 p、q，计算拼成私钥所需要的各个数值 phi(n)和 d，得到私钥后解密对称密钥得到：

b'<ce^qG;tQ]Ur3PPa'

接着对密文图像进行处理，先提取密文数据前 16 字节的 IV，接着查看密文数据的最后一个像素，即后四个字节，得到填充像素为 1912 个，据此去除前 16 字节和后 1912 个像素得到真正的密文数据。根据密钥和 IV 创建 AES 解密器解密密文数据，对解密结果去除 PKCS #7 填充，得到真正的明文，将明文重构为图像保存在 dec1.png 中。

程序运行得到的各项数据：

```
n1:  76387674976617610690709707549913693254250630946279013423814566286402944835767232938424078642982550665173547129319285780029207443251756014793429656564680463644031923211817480787
47072781931579510227911264325489542978726038418234717087300190407118433294118520933472618702443679479575537620794729662813131263436530991846042298520787592506598872301646182170531 84
65463225702619793917751403854702290634346756415673188890534555882589688544513727488004727939815801289901902372731318264452793945181148933872071830785078436660339904801762626414028 41
23069450132380165349204708385658847188179122993601433823666052262671276550074507090466095270790435243944044613783720793684122472997119531680070699442004485228493923284784588609625 29
93354945205469085683329787459439985857347689396616013680541376983864185171844693119201036175257686472243323487637399536228158886211025116661651381333721573283652863931429876417239 81
25856180588636191721466071260845118720218887728211933825338043936118709188261954383530262547224672092499162046767420018580778291004642853256878397068523459360288842230549168637391 56
83830020610512240092345516269641178633217279587340585381147938669982611180322446105897291197929327110789282688669532090634700770073422988590964225250401
n2:  82574532303527279872059960836119249274678761204587695326228921116934226834632237529963438818338196764490373355991241721449614542702725495672198830851246376328112069361579222284
05340713930749807620772385067911263399091312082641479782008387047030703693284684954741641280659665831726960206503610900283740951477479009316233313378801800575662797816335621795940 32
00785458745225345601180067960876369932797768678932700965079146297024199720773123894847426206507768527392466652937756530318950877958047549901365112440395211075819242155984763979100 24
51557155022419866908364191649354978851250736301653152786792834185002549289578088732252348814786712647056246737399818451688819799945683784702174807918167354060334981848371709009029 4
16341747050676210575698675465634899552286091842255584156427766703029135474988031623176554245662143644293663677923887964560952090753081570319221138473240827921282724560868381794480 27
86486181485594517029625554240354029491604869304796249962550896368044140058956930823413627789854365904289581283394932359204052773575657647600522999551515352635095297894182442717707 88
91317284336320446076086808680961181640941049041256254665150119759613810158261696578922846119768440364563591963843063934695815733130597665757383432415903
e:  65537
gcd:  26627466049142782101189461574085020996235089156991135799522067647594275206527359658599180991364006790412470178983959891166450396562210736651239193396713286992136142461550070 05
676280897853974555931846786448411416266568257
q:  28687549478286447305146301808820842871798355347808511796763842356627593005030328848480720213117750144513421186137860559649154083515009664247554827799501024133098958404909017410 0
28170365412343796140776819727221505068521698940404560209783113688903093022609616035265362645252705315437799400836157486001500993243670003794804588069401223165708260378509555227852 58
13333290502387910402576687654379737191291135412321562472783246527050428331009155295982718909477961460425384175619320920286755739825479456307235685718097137267834664882926528284398 77
475955635530023889178260968955056364449369412700059199558943646426934407682593
解密后的对称密钥: b'<ce^qG;tQ]Ur3PPa'
iv 7d32566555273e7571674f216b6a2b64
padding_length 1912
```

解密后的结果：

RSA_hack.py:

```
from Crypto.Cipher import AES
from PIL import Image
```

```
import numpy as np
from Crypto.Util.Padding import unpad
from Crypto.PublicKey import RSA
from Crypto.Util.number import inverse
from base64 import b64decode
from math import gcd
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.backends import default_backend
from cryptography.hazmat.primitives.asymmetric.rsa import RSAPublicKey
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives import hashes
import base64
from Crypto.Cipher import PKCS1_OAEP
public_key1_text = """
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAuz20BUTcqVDjzEOKiJF9
66LbQB/59lnXTj/SmiD07mV1XE03BLrWfi7jFh/iq5ZPzVXfbNPjHiojO9WRhWzr
wiQGZNVZ7qFoO/PzXOT8OyHyOMcrb6ogtCyFvDOeximr3M/ICmliU2JxbLSfteZj
AplHJVgs5bJ5LTW7eSy1x2Z5aOsHjesK3rkLi1yB2jM0MeaNIB/Enb82bBMKzAam
vN6tY8bQbEoRbTnlX6PUfkU9w7XsWLMa3QbpIH9mNam1Qz4ynCjWXcDo6KzYotUf
TgGlIIOOJKsAqgOgSHqTz83e8bBizPwJg+CxBzP4Ha8C9phc41i2GiEgDf4J1J0R
0BZDcJEgZIlI+B5tlvJTy/uQyvmEP+hyMD8d83RdzLYy9h8u0MNHjJygY/Kktftp
wPtZPThpMOWWbOMM72a8Y2usz5rKTBAe+bN5QyELCErc/aQB0ABUSsNf4XxaQWbz
gJdb3hEvUkas0PfHui8UB6Yuaa7RmEE6EPIELx2WF2BGw1AG8vg5mi3I+HYxpk9W
mxy2gj63UPqr1f0u7+fnig7ANlyyPYG3LLUfhBT/d9VH0W644lqF8eZo0INEHfQf
+g4qvVVSTWfuC84ky5gTnWMbzB0iqVsZD3xw4wfSrSKyK6QFNESNdOo+1E0nz83I
cQAFD+zSSMLgodHCgA9GlGECAwEAAQ==
-----END PUBLIC KEY-----
"""
public_key2_text = """
-----BEGIN PUBLIC KEY-----
MIICIjANBgkqhkiG9w0BAQEFAAOCAg8AMIICCgKCAgEAymf92H5ljvvfTE8QjuOd
xv7YPOxXC05VceuSjtZN1aDb/4gqpWxDyMzRrPS8VRQTxkqWia4nd//zj+dheHNv
6+Emb3f00IyC2bcAFvDgQmnQB0sJZf2UI3mbMfLdnsIYW2YCbvxEiFYmUUOnh6xP
AnYFtZuvh9EDpyUwT95thQS23UEO2M1y5Q9SRUZo4EeQGb6/iqB6Q5FYabRqbsXe
Ckqxk1ENkPpuLkiQCtra++bICj4WbfVCOiiYpaN/faVud6qMHxsCxxkk+2p7kcs3
ZsCSEmLBzFNmzT32pMK9pq/rAXyXbGh4ECDuTdk1va/cCxIr5Ongven4oe4qGdnj
OCD9xPNfQZDSpYMaBcn1UveM9Rrv/GYaC9AgMnVvG5PaQOYKJzETU2gJm4rdPp/M
Hc9CvN30B6X9ewsLYIaA8ES/DRIqMG4GKAgMz0siROwLXMSkLXg1u4+mLeQzBQP5
TPJ5qwAwKJc6uPPoXo9ZmFnFW4THCoEJ+caax9M0Urg4+B6ids73C2u8A6xqVXld
ng0pAdt5exZqckhPWaWajFt4mmbUmlot7GU9PxV+NDhCn4YDmhBKQRin4lkuLilM
0/WmvnVxD7IhgXbDYrP7E1j/IO9VZQOGkntVT/BtvhJLQauF6J2bxyct9GD6Ahg4
BBKL1/FPLaDsmzWqbNiJKp8CAwEAAQ==
-----END PUBLIC KEY-----
```

```
"""

encrypted_key_Base64 = """

MzhKNQx+U8ltsj5is29pSwu7yqdgoWPWIhgEwUTz3ywE84ue99Z7T/AISGOuyud6ET4E8xXFS/7wadzwYj3yL6dQrw+F9KFPJRNkTDQl10Re+3kkGt2+M68HJRvmI
cJaD1/0PNTv9gek5PdL59TNq/VerwqXusAIIOdclwhb+U1EGJzJ0RS+8Wyp/+PU4J5P2mtFSak5SKNzDB8yg00uyhRBZGriQzw+QQRZanWJYs45UFYIP+9ZMUK3lO
kf3b8CT+qGW/HcDFwG59hn59PUvN8UFER3PcOTIRD/+RBSKoi1Sdr7uxvQ3XTBvFJKlDMp1es4yzewmOgluBY2DtGV+aAbLzu5Sy6EfF7tJgid8V9T9ZQ8nqW9vtW
kt6Y2okRhdkpX+E+y240gU1BEHOUNglM6oJ1b0nGiAL5cjUtX0IknEAsZR/U2ztsMQRzvy10xJpIgipKB52aNh6BnYzFH4DYndfehKh1NjVckcJOK+krTiUNwQMNh
RYSZ8v1pZH6jR96TuDPib1KcJopjaGdf9zNa2bkdJ7NSWTe9j1jHMPJYjrP6XCefsixRTWp5dEz3KgzWEgGBHmIhz2SYYWLcy0SKb3ljYFUrY6tDwVRC+Srkk4GOe
S09OvxT3r9E/JdaiA9BXuRjrV7LeCAW18AwbpZEaTHxjrVcoZ5sWpNasCI=
"""

# 加载公钥
public_key1 = serialization.load_pem_public_key(public_key1_text.encode(), backend=default_backend())
public_key2 = serialization.load_pem_public_key(public_key2_text.encode(), backend=default_backend())

# 获取 n 和 e 值
n1 = public_key1.public_numbers().n
n2 = public_key2.public_numbers().n
e = public_key1.public_numbers().e
print("n1: " , n1 ,"\nn2: ", n2 , "\ne: " , e)

# 获取 gcd
_gcd = gcd(n1 , n2)
print("gcd: " , _gcd)

p = gcd(n1, _gcd)  # p 是 n 和公因子的最大公约数
q = n1 // p
print("p: " , p)
print("q: " , q)
phi = (p - 1) * (q - 1)
d = inverse(e, phi)
#拼接私钥
private_key = RSA.construct((n1 , e , d , p , q))

encrypted_symmetric_key = base64.b64decode(encrypted_key_Base64)
cipher_rsa = PKCS1_OAEP.new(private_key)
symmetric_key_base64 = cipher_rsa.decrypt(encrypted_symmetric_key)
symmetric_key = base64.b64decode(symmetric_key_base64)
print("解密后的对称密钥:", symmetric_key)

# 加载密文图像
image = Image.open("C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB3\\enc1.png\\enc1.png")
image_data = np.array(image)

```

```python
image_bytes = image_data.tobytes()

# 从密文图像中提取 IV

iv = image_bytes[:16]
print("iv" , iv.hex())

padding_length = image_bytes[-4:]
padding_length = int(padding_length.hex() , 16)
print("padding_length" , padding_length)

encrypted_data = image_bytes[16 : ]
encrypted_data = encrypted_data[ : -4 * padding_length  ]

# # 创建 AES 解密器
cipher = AES.new(symmetric_key, AES.MODE_CBC, iv)

# # 解密密文图像数据
decrypted_data_pad = cipher.decrypt(encrypted_data)
decrypted_data_unpad = unpad(decrypted_data_pad , AES.block_size)

dec_image = Image.frombytes("RGBA" ,(1920 , 1080), decrypted_data_unpad)

dec_image.save("C:\\Users\\zigo\\Desktop\\crypto\\LAB\\LAB3\\enc1.png\\dec1.png")
```