rsa

实验制作

实验原理

一、RSA加密算法的数论知识

1、欧拉函数

在数论中，对正整数n，欧拉函数是小于n的正整数中与n互质的数的数目，以φ(n)表示。

如果n是质数，则`ϕ(n)=n-1`

其中对RSA最重要的一种情况就是：

如果n可以分解成两个互质的整数之积

```
n = p1 × p2
```

则：

```
ϕ(n) = ϕ(p1p2) = ϕ(p1)ϕ(p2)
```

2、欧拉定理

欧拉定理表明，若m,a为正整数，且m,a互质，则以下公式成立：

$$a^{\varphi(m)} \equiv 1(\bmod m)$$

3、逆元运算：

如果两个正整数a和n互质，那么一定可以找到整数b，使得 ab-1 被n整除，或者说ab被n除的余数是1。这时，b就叫做a的“逆元”。

$$ab \equiv 1\,(mod\ n)$$

不难看出，a的 φ(n)-1 次方，就是a对模数n的模反元素：

$$a^{\phi(n)} = a \times a^{\phi(n)-1} \equiv 1\,(mod\ n)$$

4、拓展欧几里得算法

将过程用矩阵表示，（其中q表示商，r表示余数），如下图所示：

$$\begin{pmatrix} a \\ b \end{pmatrix} = \prod_{i=0}^{N} \begin{pmatrix} q_i & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} r_{N-1} \\ 0 \end{pmatrix}.$$

5、高次同余方程的解法：

假设 p 和 q 是不同的素数,并假设 $e \geq 1$,满足：

```
                 gcd(e,(p-1)(q-1)) = 1
```

则e 模(p-1)(q-1)存在逆元 d ,即:

```
                 de≡1(mod(p-1)(q-1))
```

则同余方程：

```
                  x^e≡(mod pq)
```

  那么有唯一解：

```
                   x≡c^d(mod pq)
```

二、RSA 公钥密码体系的实现方案

1、生成密钥过程

　　随机选择两个不相等的大质数p和q，计算p和q的乘积n，计算n的欧拉函数φ(n)。随机选择一个整数e，条件是1< e < φ(n)，且e与φ(n) 互质。计算e对于φ(n)的模反元素d。将n和e封装成公钥，n和d封装成私钥。

2、加密

通过公钥进行加密（n,e）

设明文为m，密文为c，则加密公式为：

```
                    c≡m^e(mod n)
```

3、通过私钥进行解密

密文为c，明文为m，解密公式为：

```
                     m≡c^d(mod n)
```

4、RSA安全性

RSA加密算法的安全性主要原理在于在已知n和e的情况下，并不能快速实现大整数的因数分解。即对于一个由两个大素数p,q组成的N=p*q，通过分解N得到p，q进而求解同余方程 x^e≡c(mod N)，得到x是很困难的。

三、算法加速原理

1、快速模幂运算

在这里我们采用平方乘算法进行快速模幂运算，模幂运算进行加速：

```python
def quickPower(x, n, m):
    res = 1
    while n > 0:
        if n % 2 == 1:
            res = (res * x) % m
        x = (x * x) % m
        n //= 2
    return res
```

2、中国剩余定理优化

选出p，q两个大素数之后，我们可以利用中国剩余定理分步计算，来让数据变小，来实现加速。

首先我们说明中国剩余定理：假设整数m_1,m_2,......，m_n两两互素，则对于任意的整数a_1,a_2,......,a_n，方程组：

$$\begin{cases} x \equiv a_1 \pmod{m_1} \\ x \equiv a_2 \pmod{m_2} \\ \cdots \\ x \equiv a_n \pmod{m_n} \end{cases}$$

都存在整数解，且若x,y都满足该放成组，则必有x≡y(mod N)，其中N=$m_1 m_2 m_3$......m_n.

下面我们来使用CTR模式更有效的计算m=c^d

首先使用拓展欧几里得算法来计算e模p-1和q-1的逆元和q模p的逆元

```
dP=e^-1 mod(p-1)
dQ=e^-1 mod(q-1)
dV=q^-1 mod p
```

下面我们来计算明文M

```
m1=c^dP mod p
m2=c^dQ mod q
h=dV*(m1-m2) mod p
m=m2+hq
```

代码如下：

```python
def crt(p, q, e, c):
    dp, _, _ = Egcd(e, p - 1)
    dq, _, _ = Egcd(e, q - 1)
    dv, _, _ = Egcd(q, p)
    while dp < 0:
        dp += p - 1
    while dq < 0:
        dq += q - 1
    while dv < 0:
        dv += p
    print('dp=', dp)
```

```
    print('dq=', dq)
    print('dv=', dv)
    m1 = quickPower(c, dp, p)
    m2 = quickPower(c, dq, q)
    print('m1=', m1)
    print('m2=', m2)
    h = (dv * (m1 - m2)) % p
    m = (m2 + h * q) % (p * q)
    return m
```

实验代码

一、函数部分

1、求解最大公因数：

```
def g_cd(a, b):
    if a > b:
        a, b = b, a
    while b != 0:
        temp = a % b
        a = b
        b = temp
    return a
```

2、拓展欧几里得算法求解逆元

```
def Egcd(a, b):
    if b == 0:
        return 1, 0, a
    else:
        x, y, q = Egcd(b, a % b)
        x, y = y, (x - (a // b) * y)
        return x, y, q
```

3、快速模幂运算

```
def quickPower(x, n, m):
    res = 1
    while n > 0:
        if n % 2 == 1:
            res = (res * x) % m
        x = (x * x) % m
        n //= 2
    return res
```

4、米勒拉宾素性检测：

```
def MillerRabin(n, s):  # 米勒拉宾素性检测
    if n == 2:
        return True
    if n & 1 == 0 or n < 2:
        return False
    m, p = n - 1, 0
```

```
    while m & 1 == 0:
        m = m >> 1
        p += 1
    for _ in range(s):
        b = quickPower(random.randint(2, n - 1), m, n)
        if b == 1 or b == n - 1:
            continue
        for __ in range(p - 1):
            b = quickPower(b, 2, n)
            if b == n - 1:
                break
        else:
            return False
    return True
```

5、生成素数

```
def get_prime():
    while True:
        num = random.randrange(2 ** 1024, 2 ** 1030)
        if MillerRabin(num, 20):
            return num
```

6、CRT加速函数

```
def crt(p, q, e, c):
    dp, _, _ = Egcd(e, p - 1)
    dq, _, _ = Egcd(e, q - 1)
    dv, _, _ = Egcd(q, p)
    while dp < 0:
        dp += p - 1
    while dq < 0:
        dq += q - 1
    while dv < 0:
        dv += p
    print('dp=', dp)
    print('dq=', dq)
    print('dv=', dv)
    m1 = quickPower(c, dp, p)
    m2 = quickPower(c, dq, q)
    print('m1=', m1)
    print('m2=', m2)
    h = (dv * (m1 - m2)) % p
    m = (m2 + h * q) % (p * q)
    return m
```

二、完整代码：

```
import random
from libnum import invmod

from AITMCLAB.libnum import s2n, n2s
```

```python
def g_cd(a, b):
    if a > b:
        a, b = b, a
    while b != 0:
        temp = a % b
        a = b
        b = temp
    return a


def quickPower(x, n, m):
    res = 1
    while n > 0:
        if n % 2 == 1:
            res = (res * x) % m
        x = (x * x) % m
        n //= 2
    return res


def relatively_prime(a, b):  # a > b
    while b != 0:
        temp = b
        b = a % b
        a = temp
    if a == 1:
        return True
    else:
        return False


def MillerRabin(n, s):  # 米勒拉宾素性检测
    if n == 2:
        return True
    if n & 1 == 0 or n < 2:
        return False
    m, p = n - 1, 0
    while m & 1 == 0:
        m = m >> 1
        p += 1
    for _ in range(s):
        b = quickPower(random.randint(2, n - 1), m, n)
        if b == 1 or b == n - 1:
            continue
        for __ in range(p - 1):
            b = quickPower(b, 2, n)
            if b == n - 1:
                break
        else:
            return False
    return True


def get_prime():  # 生成大素数
    while True:
```

```python
        num = random.randrange(2 ** 1024, 2 ** 1030)
        if MillerRabin(num, 20):
            return num


def Egcd(a, b):
    if b == 0:
        return 1, 0, a
    else:
        x, y, q = Egcd(b, a % b)
        x, y = y, (x - (a // b) * y)
        return x, y, q


def crt(p, q, e, c):
    dp, _, _ = Egcd(e, p - 1)
    dq, _, _ = Egcd(e, q - 1)
    dv, _, _ = Egcd(q, p)
    while dp < 0:
        dp += p - 1
    while dq < 0:
        dq += q - 1
    while dv < 0:
        dv += p
    print('dp=', dp)
    print('dq=', dq)
    print('dv=', dv)
    m1 = quickPower(c, dp, p)
    m2 = quickPower(c, dq, q)
    print('m1=', m1)
    print('m2=', m2)
    h = (dv * (m1 - m2)) % p
    m = (m2 + h * q) % (p * q)
    return m


def crt_decryption(p, q, c, e):  # 通过中国剩余定理加速解密
    phi = (p - 1) * (q - 1)
    c1 = c % p
    c2 = c % q
    d = invmod(e, phi)
    d1 = d % (p - 1)
    d2 = d % (q - 1)
    m1 = pow(c1, d1, p)
    m2 = pow(c2, d2, q)
    print('m1=', m1)
    print('m2=', m2)
    p1 = invmod(p, q)
    q1 = invmod(q, p)
    N = p * q
    m = (m2 * p * p1 + m1 * q * q1) % N
    ans = m
    return ans
```

```python
p = get_prime()
q = get_prime()
print('p=', p)
print('q=', q)
N = p * q
phi = (p - 1) * (q - 1)
g = g_cd(p - 1, q - 1)
e = 5
while g_cd(e, phi) != 1:
    e += 1
d, _, _ = Egcd(e, phi // g)
if d < 0:
    d += phi
print('p=', p)
print('q=', q)
print('e=', e)
print('phi=', phi)
print('d=', d)
s = input('请输入密文')
s = s2n(s)
m = s ** e % (p * q)
print('明文为（m,N）=', (m, p * q))
r = quickPower(m, d, N)
print("解密得r=", r)
q_r = crt(p, q, e, m)
print('密码为', q_r)
print('密码为', n2s(q_r))
x = crt_decryption(p, q, m, e)
print('x=',x)
print('crt加速得到密码为=', n2s(x))import random
from libnum import invmod

from AITMCLAB.libnum import s2n, n2s


def g_cd(a, b):
    if a > b:
        a, b = b, a
    while b != 0:
        temp = a % b
        a = b
        b = temp
    return a


def quickPower(x, n, m):
    res = 1
    while n > 0:
        if n % 2 == 1:
            res = (res * x) % m
        x = (x * x) % m
        n //= 2
    return res
```

```python
def relatively_prime(a, b):  # a > b
    while b != 0:
        temp = b
        b = a % b
        a = temp
    if a == 1:
        return True
    else:
        return False


def MillerRabin(n, s):  # 米勒拉宾素性检测
    if n == 2:
        return True
    if n & 1 == 0 or n < 2:
        return False
    m, p = n - 1, 0
    while m & 1 == 0:
        m = m >> 1
        p += 1
    for _ in range(s):
        b = quickPower(random.randint(2, n - 1), m, n)
        if b == 1 or b == n - 1:
            continue
        for __ in range(p - 1):
            b = quickPower(b, 2, n)
            if b == n - 1:
                break
        else:
            return False
    return True


def get_prime():  # 生成大素数
    while True:
        num = random.randrange(2 ** 1024, 2 ** 1030)
        if MillerRabin(num, 20):
            return num


def Egcd(a, b):
    if b == 0:
        return 1, 0, a
    else:
        x, y, q = Egcd(b, a % b)
        x, y = y, (x - (a // b) * y)
        return x, y, q


def crt(p, q, e, c):
    dp, _, _ = Egcd(e, p - 1)
    dq, _, _ = Egcd(e, q - 1)
    dv, _, _ = Egcd(q, p)
    while dp < 0:
        dp += p - 1
```

```python
    while dq < 0:
        dq += q - 1
    while dv < 0:
        dv += p
    print('dp=', dp)
    print('dq=', dq)
    print('dv=', dv)
    m1 = quickPower(c, dp, p)
    m2 = quickPower(c, dq, q)
    print('m1=', m1)
    print('m2=', m2)
    h = (dv * (m1 - m2)) % p
    m = (m2 + h * q) % (p * q)
    return m


def Sha256sum(message: bytes) -> bytes:  # 通过hash生成内容摘要
    h0 = 0x6a09e667
    h1 = 0xbb67ae85
    h2 = 0x3c6ef372
    h3 = 0xa54ff53a
    h4 = 0x510e527f
    h5 = 0x9b05688c
    h6 = 0x1f83d9ab
    h7 = 0x5be0cd19
    K = (0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5, 0x3956c25b, 0x59f111f1,
         0x923f82a4, 0xab1c5ed5, 0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3,
         0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174, 0xe49b69c1, 0xefbe4786,
         0x0fc19dc6, 0x240ca1cc, 0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,
         0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7, 0xc6e00bf3, 0xd5a79147,
         0x06ca6351, 0x14292967, 0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13,
         0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85, 0xa2bfe8a1, 0xa81a664b,
         0xc24b8b70, 0xc76c51a3, 0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,
         0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5, 0x391c0cb3, 0x4ed8aa4a,
         0x5b9cca4f, 0x682e6ff3, 0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208,
         0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2)

    def R(x, n):
        return ((x >> n) | (x << (32 - n))) & 0xffffffff

    def W(i1, i2, i3, i4):
        return (i1 << 24) | (i2 << 16) | (i3 << 8) | i4

    ascii_list = list(map(lambda x: x, message))
    msg_length = len(ascii_list) * 8
    ascii_list.append(128)
    while (len(ascii_list) * 8 + 64) % 512 != 0:
        ascii_list.append(0)
    for i in range(8):
        ascii_list.append(msg_length >> (8 * (7 - i)) & 0xff)
    for i in range(len(ascii_list) // 64):
        w = []
        for j in range(16):
            s = i * 64 + j * 4
            w.append(W(ascii_list[s], ascii_list[s + 1],
```

```python
                        ascii_list[s + 2], ascii_list[s + 3]))
        for j in range(16, 64):
            s0 = (R(w[j - 15], 7)) ^ (R(w[j - 15], 18)) ^ (w[j - 15] >> 3)
            s1 = (R(w[j - 2], 17)) ^ (R(w[j - 2], 19)) ^ (w[j - 2] >> 10)
            w.append((w[j - 16] + s0 + w[j - 7] + s1) & 0xffffffff)
        a, b, c, d, e, f, g, h = h0, h1, h2, h3, h4, h5, h6, h7
        for j in range(64):
            s0 = R(a, 2) ^ R(a, 13) ^ R(a, 22)
            maj = (a & b) ^ (a & c) ^ (b & c)
            t2 = s0 + maj
            s1 = R(e, 6) ^ R(e, 11) ^ R(e, 25)
            ch = (e & f) ^ ((~e) & g)
            t1 = h + s1 + ch + K[j] + w[j]
            h = g & 0xffffffff
            g = f & 0xffffffff
            f = e & 0xffffffff
            e = (d + t1) & 0xffffffff
            d = c & 0xffffffff
            c = b & 0xffffffff
            b = a & 0xffffffff
            a = (t1 + t2) & 0xffffffff
            h0 = (h0 + a) & 0xffffffff
            h1 = (h1 + b) & 0xffffffff
            h2 = (h2 + c) & 0xffffffff
            h3 = (h3 + d) & 0xffffffff
            h4 = (h4 + e) & 0xffffffff
            h5 = (h5 + f) & 0xffffffff
            h6 = (h6 + g) & 0xffffffff
            h7 = (h7 + h) & 0xffffffff

    digest = (h0 << 224) | (h1 << 192) | (h2 << 160) | (h3 << 128)
    digest |= (h4 << 96) | (h5 << 64) | (h6 << 32) | h7
    return hex(digest)[2:]


def sign(N, e, m):
    ans = quickPower(m, e, N)
    return ans



p = get_prime()
q = get_prime()
print('p=', p)
print('q=', q)
N = p * q
phi = (p - 1) * (q - 1)
g = g_cd(p - 1, q - 1)
e = 5
while g_cd(e, phi) != 1:
    e += 1
d, _, _ = Egcd(e, phi // g)
if d < 0:
    d += phi
print('p=', p)
print('q=', q)
```

```python
print('e=', e)
print('phi=', phi)
print('d=', d)
s = input('请输入密文')
smessage = s.encode('utf-8')
s = s2n(s)
m = s ** e % (p * q)
print('明文为（m,N）=', (m, p * q))
ssmessage = Sha256sum(smessage)
ssmessage=s2n(ssmessage)
print('文字摘要为:', ssmessage)
S_sign = sign(N, e, s)
print('加密签字为: ', S_sign)
r = quickPower(m, d, N)
print("解密得r=", r)
q_r = crt(p, q, e, m)
print('密码为', q_r)
print('crt加速的到的密码为', n2s(q_r))
print('加密签字为', S_sign)
ssmassage= crt(p, q, e, S_sign)
print('文字摘要为:', ssmessage)
```

三、实验结果截图

```
p= 451738858304179754248304935529981950550570662101963536366383479813150248594143231698607
q= 522636469265373635280119606506890504548671567045579878506647148025709119545923739358963
p= 451738858304179754248304935529981950550570662101963536366383479813150248594143231698607
q= 522636469265373635280119606506890504548671567045579878506647148025709119545923739358963
e= 5
phi= 23609520193406741772709286905098058232277335140512637432278033181310198980695444554l4
d= 21248568174066067595438358214588252409049601626446137368905022986317917908262590009873l6
请输入密文>? I LOVE BUAA
明文为（m,N）= (539960807238988005338817349885957308109013703330160504876770578821975560553
文字摘要为: 273335964215919842831148703123093098438080935626081044972837099083475003345622282
加密签字为:  5399608072389880053388173498859573081090137033301605048767705788219755605539707
解密得r= 88404108228702142173888833
dp= 180695543321671901699321974211992780220228264840785414546553391925260099437657292679442
dq= 418109175412298908224095685205512403638937253636463902805317718420567295636738991487176
dv= 223408529861367933264269809848730610270316781848100664089051821656420775305816164440074
m1= 88404108228702142173888833
m2= 88404108228702142173888833
密码为 88404108228702142173888833
crt加速的到的密码为 b'I LOVE BUAA'
加密签字为 53996080723898800533881734988595730810901370333016050487677057882197556055397076
dp= 180695543321671901699321974211992780220228264840785414546553391925260099437657292679442
dq= 418109175412298908224095685205512403638937253636463902805317718420567295636738991487176
dv= 223408529861367933264269809848730610270316781848100664089051821656420775305816164440074
m1= 88404108228702142173888833
m2= 88404108228702142173888833
文字摘要为: 273335964215919842831148703123093098438080935626081044972837099083475003345622282
```

思考题

rsa参数选择

1、使用不同指数进行解密时，不能使用相同的模数。若使用了相同的模数N，且采用不同的指数e1，e2对同一密文进行加密。可以由下式求出m

$c_1^u * c_2^v \equiv (m^{e1})^u * (m^{e2})^v \equiv m^{e_1 * u + e_2 * v} \equiv m^{gcd(e_1+e_2)}$(mod N)

且当gcd（e_1,e_2）=1时，可以算出密文。

2、若选择不同模数，要避免出现公因子，若存在公因子，密钥就被泄露了。

3、p,q都应该为强素数，且相差不能较大或较小。相差太大用Format可以快速将n分解成功。相差太小可以由$((p+q)/2)^2$=N+$((p-q)/2)^2$,快速求出p+q，进而求出p,q.

4、体系中的密钥d要求满足d>$N^{1/4}$,否则根据连分数理论进行破解。

5、e不能过小，且e要避免选择了不动点加密，存在（e-1,p-1）*(e-1,q-1)个不动点，即经过加密但明文未发生改变。

6、若e较小，要避免使用相同指数来加密，否则根据中国剩余定理易解。

RSA攻击

一、level 1

1、实验原理

本次实验主要采取共模攻击：即采取同一个模数N和不同e得到的不同的c，当c1与c2互素时，我们可以采用如下方式

$c_1^u * c_2^v \equiv (m^{e1})^u * (m^{e2})^v \equiv m^{e_1 * u + e_2 * v} \equiv m^{gcd(e_1+e_2)}$(mod N)

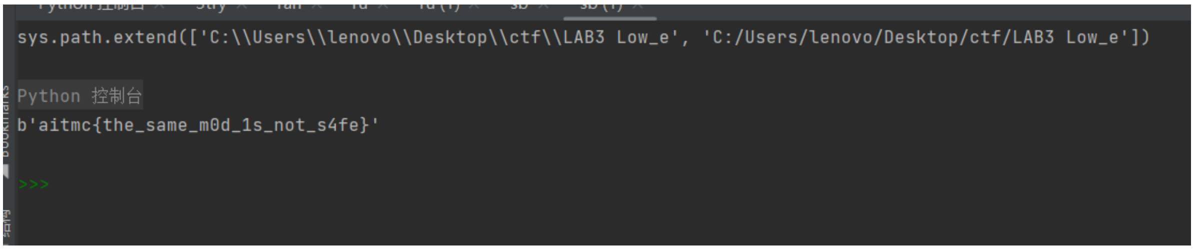
有n个e时，若所有e的最大公因数为1，即得到密文。

2、实验代码

```
from libnum import*    #python第三方库
from gmpy2 import*    #python第三方库

from AITMCLAB.Crypto.Util.number import long_to_bytes

n =
2004864788734120552344497735501070776527424082019874088606388851250203358776986831448627112933369441310938707156205569000522852555993404327716104630831189800075277699909232742106159531182992199753285420426716157807980101262472719612358837443084617921332501600539749026602673897263880833754599005536992855246699741158767585222276179402618050770796631243227406745329696766662159153338652075684244194305177372501553297472311649166005593706029879033034560687030974313811845943434679692310288751477528788188343990808958263264006125925874000741987071896670760752115929179109762406722996928107714335464300166440391399164446
```

```
c1 =
1413750055207022941568329187446066487403500765460761572227376144607660240395326522973307046004979624601604213224918988321501270059071500066882064209247917554745581793896555798560939410733596915266418532287017013823435344990036597967445573969487062227380529627028612285802464882496212301769067172853758330070138009767394870042613078386721339264858678615843222784832621954578982376854514839736276400798150419214497934598226180352252977305319622252080960481078671833413346188828033910689631932089823729264542083315620234674282590493797576023672980537919574980902978546192003654543357786779471585690173302454429025344059
c2 =
3375578062147608533850942853901102056040887531669784350666079177604242560382170075843460566578012075575179567441729528870198216253852376401440958520033848614051445851340337944894932838437790028030819526143444835060144971455117365151495541390380862298201820501237365577224283542534902146440674742286322212972077455029961075853352777137395404560387408496866501461784748715553927384312433630677956229220272763083964452080984735875087340855266572703126596019883254182483014350864897489088531330274319866910219337758689741037902819077568689721448942277419844163504591449807017766332629276288893591669645092166348329361353

e1 = 65537
e2 = 963419
s = gcdext(e1, e2)
#print(s[1],s[2])
m = pow(c1, s[1], n) * pow(c2, s[2], n) % n
print(long_to_bytes(m))
```

3、实验结果截图



二、level 2

1、实验原理

在本题中，我们已知N,d,e,求p和q：

我们首先可以计算c^d(mod N)即可得出msg部分

在flag部分，根据*ed-1=h*φ(n)*N=p*q,又因为p，q都为素数，且φ（n）和N相差不大，可以求出h。又根据N-φ(n)+1=p+q

在这里我们由此可得x^2-(N-φ(n)+1)^x+N=0的两个根为p和q。

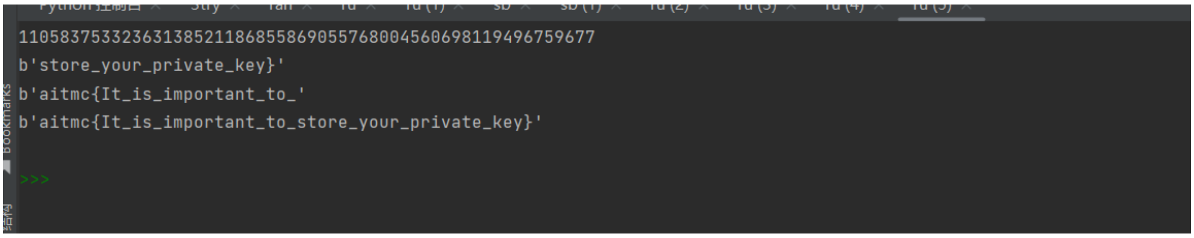
2、实验代码

```
from AITMCLAB.libnum import n2s, invmod
```

```
N =
25970726610338659267902671451464773756712155863832930855238390290240456722241080485553564940848022863998809229728073350855413141350633381566623815046007291793873422432707151304463711842596343644646339091082719750213476751217802777888996404230739705019320445268505461958762957221270655475304524829617525677088465316537506197975678939746161327157805036227593539044106874299010464860542026296675840517906452722561308125819784101289137200063394910686693411351196911509767492526029916359897706663836818160229615043283180754137324552167683470604665636279139090871144489388856313368869399929663263862202482327851938153478759
c =
1106457896298103630917249642458722346474677222861194377991296904908252699798031563494363326336894592629136661844258576535767120669622903791058065190063084714380148528162687937982564278448020161313670117700363580801883885534668777011170432453679822127707173286759174958426435319148515091285062232066599613616629688739231208437096109411353877310125531710568376397681069501899277187318566953296542416849005169947230862957770313551582386882365296224449746092962434411076934234168282980443227978327227507988981325787652160018086902216632561614738343953761895438255726209955476693816592548604555307970211977868490809608856
d =
1263189345777585760687427257603431204232180325540339951403263986613571113330479442023555100140356692551575757167871573306623104732423850343172504747469275040805701149920527822209838046434276639007011661796501134175122393562274203079122926113820766978532025524984619845974295329384037690323289039253278442302346493234322126877772895004287203360311119218262183975257501809671239968114579651654280666794446143026125938586026867979675180269044399764760495655040874402472300447158783357074688152034108590045635565913864069408039472155844680890349435674640890848320604800642308553825239071780077095211804513796584758407018
3
e =
1318663786333660890934884332616457183048836984938239050925767197568154308341372641397286786205445002742322412134697646276079346406563823269260285747067849297762411966747806494031676411301470085935007801369331811316632227455436296272959501090911683054699071154520786435526716943913393960543273929874861346408981637964722627072929818718298954905774873855658251999604066052499524634091204505647547750116214074209396197152446927976412702047074045885033778867535117174389228020891919053805726197581019367850053502747711610344785165033842605343806797328141637929638963291431200508762633049627205449455564438649783967417168
7
c2 =
7745900749084423049596580137321067188819246985151183218015597589581022409864808266858948232299615968686964811161125550884105160017151869173459093139109621916472239592067251032551186107896281868610138241925321663060190442224402999884583547794487559214162838724183384077249994510523976393458976643350133536750624067724004267632144552098017800797908757309798013165532750934944348052352866718084902729599783700928207617893491152926693616415257098134854954791358748362611389718775738525762030216522801016951169610798804099879973572604248772597531628542574277722963655793995477408199234112196383428963526140553678791674014
e2 = 0x20211011
msg = pow(c, d, N)
print(msg)
print(n2s(msg))
```

```
phi =
25970726610338659267902671451464773756712155863832930855238390290240456722241080
48555356494084802286399880922972807335085541314135063338156662381504600729179387
34224327071513044637118425963436446463390910827197502134767512178027778889964042
30739705019320445268505461958762957221270655475304524829617525677088141644770856
42329546026423995302418954182759072572185100441324140942084669209217102640959980
91543256146216212477990970732658557755591311179679089636021031436299457968984462
21673953810300285204273143243934428552554149723138154490818775361700341682147775
248358639594109989601328703498871829212758339718956314120
d2 = invmod(e2, phi)
flag = pow(c2, d2, N)
print(n2s(flag))
print(n2s(flag)+n2s(msg))
```

3、实验结果截图



三、level 3

1、实验原理

当d足够小时，我们可以采用连分数理论的相关知识破解出φ(N)。

前提条件为3*d<$n^{1/4}$且q<p<2q.说明如下：

ab≡1（modφ(n)）,即de-*t*φ(n)=1。又已知n=pq>q^2，所以有q<$\sqrt{n}$,所以有如下放缩：0<n-φ(n)=p+q-1<2q+q-1<3$\sqrt{n}$

所以有|e/n-t/d|=|(de-*tn*)/（*an)*|=|*(1+t*(φ(n)-n))/(an)|<3t$\sqrt{n}$/an=3*t*/*a*$\sqrt{n}$,

有根据已知t<d,所以有3*t*<3d<$n^{1/4}$故而上式化为|e/n-t/d|<1/(3a^2).由连分数的理论不难得到，在表示e\n的连分数过程中，必定有一个收敛子列t/d.

对rsa攻击算法转化为求解带入e/n的所有收敛子列，通过求解一元二次方程并判断根是否存在来判断出正确的收敛子列，进而得到φ(N).

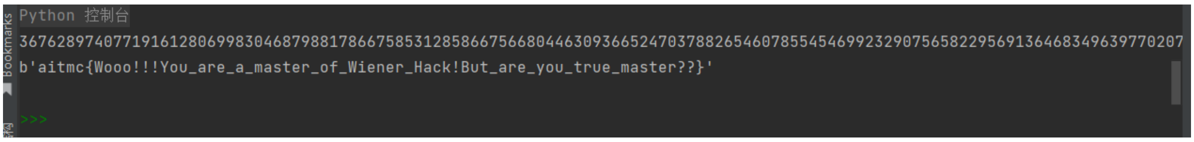
2、实验代码

```
import gmpy2
import libnum
from libnum import n2s
```

```
n =
13777970255200678809137296518027992113852131244629924250922853378036497013574932
07265263054897005792123217107674426360061242665393347075332800665403137169665899
97079841622385754284353195124750075551414568276611675612643713669223398654398663
62852460188560464264160060177347729459691652836652319714768498781233811232348191
52531373223321810050835127753529896510379213062316554810701990790952865633928326
63040581410124651877938843235540279142103554865560004305083012700943264508443751
15558524140185325101668988902728884210078075350221853355818284208197637757656070
8474324601508321090235516171622234210667277403353351991 67
e =
11743259336559359282080468041122941428148861549834698580115094234342752097337312
39498568265027712366089327039842520463513934001297819916352996051526604906533759
24681476944179895073940118005990807623251157335715127237073937476463464068906323
40224724856560864835136805816144772011267138503469724116934421471516068990455730
64571725151552793392451575071371020403094608020720779276556865055458086418644490
37759069200804188321911408465187526292444566117287003442061270630994709226350266
55173951771706152239879002529440731954946297903178159920211222015998828456232099
678376615013226390558468086420150061943604288286295059617
c =
69581210999019861571820161664441669289096521929334695478250464755090863409402 75
73126559330742493064338102299865853667605088850014504362468156148001946017993242
77590701026309618064736129097867494544143290174660181916684524470814341160928863
23526500617553053234919849133329471954365422291627345073860241656562308137845493
18638912082839526655980638870529773223781880273069310149710004932359082558061566
54279276756728962375017530094070433749796584435867730007859219028370696136625910
39149785702035234677351258860844881262226215180628380083920481719902606203135520 9
140328981551170737875364076989089551061441896962629 51803


def cal(ans):
    num = 0
    den = 1
    for x in ans[::-1]:
        num, den = den, x * den + num
    return num, den

def solve(a, b, c):
    par = gmpy2.isqrt(b * b - 4 * a * c)
    return (-b + par) // (2 * a), (-b - par) // (2 * a)


ans = []
x, y = e, n
while y:
    ans.append(x // y)
    x, y = y, x % y
ans2 = []
for i in range(1, len(ans) + 1):
    ans2.append(cal(ans[:i]))
for d, k in ans2:
    if k == 0:
        continue
    if (e * d - 1) % k != 0:
        continue
    phi = (e * d - 1) // k
```

```
    p, q = solve(1, n - phi + 1, n)
    if p * q == n:
        break
flag = pow(c, d, n)
print(flag)
print(n2s(flag))
```

3、实验结果截图



拓展部分

对消息进行数字编码和加解密

一、实验原理

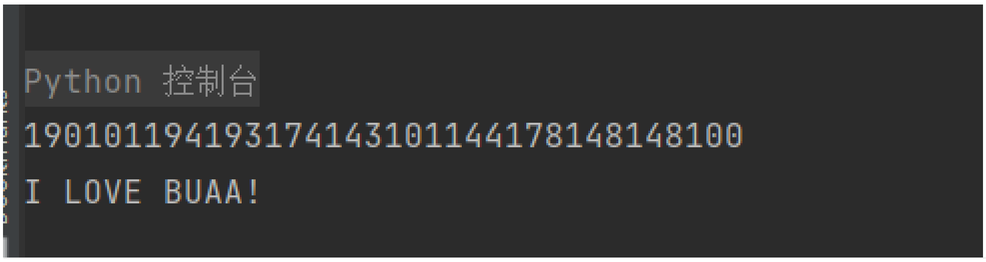
数字编码：把字符串先变成字节数组，把每个字节都转化为数字字符串，区间在0~255，并在每个数字串前加上长度位。再随机出一种简单数字进行替换，保证意义对应

二、实验代码

```
import random


def k_ey():
    ss = set()
    ret = ''
    while 1:
        length = len(ss)
        item = random.randint(0, 9)
        ss.add(item)
        if len(ss) > length:
            ret += str(item)
        if len(ret) == 10:
            return ret


def encrypt(str1, password='1938762450'):
    data = bytearray(str1.encode('utf-8'))
    List = [str(byte) for byte in data]
    List = [str(len(s)) + s for s in List]
    for index0 in range(len(List)):
        item = ''
        for index in range(len(List[index0])):
            item = item + password[int(List[index0][index])]
        List[index0] = item
    return ''.join(List)


def decrypt(strr, password='1938762450'):
```

```
    tem = ''
    for index in range(len(strr)):
        tem += str(password.find(strr[index]))
    index = 0
    list = []
    while 1:
        length = int(tem[index])
        s = tem[index + 1:index + 1 + length]
        list.append(s)
        index += 1 + length
        if index >= len(tem):
            break
    data = bytearray(len(list))
    for i in range(len(data)):
        data[i] = int(list[i])
    return data.decode('utf-8')


m = 'I LOVE BUAA!'
key = k_ey()
password = ''
d = encrypt(m, password=key)
print(d)
print(decrypt(d, password=key))
```

三、实验结果截图



思考与感悟

纸上得来终觉浅，绝知此事要躬行。rsa理论知识并不难懂，然而实践起来，在参数的选择，和利用参数漏洞来攻击却别有一番感悟，也更加加深我对rsa加密算法理解。