RSA实验报告

【实验目的】

1.掌握RSA算法原理及实现。

2.了解常见的RSA攻击方法。

【实验环境】

本次实验使用python语言编写，在Pycharm的python3.9环境下运行并测试。

【实验内容】

1 RSA

1.1 算法流程

加解密模式的函数调用图如下所示：

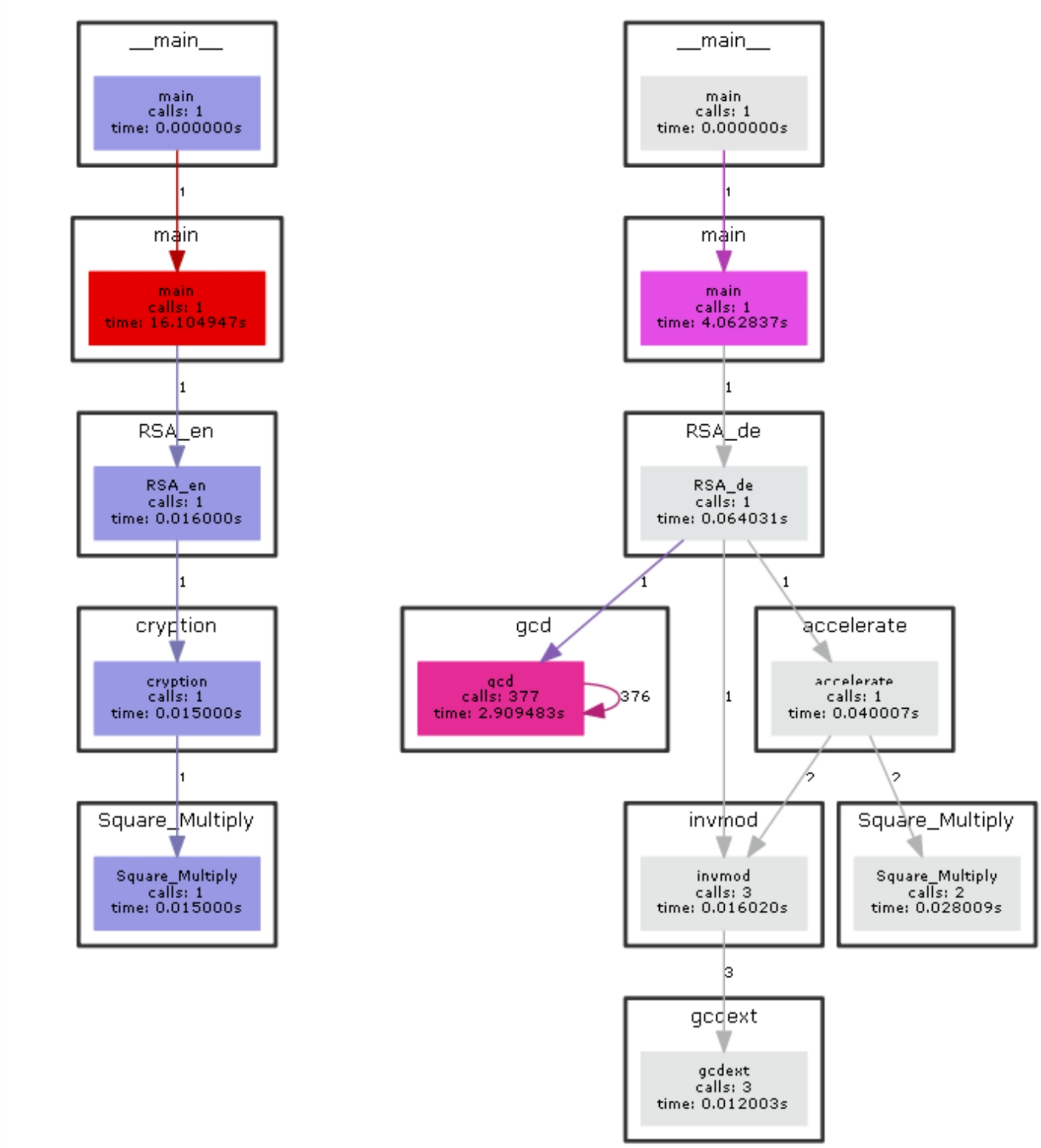

图1 RSA加解密函数调用

其中accelerate模块是利用中国剩余定理加速解密，原理如下：在解密时，进行

$$m \equiv c^d \ mod \ p * q$$

运算时，可以利用中国剩余将其分解为方程组

$$\{\begin{matrix} m \equiv c^d \ mod \ p \\ m \equiv c^d \ mod \ q \end{matrix}$$

缩小模数及幂次以加快运算速度，其中，令

$$d_1 \equiv d \ mod \ p-1$$
$$d_2 \equiv d \ mod \ q-1$$
$$c_1 \equiv c \ mod \ p$$
$$c_2 \equiv c \ mod \ q$$
$$m_1 \equiv c_1^{d_1} \ mod \ p$$
$$m_2 \equiv c_2^{d_2} \ mod \ q$$
$$q' \equiv q^{-1} \ mod \ p$$

那么最后方程的解为$m \equiv m_1 \times q \times q' + m_2 \times p \times p' \ mod \ N$

下面给出参数p，q，e的生成伪代码：

1.1.1 p，q的生成

算法 1 生成大素数p

```
输入: n
输出: p
function GETPRIME(n)
    while 1 do
        x = randrange(1 << (n//4), 1 << (n//4 + 1))
        if MillerRabintest(x)==1 then
            break
        end if
    end while
    while 1 do
        q1 = randrange(1 << (n//4), 1 << (n//4 + 1))
        r = q1 × x + 1
        if MillerRabintest(r)==1 then
            break
        end if
    end while
    while 1 do
        q2 = randrange(1 << (n//4), 1 << (n//4 + 1))
        p = r × q2 + 1
        if MillerRabintest(p)==1 then
            break
        end if
    end while
    return p
end function
```

图2 生成p，q的伪代码

其中用到的MillerRabin算法在实验一中已经实现，这里就不再赘述，本次实验中以10次MillerRabin测试为依据来判定生成的随机数是不是素数，这是因为在素性检验时正确的概率是75%，那么十次后的误判概率为$\frac{1}{4^{10}}$ <0.0001%，几乎满足实际需要。且这样的素数满足了强素数的需要。

> 1. p是很大的素数
> 2. p-1有很大的质因数，即对于某个整数a以及大素数q1，有
>
> $$p = a \times q_1 + 1$$
>
> 3. q1 -1有很大的质因数，即对于某个整数b与大素数q2，有

> $$q_1 = b \times q_2 + 1$$

根据理论，当RSA算法中的p,q是强素数时，安全性较强。

另外，还要保证p,q相差要大，否则根据p-q的差值很容易计算出p+q，进而得到N的分解。

所以执行如下：

```
p = getPrime(1025)
q = getPrime(1024)
```

生成了1025位的大素数p以及1024位的大素数q，这样满足了大小要求，同时也使得p，q的大小相差较大。

1.1.2 e的生成

生成了e之后可以相继确定d，在生成e，d时需要注意以下几点：

> 1. e不能过小，否则可以直接分解。
> 2. d不可过小，需要满足$d >= \frac{1}{3} \times N^{\frac{1}{4}}$否则可以通过连分式理论破解。

e的选择主要有两种，一种是直接选择e=65537， 这样得到的d也较大，另一种是先选择d使之满足第二点的要求，再通过求逆元得到e。

1.1.3 综合

综上得到的p，q，e安全性较高

1.2 测试结果

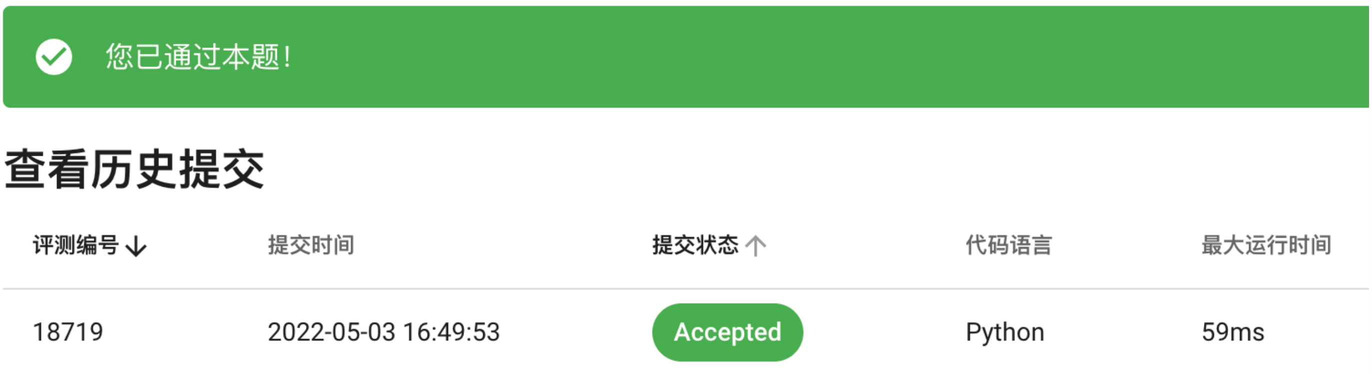

图3 RSA测试结果

1.3 讨论与思考

> 1. 在本节1.1中已经给出了一些提高RSA安全性的措施，如：选择至少1024位的大素数作为p,q；保证p,q是强素数；保证p,q差值较大；保证e不能过小；保证$d >= \frac{1}{3} \times N^{\frac{1}{4}}$；对于一个模数N，不能用两组及以上的公钥加密同一消息，即：选定一个模数N后只生成一对e,d等
> 2. RSA的NP问题就是基于大数分解的困难性，选择大素数首先保证了密钥的安全性，其次更好满足了对于e，d的要求。

2 小指数广播攻击

2.1 算法流程

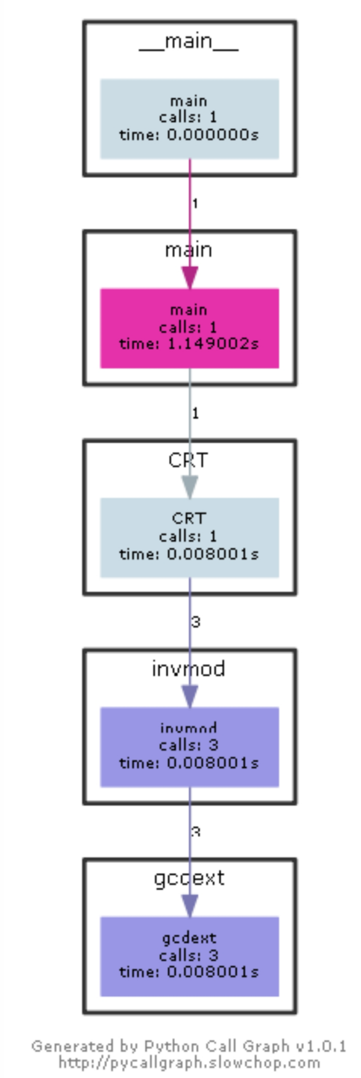

图4 广播攻击的函数调用

加密指数过小，可以尝试对密文c直接进行开n(3,5等等)次根运算，尝试得到明文m。若开根后得不到明文，还可利用中国剩余定理将不同密文（至少两组）组成方程组求解m^n，再进行开根运算，若依然得不到明文m，则换不同的组合依次尝试，直到解出明文m。

本次实验中，我采用将3组密文进行组合，尝试破解，一旦开根运算能得到整数，就证明破解成功。

2.2 测试结果

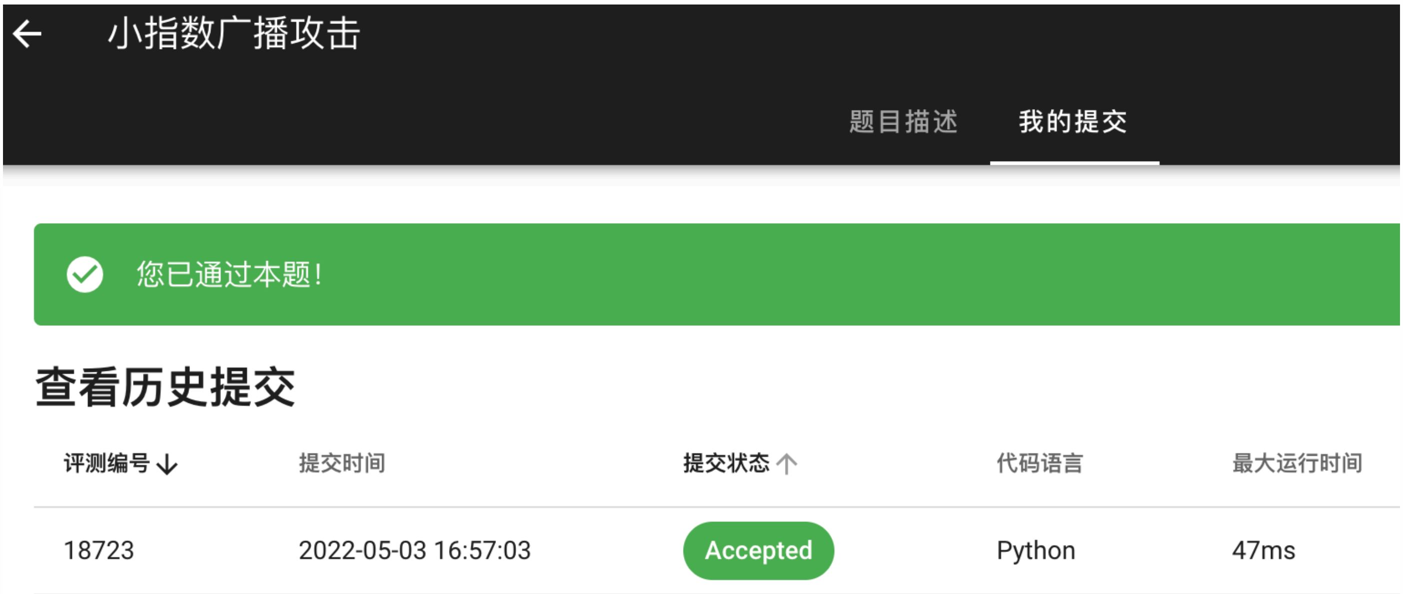

图5 广播攻击测试结果

2.3 讨论与思考

这提醒我们指数e的选择不能太小。

3 共模攻击

3.1 算法流程

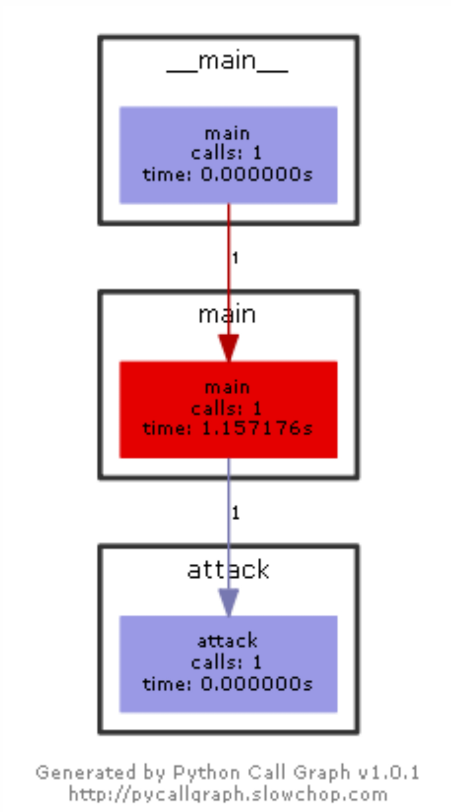

图6 共模攻击流程图

若$(e_1, e_2) = 1$那么可以利用欧几里得算法得到$e_1 \times s_1 + e_2 \times s_2 = 1$所以有

$$m \equiv m^{c_1 \times s_1 + c_2 \times s_2} \equiv c_1^u \times c_2^v \ mod \ N$$

从而得到明文m。

3.2 测试结果

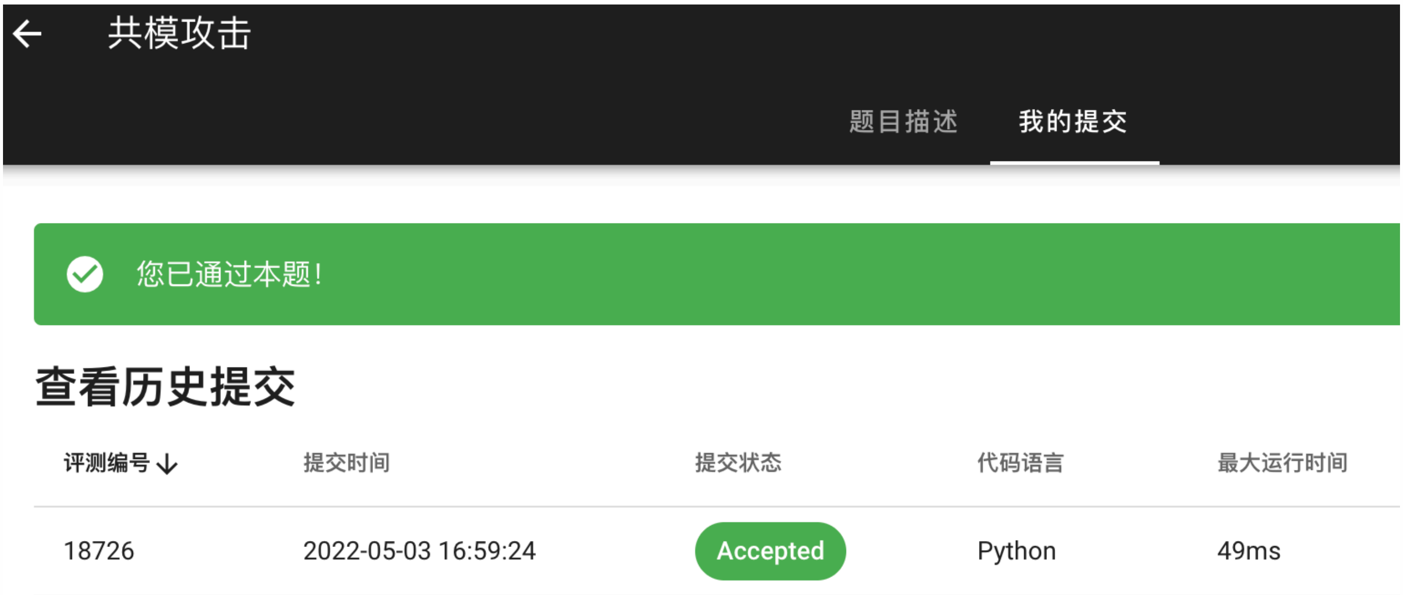

图7 共模攻击测试结果

3.3 讨论与思考

对于一个大整数N不能生成两个不同的公钥e1,e2，更不能用这两个不同的公钥加密同一个消息。

4 已知公私钥分解合数N

4.1 算法流程

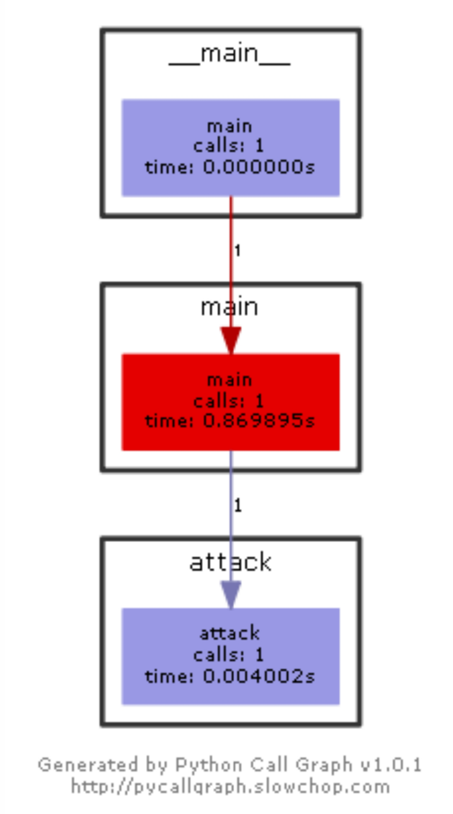

图8 分解合数N函数调用

计算$k=e*d-1$，选一个随机数计g算$g^k\equiv g^{e*d-1}\equiv 1\ mod\ N$，k是偶数，可被分解为：$k=2^t\times r$，且有

$$g^k-1\equiv (g^{\frac{k}{2}}-1)(g^{\frac{k}{2}}+1)\equiv 0\ mod\ n$$

验证$g^{\frac{k}{2}}-1$是否是n的因子，即$n\ mod\ g^{\frac{k}{2}}-1$是否是0。若是，则$g^{\frac{k}{2}}-1$为p,q其一；若不是，继续将$g^{\frac{k}{2}}-1$平方差分解，也即将k除以2，验证$g^{\frac{k}{2^2}}-1$是否是n的因子。以此类推。在k被分解出奇数r前，总能得到某个$g^{\frac{k}{2^l}}-1$为n的因子，也即得到p,q。

4.1.1 算法伪代码

```
算法 2 已知e，d分解N
输入: e, d, n
输出: p, q
function RESOLVE(e, d, n)
    while 1 do
        k = e * d - 1
        g = random(2, n)
        while k mod 2 == 0 do
            k = k//2
            temp = g^k (mod n) - 1
            if gcd(temp, n) > 1  &  temp != 0 then
                p = gcd(temp, n)
                q = n//p
                if p > q then
                    p, q = q, p
                end if
                return p, q
            end if
        end while
    end while
end function
```

图9 分解合数伪代码

4.2 测试结果

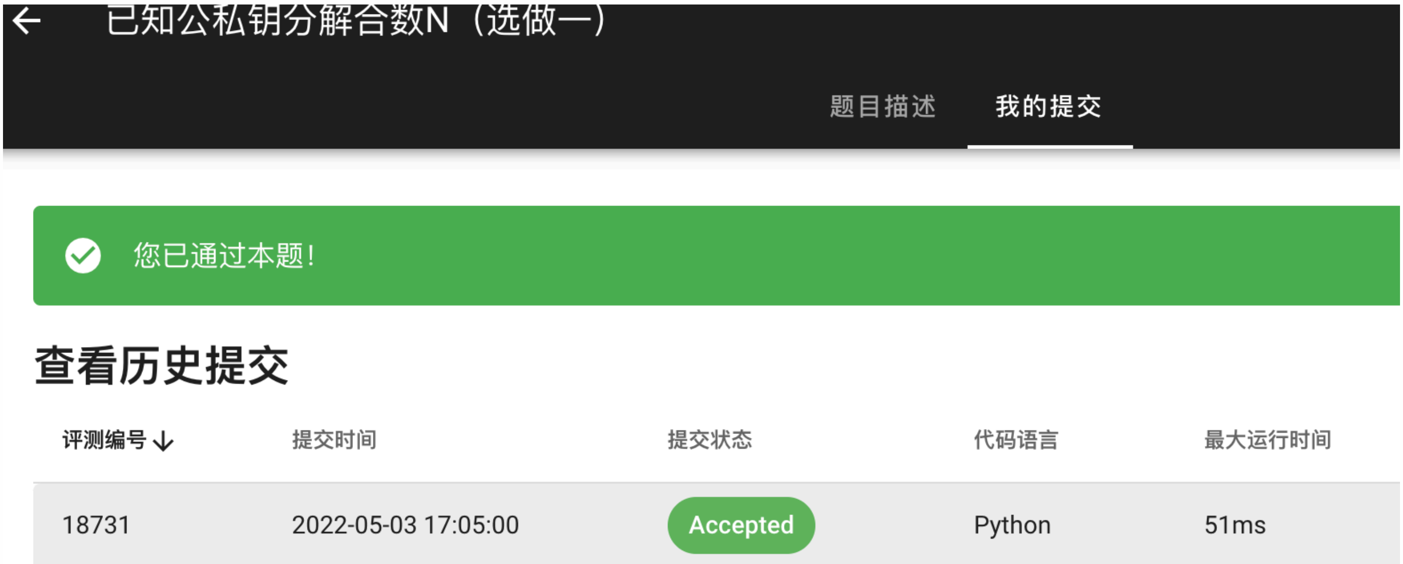

图10 分解合数测试结果

5 维纳攻击

5.1 算法流程

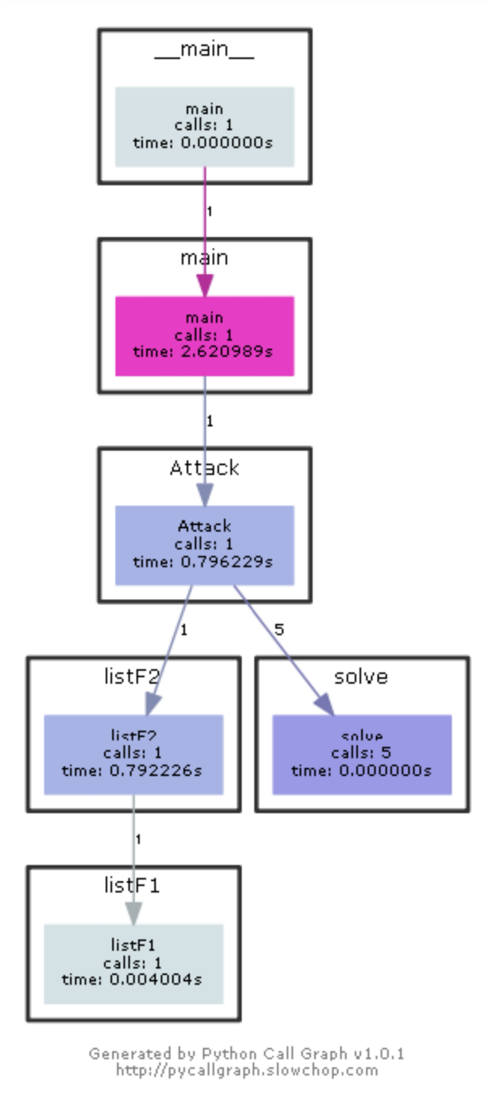

图11 维纳攻击流程图

当d较小满足Wiener攻击的条件时，可利用连分数理论计算出$\frac{e}{n}$的渐进分数，其覆盖了$\frac{d}{k}$，在得到e,d,n,k等值后，便可轻易计算出$\phi(n)$，进而根据n的值建立二次方程p,q的值。

5.1.1 算法伪代码

求渐进分数的伪代码如下

算法 3 求渐进分数list

输入： x, y

输出： ans

```
function LISTF(x, y)
    while y do
        a.append(x//y)
        x, y = y, x mod y
    end while
    ans = []
    for i = 1 → len(a) do
        d, k = 0, 1
        for j = i → 0 do
            d, k = k, a[j] * k + d
        end for
        ans.append((d, k))
    end for
    return ans
end function
```

图12 渐进分数伪代码

算法 4 维纳攻击

输入： e, n

输出： d, p, q

```
function WEINERATTACK(e, n)
    a = listF(e, n)
    for i = 0 → len(a) do
        d, k = a[i]
        if k == 0 then
            continue
        end if
        if (e * d − 1)modk! = 0 then
            continue
        end if
        phi = (e * d − 1)//k
        dlt = sqrt((n − phi + 1)^2 − 4 * n)
        px, qy = (n − phi + 1 + dlt)//2, (n − phi + 1 − dlt)//2
        if px * qy == n then
            p, q = int(px), int(qy)
            d = invmod(e, (p − 1)(q − 1))
            if p > q then
                p, q = q, p
            end if
            return d, p, q
        end if
    end for
end function
```

图13 维纳攻击伪代码

5.1.2 流程图

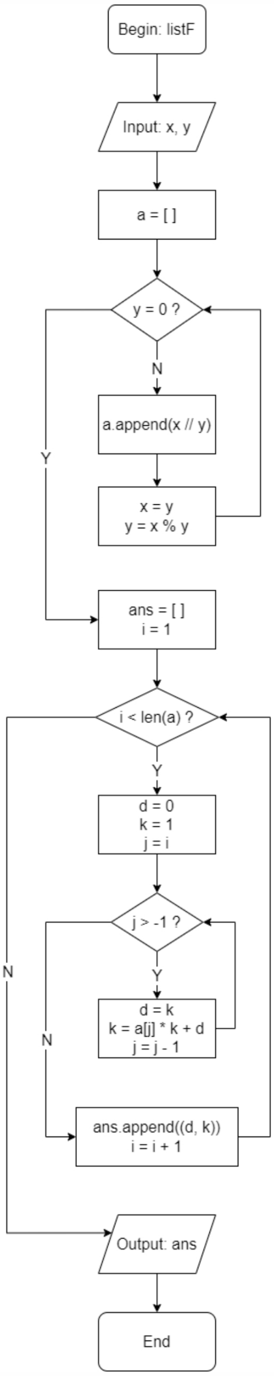

图14 渐进分数流程图

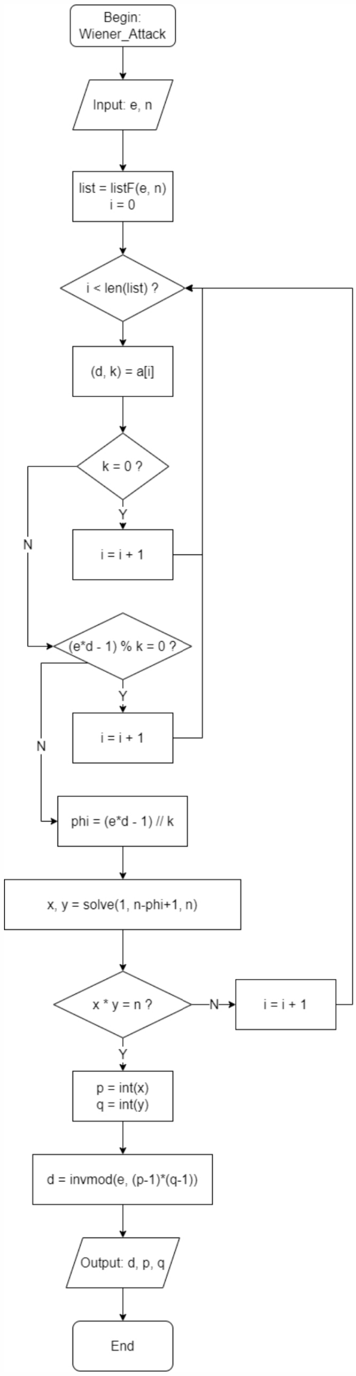

图15 维纳攻击流程图

5.2 测试结果

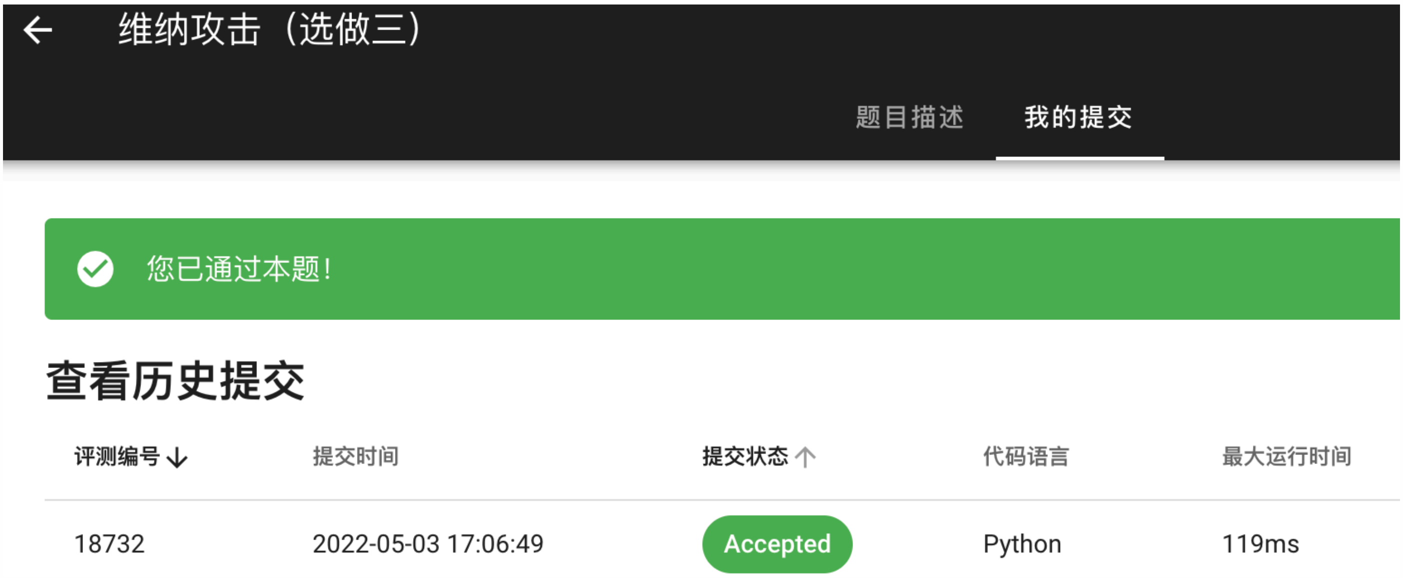

图16 维纳攻击测试结果

6 思考与感悟

选择密文攻击：对于收到的消息$c = m^e \mod N$，截获c后，运算得$x = c * 2^e\ mod\ N$

然后选择密文x解密：

$$y = x^d\ mod\ N = c^d * 2^{e*d}\ mod\ N = (2m)^{ed}\ mod\ N = 2m\ mod\ N$$

从而得到$m = 2^{-1} * y\ mod\ N$完成破解！