Github 账号：sptttt

实验摘要：

RSA 密码算法是使用最为广泛的公钥密码体制。该体制简单且易于实现，RSA 体制破译相当于已知$m^e \bmod n$，能否还原 m 的数论问题。目前模数规模为 1024 比特的 RSA 算法一般情况下是安全的，但是如果参数选取不当，同样存在被破译的可能。

根据 2016 全国高校密码数学挑战赛赛题三，编程实现几种 RSA 体制的攻击方法，破解题目所给密文。

实验题目

RSA 加密体制破译

1.1 问题描述

RSA 密码算法是使用最为广泛的公钥密码体制。该体制简单且易于实现，只需要选择 5 个参数即可（两个素数p和q、模数$N = pq$、加密指数e和解密指数d）。设m为待加密消息，RSA 体制破译相当于已知$m^e \bmod N$，能否还原m的数论问题。目前模数规模为 1024 比特的 RSA 算法一般情况下是安全的，但是如果参数选取不当，同样存在被破译的可能。

有人制作了一个 RSA 加解密软件（采用的 RSA 体制的参数特点描述见密码背景部分）。已知该软件发送某个明文的所有参数和加密过程的全部数据（加密案例文件详见附件 3-1）。Alice 使用该软件发送了一个通关密语，且所有加密数据已经被截获，请问能否仅从加密数据恢复该通关密语及 RSA 体制参数？如能请给出原文和参数，如不能请给出已恢复部分并说明剩余部分不能恢复的理由？

1.2 实例破解

在本次竞赛问题中，我们选取了一个具体加密实例供大家破解，整个算法与加密过程描述如下，截获的加密数据见附件 3-2。

1. RSA 密码算法描述如下，包含体制参数选取和加解密过程。

1）RSA 体制参数选取

Step1. 每个使用者，任意选择两个大素数p和q，并求出其乘积$N = pq$。

Step2. 令$\varphi(N) = (p-1)(q-1)$,选择整数e，使得 GCD$(e, \varphi(N)) = 1$，并求出e模$\varphi(N)$的逆元d，即$ed \equiv 1 \bmod \varphi(N)$.

Step3. 将数对(e, N)公布为公钥，d保存为私钥。

2）加解密过程

Bob 欲传递明文m给 Alice，则 Bob 首先由公开途径找出 Alice 的公钥(e, N)，Bob 计算加密的信息c为：$c \equiv me \bmod N$。

Bob 将密文c传送给 Alice。随后 Alice 利用自己的私钥d解密：

$$c^d \equiv (m^e)^d \equiv m^{ed} \equiv m \bmod N$$

2. Alice 使用的 RSA 密码体制，有以下事项需要说明：

1） 模数$N = pq$规模为 1024 比特，其中p，q为素数；

2） 素数p由某一随机数发生器生成；

3） 素数q可以随机选择，也可以由 2）中的随机数发生器产生；

4） 可以对文本加密，每次加密最多 8 个明文字符；

5） 明文超过 8 个字符时，对明文分片，每个分片不超过 8 个字符；

6） 分片明文填充为 512 比特消息后再进行加密，填充规则为高位添加 64 比特标志位，随后加上 32 比特通信序号，再添加若干个 0，最后 64 比特为明文分片字符对应的 ASCII 码（注：填充方式参见加密案例，但注意每次通信的标志位可能变化）；

7） 分片加密后发送一个加密帧数据，帧数据文件名称为 FrameXX，其中 XX 表示接收序号，该序号不一定等于通信序号；

8） 帧数据的数据格式如下，其中数据都是 16 进制表示，结构如下

1024bit 模数 N | 1024bit 加密指数 e | 1024bit 密文 me mod N。

9） 由于 Alice 初次使用该软件，可能会重复发送某一明文分片。

1.3 成绩评判

通过数论方法获得的原始明文及 RSA 参数数量，数量多者获胜。

实验内容

一 实验基本原理及步骤

1、 实验环境

编程语言：Python 3.7

依赖库： binascii、 gmpy2、 time、 itertools

2、 实验原理及过程

（1） 费马(Fermat)分解法

费马分解法基于如下思想：设 $N = pq$，其中 $p \le q$ 都是奇数，令 $x=\frac{1}{2}(p+q)$，$y=\frac{1}{2}(p-q)$，可以找到 $N=x^2-y^2=(x+y)(x-y)$ 或 $y^2=x^2-N$。

即编写代码，实现从$x=\sqrt{N}$开始增大，直到找到一个数x^2-N是完全平方数，即可计算出$p=x+y$、$q=x-y$，关键代码如下：

```
def Fermat_factorize(input_N, time_limit):    # 费马分解法 分解N得p,q
    start_time = time.time()
    x = gmpy2.iroot(input_N, 2)[0]
    while(time.time()-start_time < time_limit):
        x += 1
        if (gmpy2.iroot(x**2 - input_N, 2)[1] == True):
            y = gmpy2.iroot(x**2 - input_N, 2)[0]
            p = x + y
            q = x - y
            return (p, q)
    return None
```

对 21 个分片中的 N 进行经测试，发现 Frame10 的 N 可在较短时间内被成功分解。

获得了 p、q，可根据密钥生成的方法计算出私钥 d，即先计算$\varphi(N) = (p-1)(q-1)$，求出e模$\varphi(N)$的逆元为d，即$ed \equiv 1 \bmod \varphi(N)$. 获得了 d 即可解密消息：

$$c^d \equiv (m^e)^d \equiv m^{ed} \equiv m \bmod N$$

计算 d 及解密的相关代码如下：

```
def decrypt_by_p_q(n, e, c, p, q):  # 已知 p,q，返回明文 bytes
    phi = (p-1)*(q-1)
    d = gmpy2.invert(e, phi)
    m = gmpy2.powmod(c, d, n)
    return binascii.a2b_hex(hex(m)[2:])
```

获取了明文字节码，再根据题目描述的帧格式恢复出原始明文，关键代码如下：

```
def analyse_ptbytes(ptbytes):  # m 恢复明文分片
    flags = ptbytes[0:8]
    number = int.from_bytes(ptbytes[8:12], 'big')
    massage = ptbytes[-8:]
    return (flags, number, massage)
```

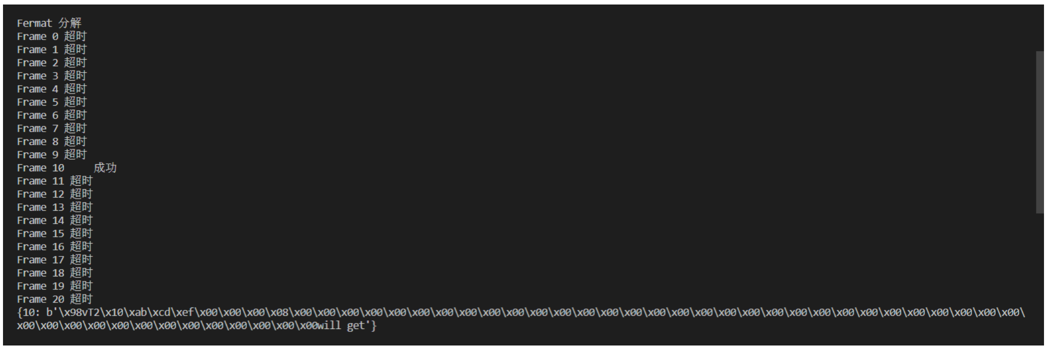

图 1 费马分解 Frame10

（2） Pollard p-1 分解法

设$N = pq$，其中p、q为两个不同素数。选取一个整数k使其满足$(p-1) \mid k!$，根据欧拉定理，对任意与p互素的整数g可以得到$g^{p-1} \equiv 1 (\bmod p)$，因此有$g^{k!} \equiv 1 (\bmod p)$，即$p \mid g^{k!} - 1$；又由于$p \mid n$，因此$p \mid ((g^{k!} \bmod n) - 1)$。

可取$g = 2$，k从 2 开始测试，每次k增加 1，直到$2^{k!} - 1$与n有公因数，即可得出p、q。主要代码如下：

```
def Pollard_p_1(input_N, time_limit):  # Pollard p-1 分解法 分解 N 得 p,q
    start_time = time.time()
    B = 2**20
    a = 2
    for i in range(2, B+1):
        if(time.time()-start_time > time_limit):
            break
        a = pow(a, i, input_N)
        p = gmpy2.gcd(a-1, input_N)
        if (1 < p and p < input_N):
            q = input_N//p
            return (p, q)
    return None
```

对所有分片的 N 进行测试后，成功分解了 Frame2、Frame6 和 Frame19，进而得到相应明文。

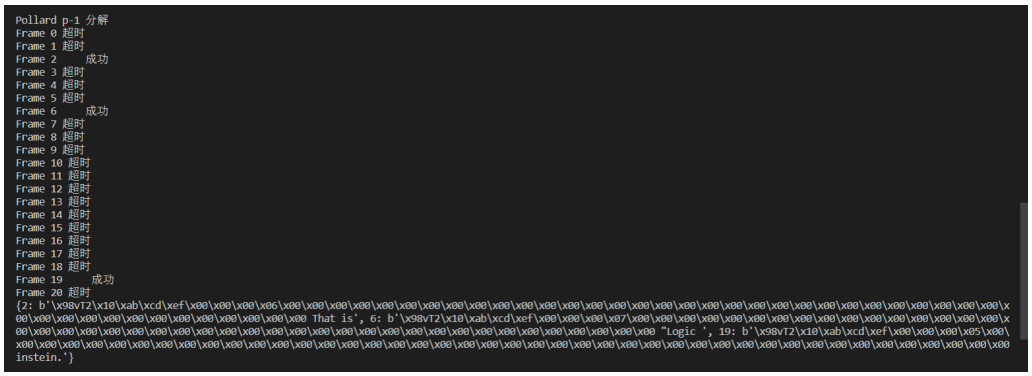

图 2 Pollard p-1 分解 Frame2、6、19

（3） 低加密指数攻击

如果相同的消息使用同样的加密指数加密后发送给不同的接收者，则该明文消息可以被非常有效的恢复。该攻击方法基于中国剩余定理。

假设有三组密文是由同一明文、同一加密密钥加密得到，则根据该定理可以有效地还原其明文。设密钥 $e_1 = e_2 = e_3 = 3$，可列出同余方程组：

$$\begin{cases} c_1 \equiv m^3 \pmod{N_1} \\ c_2 \equiv m^3 \pmod{N_2} \\ c_3 \equiv m^3 \pmod{N_3} \end{cases}$$

应用中国剩余定理可得：

$$\begin{aligned} m^3 \equiv & c_1 \cdot (N_2 \cdot N_3) \cdot \left((N_2 N_3)^{-1} \bmod N_1 \right) \\ & + c_2 \cdot (N_1 \cdot N_3) \cdot \left((N_1 N_3)^{-1} \bmod N_2 \right) \\ & + c_3 \cdot (N_1 \cdot N_2) \cdot \left((N_1 N_2)^{-1} \bmod N_3 \right) \left(\bmod N_1 \cdot N_2 \cdot N_3 \right) \end{aligned}$$

所以 $m = (m^3)^{1/3}$.

编程实现中国剩余定理：

```python
def chinese_remainder_theorem(items):  # 中国剩余定理
    M = 1
    m_list = []
    for (a, m) in items:
        m_list.append(m)
        M *= m
    if(is_relatively_prime(m_list) == False):
        print("不两两互素")
        return None
    result = 0
    for (a, m) in items:
        result += a * (M//m) * gmpy2.invert(M//m, m)
    return (result % M)
```

调用中国剩余定理，编程实现上述过程的求解：

```python
def low_encrypt_e(e, index, length):
  for choice in itertools.combinations(index, length):
      c_m_list = []
      for i in choice:
          c_m_list.append((c[i], N[i]))
      res = chinese_remainder_theorem(c_m_list)
      res = gmpy2.iroot(res, e)
      try:
          return binascii.a2b_hex(hex(res[0])[2:])
      except:
          print("格式错误")
          return None
```

低加密指数攻击法适用于加密密钥相同且模数互素的数，首先编程查找满足条件的帧。

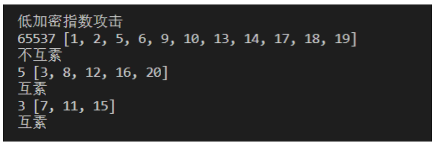

图 3 符合低加密指数攻击的条件的帧

经测试，Frame3、Frame8、Frame12、Frame16、Frame20 和 Frame7、Frame11、Frame15 这两组数分别符合低加密指数攻击的条件。

对于 Frame3、Frame8、Frame12、Frame16、Frame20，发现 m^e 与 N 满足大小关系：

$$|m^3| = 1536 < N \cdot N\,(i = 3,8,12,16,20, j = 3,8,12,16,20)。$$

则只要 5 个分片中有 3 个是由同一明文加密得到的，就符合低加密指数攻击的条件。

调用函数发现 5 个分片均由同一明文加密得到。

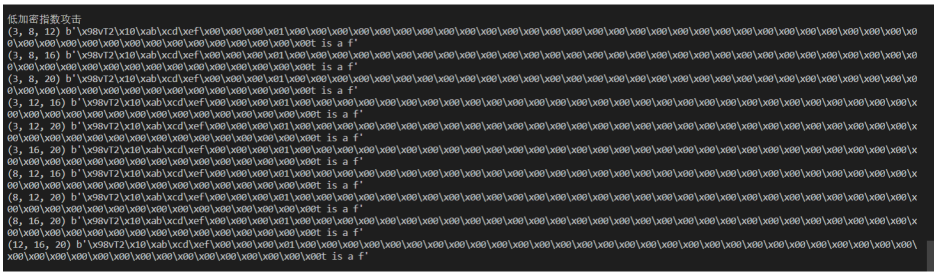

图 4 低加密指数加密——5 个分片

对于 Frame7、Frame11、Frame15，假设有两组是由同一明文加密得到，调用函数求解，发现其在转化式报错，说明解密结果不符合题目要求的格式，即这 3 个帧不是由同样的明文加密得到的。

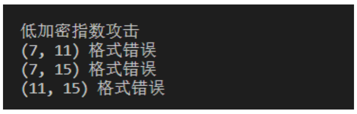

图 5 低加密指数加密——3 个分片

（4） 公共模数攻击

编程查找发现 Frame0 与 Frame4 有相同的模数 N。

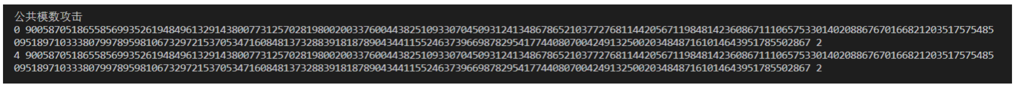

图 6 寻找公共模数

假设他们是由同一明文信息加密得到的，即满足:

$$c_0 \equiv m^{e_0} \bmod N$$

$$c_4 \equiv m^{e_4} \bmod N$$

则可以尝试公共模数攻击法。

应用扩展欧几里德算法的原理，可得到 x、y 使得 $e_0 \cdot x + e_4 \cdot y = 1$，则可解的明文

$$m \equiv c^{x} \cdot c^{y}$$

编程发现解出的明文满足要求的格式，说明这两帧的明文确实相同。

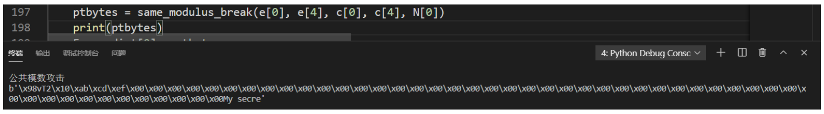

图 7 公共模数攻击

（5） 因数碰撞法

当两个模数不同且不互素时，利用他们的最大公约数即可直接分解模数。编程查找发现帧 1 和 18 的模数不互素，调用最大公约数函数等可实现破解。

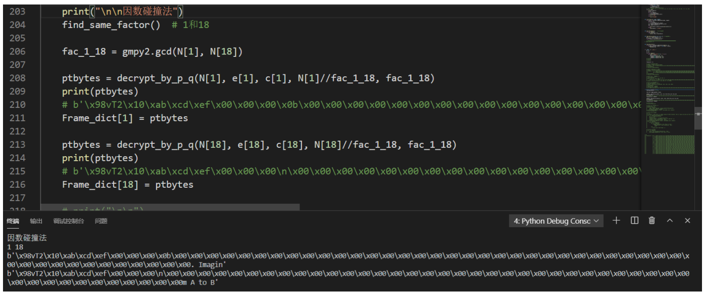

图 8 因数碰撞

（6） 猜测明文

编写函数分析已经获得的所有帧，并按通信序号排序输出。

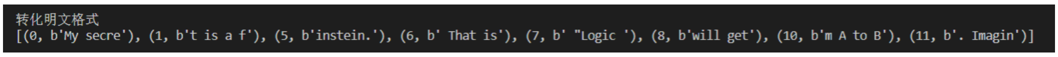

图 9 猜测明文

猜测其明文为：My secret is a famous saying of Albert Einstein. That is "Logic will get you from A to B. Imagination will take you everywhere."，共 128 个字符，恰可分为 16 组，对应于已知的明文分片。

重新分片后，按照题目描述的格式组成帧，对于未破解的帧进行验证：

```python
m = []
m_original = b'My secret is a famous saying of Albert Einstein. That is "Logic will get you from A to B. Imagination will take you everywhere."'
for i in range(len(m_original)//8):
    m_o = m_original[8*i:8*i+8]
    print(i, m_o)
```

```
    # massage_dict[i] = m_original[8*i:8*i+8]
    m_bytes = flags + i.to_bytes(4, "big") + b'\x00'*44 + m_o
    m.append(int.from_bytes(m_bytes, "big"))
# print(sorted(massage_dict.items(), key=lambda item: item[0]))
for i in range(21):
    if(i not in Frame_dict):
        for j in range(16):
            if(gmpy2.powmod(m[j], e[i], N[i]) == c[i]):
                Frame_dict[i] = m[j].to_bytes(64, 'big')
                break
        else:
            print("can't find Frame", i)

# print("\n\n 验证")
for i in range(21):
    m_int = int.from_bytes(Frame_dict[i], 'big')
    assert(gmpy2.powmod(m_int, e[i], N[i]) == c[i])
```

发现未解密出的密文均可对应一个明文分片，进而又验证了所有帧均满足加密关系。至此，所有帧被破解。

二 实验结果

帧号	标志	通信序号	消息
0	b'\x98vT2\x10\xab\xcd\xef'	0	b'My secre'
1	b'\x98vT2\x10\xab\xcd\xef'	11	b'. Imagin'
2	b'\x98vT2\x10\xab\xcd\xef'	6	b' That is'
3	b'\x98vT2\x10\xab\xcd\xef'	1	b't is a f'
4	b'\x98vT2\x10\xab\xcd\xef'	0	b'My secre'
5	b'\x98vT2\x10\xab\xcd\xef'	12	b'ation wi'
6	b'\x98vT2\x10\xab\xcd\xef'	7	b' "Logic '
7	b'\x98vT2\x10\xab\xcd\xef'	2	b'amous sa'
8	b'\x98vT2\x10\xab\xcd\xef'	1	b't is a f'
9	b'\x98vT2\x10\xab\xcd\xef'	13	b'll take '
10	b'\x98vT2\x10\xab\xcd\xef'	8	b'will get'
11	b'\x98vT2\x10\xab\xcd\xef'	3	b'ying of '
12	b'\x98vT2\x10\xab\xcd\xef'	1	b't is a f'
13	b'\x98vT2\x10\xab\xcd\xef'	14	b'you ever'
14	b'\x98vT2\x10\xab\xcd\xef'	9	b' you fro'
15	b'\x98vT2\x10\xab\xcd\xef'	4	b'Albert E'
16	b'\x98vT2\x10\xab\xcd\xef'	1	b't is a f'
17	b'\x98vT2\x10\xab\xcd\xef'	15	b'ywhere.'"
18	b'\x98vT2\x10\xab\xcd\xef'	10	b'm A to B'
19	b'\x98vT2\x10\xab\xcd\xef'	5	b'instein.'
20	b'\x98vT2\x10\xab\xcd\xef'	1	b't is a f'

三 实验结果的分析

1、 使用费马分解、 Pollard p-1 分解、低指数加密攻击、公共模数攻击、因数碰撞法可破解部分帧，再根据明文语义猜测并验证可破解所有帧。

2、 根据通信序号可得全部的明文消息

My secret is a famous saying of Albert Einstein. That is "Logic will get you from A to B. Imagination will take you everywhere."

实验总结

1、规模为 1024 比特的 RSA 算法一般情况下是安全的，但是如果参数选取不当，同样存在被破译的可能。我们在了解了其加密原理后尽可能使用标准库进行加密，不要自己实现加密算法。

2、秘密完全寓于密钥中，加密算法的过程和细节是公开。

3、以后可以考虑使用多个文件的代码结构，使得代码结构更清晰。

4、参考他人已经做的相关工作可以事半功倍。

参考文献

[1]韩旭,李钰汀,张兴隆,等. RSA 加密体制破译报告—双河安团队答题卷[R]. 全国高校密码数学挑战赛, 2016.

[2]blank-vax. RSA_breaking[DB/OL]. (2020-05-11)[2020-12-02]. https://github.com/blank-vax/RSA_breaking.

[3]ba0bao. RSA-Crack[DB/OL]. (2017-11-27)[2020-12-02]. https://github.com/ba0bao/RSA-Crack.