**ElGamal公钥密码算法**

1. 实验目的（包括实验环境、实现目标等等）
2. 实验环境：python3.12.7
3. 实现目标：

本实验旨在用代码编程实现基于Diffie-Hellman密钥交换协议的ElGamal加密算法。我们需要理解并实现大素数和本原根的生成方法,编写程序实现生成一个150位的大素数并且找到其生成元，运用该算法进行加解密运算和验证操作。

二、方案设计

（包括背景、原理、必要的公式、图表、算法步骤等等）

1. 背景：

ElGamal公钥加密算法是基于Diffie-Hellman密钥交换协议的公钥加密算法。该算法由塔希尔·盖莫尔（Taher Elgamal）于1985年提出，并在GnuPG和PGP等许多密码学系统中得到了广泛应用。

1. 原理：

ElGamal公钥加密算法利用大素数和本原根生成公钥和私钥，并且通过这些密钥进行加密和解密操作。

1. 必要公式

1. 生成大素数和本原根：

* 生成一个150位大素数，其形式为，其中也是一个素数。
* 选取一个本原根作为生成元。

2. 密钥生成：

* + 随机选取一个私钥
  + 计算公钥

3. 加密过程：

* + 随机选取一个临时密钥
  + 计算密文对，其中和

4. 解密过程：

* + 使用私钥解密密文对，恢复原文
  + 计算；
  + 计算
  + 验证：

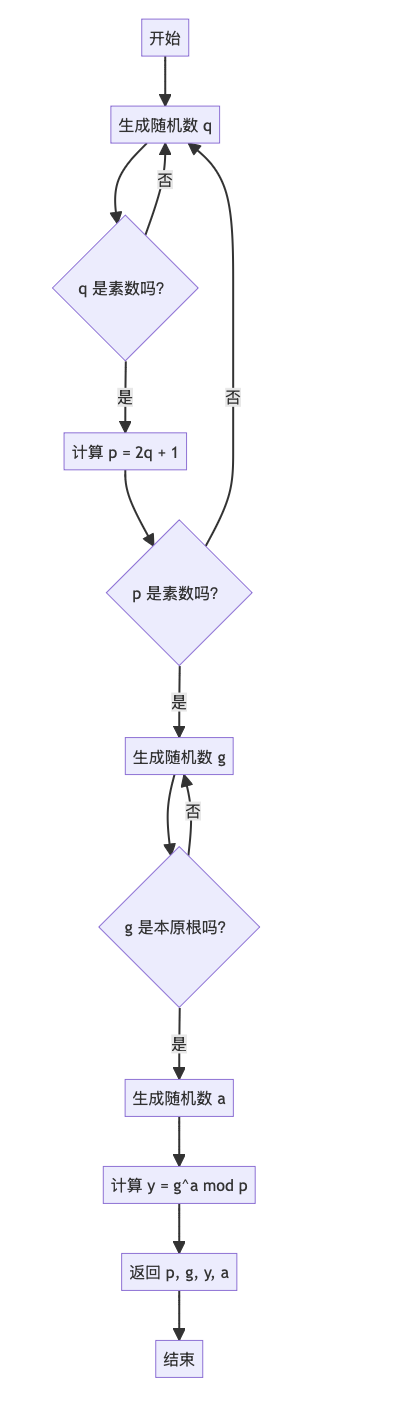
本原元判断公式：

1. 随机生成一个大素数，当为素数时，判断是否为素数。
2. 生成一个随机数，若同时满足且 ，则为大素数的本原根。
3. 算法步骤

首先读取secret文件，将字符型数据转换为整型然后运行gen\_key()函数进行密钥的生成，这里运用本原元判断公式和大素数生成的公式进行生产公钥，以及私钥。其后运行加解密函数然后将解密后的明文和原始明文进行对比，若相同则输出中间值（公钥、私钥）输出明文密文相同，加密解密成功!字样，若不相同输出加密解密失败!字样。

三、方案实现

1. 流程图: 主要算法是密钥生成算法gen\_key()其算法流程图如下：



1. 主要函数介绍

1）密钥生成函数gen\_key()

本函数首先生成150位的大素数p然后运用本原元的判断条件生成合适的g。最后输出生成的密钥：p, g, y, a

def gen\_key():

while True:

q = random.randint(10\*\*149 // 2, 10\*\*150 // 2 - 1)

if gmpy2.is\_prime(q):

p = 2 \* q + 1

if gmpy2.is\_prime(p):

break

a = random.randint(0, p - 1)

while True:

g = random.randint(2, p - 1)

if pow(g, 2, p) != 1 and pow(g, q, p) != 1:

break

y = pow(g, a, p)

return p, g, y, a

2）求逆元函数invmod(a, m)

这里我们运用欧几里得算法进行求逆元运算。

def egcd(a, b):

if a == 0:

return b, 0, 1

else:

g, x, y = egcd(b % a, a)

return g, y - (b // a) \* x, x

def invmod(a, m):

g, x, y = egcd(a, m)

if g == 1:

return x % m

else:

return None

3）加密函数enc(p, g, y, m)

根据加密过程计算密文对，其中和输出随机选择的和密文对，这里需要注意，随机生成k的时候需要保证k与p-1满足互素关系否则会有可能导致的情况，导致不安全。

def enc(p, g, y, m):

while True:

k = random.randint(1, p - 1)

if egcd(k, p - 1)[0] == 1: # k与p-1互素，否则有安全问题

break

c1 = pow(g, k, p)

c2 = m \* pow(y, k, p) % p

return k, c1, c2

4）解密函数dec(p, a, c1, c2)

使用私钥解密密文对，计算然后计算

def dec(p, a, c1, c2):

V = pow(c1, a, p)

m = c2 \* invmod(V, p) % p

return m

5）主函数

用于读取文件并且调用以上生成密钥函数、加解密函数及对结果进行判断。

若解密后的明文和原始明文相同则输出相应的中间值（公钥和私钥），反之则输出加解密失败字样。

if \_\_name\_\_ == '\_\_main\_\_':

f = open('secret4.txt', 'rb')

m = int(f.read())

print(f'明文m = {m}')

p, g, y, a = gen\_key()

k, c1, c2 = enc(p, g, y, m)

m\_de = dec(p, a, c1, c2)

print(f'解密m\_de = {m\_de}')

if m == m\_de:

print(f'p = {p}\ng = {g}\ny = {y}\nk = {k}')

print(f'C = (c1,c2)\n= ({c1},{c2})')

print('明文密文相同，加密解密成功!')

else:

print('加密解密失败!')

f.close()

四、数据分析(包括算法测试数据的分析，运行结果截图等等)

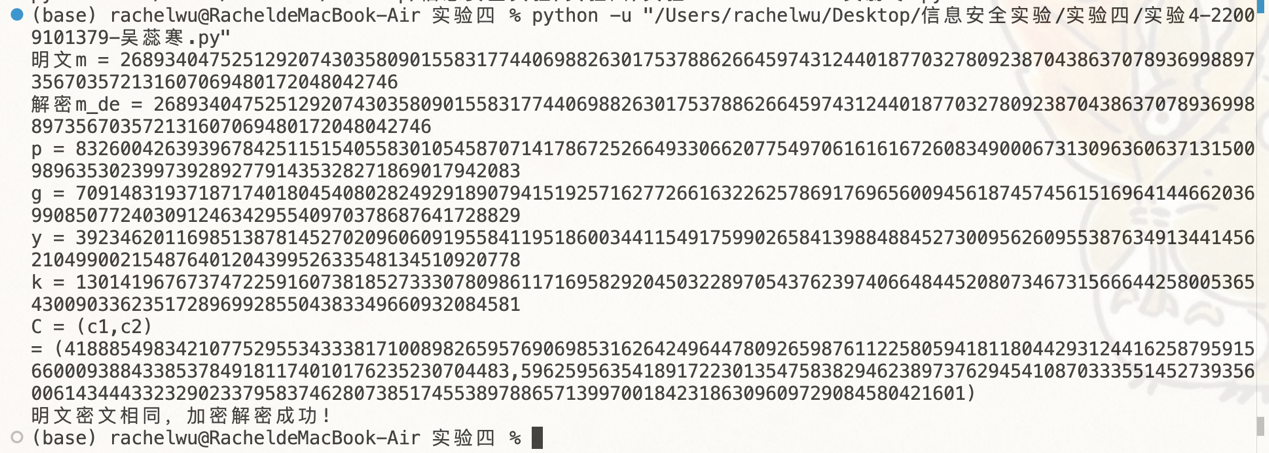
1. 对于secret0.txt



1. 对于secret1.txt



1. 对于secret2.txt



1. 对于secret3.txt



1. 对于secret4.txt



五、思考与总结

1. 请简述什么是本原根，给定素数P，如何求其本原根？。

答：本原根是能够对于一个给定的素数，通过的不同幂次可以生成从到的所有整数。是模的一个本原根，则的幂次 （）生成了从 到的所有整数。给定素数P

1）找到P-1的所有质因数：设 ，其中 是质因数。

2）然后验证候选本原根：

选择一个候选数 ，对于每个质因数 ，计算

如果对于所有的质因数，上述计算结果都不等于1，则 是素数P一个本原根。

1. 如果𝑘与𝑝−1不互素，可能会发生什么情况？

答：若𝑘与𝑝−1不互素，意味着𝑘与𝑝−1有公因数。在ElGamal加密过程中 ，。此时， 的值可能会是1会导致 泄露了明文。

1. 实验过程中还遇到了什么问题，如何解决的？通过该实验有何收获？

答：实验过程中需要保证生成150位的大整数，这里需要这里前面是可以取等号的，起初没有注意，后来补上了。所以的取值是。即。

在确定随机数k的时候仍需要加上判断与p-1不互素以保证安全性。

通过本次实验我能够更加熟练的编写ElGamal加解密程序，并且对求大素数的本原元的数学原理及算法有了更深的理解。