山东大学网络空间安全学院

Python高级程序设计课程实验报告

|  |  |  |  |
| --- | --- | --- | --- |
| 学号：202100460065 | 姓名：李昕 | | 班级：21级密码2班 |
| 实验题目：实验十. 利用math和random模块实现RSA加密算法 | | | |
| 实验学时：2 | | 实验日期：2022/12/1 | |
| 实验目的：熟悉函数的定义与使用，编写代码，完成对自定义函数的编写与调用 | | | |
| 硬件环境：  AMD Ryzen 7 5800H with Radeon Graphics     3.20 GHz  机带 RAM 16.0 GB (13.9 GB 可用) | | | |
| 软件环境：  操作系统：windows 11  编译器：IDLE | | | |
| 实验步骤与内容：  本次实验包括六个内容：  1、编写求最大公因子的函数；  2、编写求模逆的扩展欧几里得算法函数；  3、编写rabin-miller素性检测算法函数；  4、编写生成大素数的算法函数；  5、编写生成RSA公私钥对的函数；  6、编写RSA加密和解密函数。  【求最大公因子】  1、使用辗转相除法（Euclid算法），不断地用两个数中较大的那个除以较小的数取出他们地余数，如果余数为0，则返回较小的数b，如果不为0则不断循环直至余数为0  2、小于b的正整数只有有限个以及1能整除任何一整数，所以循环最终一定会结束。  代码实现：   1. def gcd(a,b):*#辗转相除* 2. if(a%b==0): 3. return b 4. else: 5. return gcd(b,a%b)   【求模逆的扩展欧几里得算法】  1、根据Euclid辗转相除法可知，一定存在x，y使得ax+my=1成立，所以此处只要取c=x就满足要求  2、求模逆得前提条件是（a，m）=1，如果最大公因子不为1，则不存在模逆  代码实现：   1. def findModInverse(a, m): *#扩展欧几里得* 2. if gcd(a, m) != 1: 3. return None *#如果a和m不互质，则不存在模逆* 4. u1, u2, u3 = 1, 0, a 5. v1, v2, v3 = 0, 1, m 6. while v3 != 0: 7. q = u3 // v3 *# //是整数除法运算符* 8. v1 , v2, v3, u1 , u2, u3 = (u1 - q \* v1), (u2 - q \* v2), (u3 - q \* v3), v1 , v2, v3 9. return u1 % m   【rabin-miller素性检测算法函数】  1、算法的理论基础是费马小定理,即设 n 是正奇整数，如果 n 是素数，则对于任意整数 b（1 < b < n-1），（b，n）=1，有 b^(n-1)≡1(mod n)。若 b^(n-1)≡1(mod n) 不成立，则n是合数。  2、但该算法存在伪素数，即若 b^(n-1)≡1(mod n)，n 可能是素数，也可能是合数。若n是奇合数，则至少有一半的b使得式子不成立，其优化为Solovay-Strassen算法，SS算法可以使判断误差降到1/4  代码实现：   1. def rabinMiller(num): *#The Rabin-Miller Primality Test* 2. ''' 3. *Returns True if num is a prime number.* 4. *num -1 = 2^{t}\*s* 5. *或者a^{s}=1 mod num* 6. *或者存在某个i使得a^{2^{t-i}\*s}=1 mod num = num-1* 7. ''' 8. s = num - 1 9. t = 0 10. while s % 2 == 0: 11. *# keep halving s while it is even (and use t to count how many times we halve s)* 12. s = s // 2 13. t += 1 14. for trials in range(5): *# try to falsify num's primality 5 times* 15. a = random.randrange(2, num - 1) 16. v = pow(a, s, num) *#a^{s} mod num* 17. if v != 1: *# this test does not apply if v is 1.* 18. i = 0 19. while v != (num - 1): *#顺便测试了一下a^{s}是否等于-1* 20. if i == t - 1: 21. return False 22. else: 23. i = i + 1 24. v = (v \*\* 2) % num  *#a^{2s},a^{4s},...,a^{2^{t}\*s} mod num* 25. return True 26. def isPrime\_rabin\_miller(num): 27. *# Return True if num is a prime number. This function does a quicker* 28. *# prime number check before calling rabinMiller().* 29. if (num < 2): 30. return False *# 0, 1, and negative numbers are not prime* 31. *#print(primeSieve(1000))* 32. lowPrimes = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 33. 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 34. 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 35. 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 36. 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 37. 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397, 401, 409, 38. 419, 421, 431, 433, 439, 443, 449, 457, 461, 463, 467, 479, 487, 39. 491, 499, 503, 509, 521, 523, 541, 547, 557, 563, 569, 571, 577, 40. 587, 593, 599, 601, 607, 613, 617, 619, 631, 641, 643, 647, 653, 41. 659, 661, 673, 677, 683, 691, 701, 709, 719, 727, 733, 739, 743, 42. 751, 757, 761, 769, 773, 787, 797, 809, 811, 821, 823, 827, 829, 43. 839, 853, 857, 859, 863, 877, 881, 883, 887, 907, 911, 919, 929, 44. 937, 941, 947, 953, 967, 971, 977, 983, 991, 997] 45. if num in lowPrimes: 46. return True 47. *# See if any of the low prime numbers can divide num* 48. for prime in lowPrimes: 49. if (num % prime == 0): 50. return False 51. *# If all else fails, call rabinMiller() to determine if num is a prime.* 52. return rabinMiller(num)   【生成大素数的算法】  1、随机生成一个在2\*\*(keysize -1)到2\*\*(keysize -1)之间的大整数  2、使用rabin-miller素性检测算法判断生成的是否是一个素数  代码实现：   1. def generateLargePrime(keysize=1024): 2. *# Return a random prime number of keysize bits in size.* 3. while True: 4. num = random.randrange(2\*\*(keysize -1), 2\*\*(keysize -1)) 5. if isPrime\_rabin\_miller(num): 6. return num   【生成RSA公私钥对】  选择两个大素数p和q，计算p和q的乘积n，计算p-1和q-1的乘积，选择一个与p-1和q-1乘积互质的数e，计算出d  代码实现：   1. from Crypto.PublicKey import RSA 2. from Crypto.Cipher import PKCS1\_OAEP 3. import time 4. *#生密钥* 5. start = time.time() 6. for i in range(100): key = RSA.generate(1024) 7. end = time.time() 8. print('generatekey time:%d circles using %f seconds'%(100, end-start)) 9. *# 提取私钥并存入文件* 10. private\_key = key.export\_key()   *#提取私钥* 11. file\_out = open("private\_key.pem", "wb") 12. file\_out.write(private\_key) 13. file\_out.close() 14. *# 提取公钥存入文件* 15. public\_key = key.publickey().export\_key() *#提取公钥* 16. file\_out = open("public\_key.pem", "wb") 17. file\_out.write(public\_key) 18. file\_out.close()   【RSA的加密和解密】  1、使用RSA一般需要产生公钥和私钥，当采用公钥加密时，使用私钥解密；采用私钥加密时，使用公钥解密。  2、明文为M，密文为C，公匙PU={e, n}，密匙PR={d, n}  选择两个大素数p和q，计算p和q的乘积n，计算p-1和q-1的乘积，选择一个与p-1和q-1乘积互质的数e，计算出d  解密为：C=M^e mod n 解密为：M=C^d mod n  代码实现：   1. *#加密* 2. data = b"202100460065" 3. public\_key = RSA.import\_key(open("public\_key.pem").read()) *#读取公钥* 4. cipher = PKCS1\_OAEP.new(public\_key) *# 实例化加密套件* 5. encrypted\_data = cipher.encrypt(data) *# 加密* 6. *# 将加密后的内容写入到文件* 7. file\_out = open("encrypted\_data.bin", "wb") 8. file\_out.write(encrypted\_data) 9. file\_out.close() 10. *#编写RSA加密和解密函数。* 11. private\_key = RSA.import\_key(open("private\_key.pem", "rb").read()) *#读取私钥* 12. cipher = PKCS1\_OAEP.new(private\_key) *# 实例化加密套件* 13. encrypted\_data = open("encrypted\_data.bin", "rb").read() *# 读取加密内容* 14. data = cipher.decrypt(encrypted\_data) *# 解密* 15. print(data) | | | |
| 结论分析与体会：  通过本次实验，我熟悉了python的密码学编程方面的应用，动手实现了简单的RSA有关代码，加深了对math和random模块的理解，也熟练了密码学所用到的数学知识。 | | | |