

天津大学本科生实验报告专用纸

学院 智能与计算学部 年级 2017 专业 软件工程 班级 2班 姓名 侯雨茜 学号 3017218092

课程名称 信息安全技术 实验日期 2019-10-25 成绩

同组实验者

实验 2：RSA 加密算法实现

一、实验目的

实现 RSA 加密算法。

二、实验环境

1. 操作系统：Mac OS
2. 实现语言：Python

三、实验内容

1. 利用 Miller-Rabin 测试生成两个 256（或 512）比特的素数，记录素数生成的时间。
2. 计算 $n = pq$. 利用 Euclidean 算法生成与 $\Phi(n)$ 互素的整数作为私钥，利用扩展 Euclidean 算法计算 $b = a^{-1}$ 作为公钥。
3. 随机生成明文 $m \in Z_n$ ，利用平方乘积（Square and Multiply）算法计算密文 $c = m^b \bmod n$ ，并进行解密运算 $m' = c^a \bmod n$ ，检验解密的正确性，即 $m = m'$ 是否成立。
4. 记录一组明文的 RSA 加密和解密时间。使用 DES 对相同的明文进行加解密，记录 DES 加密和解密时间。（建议选取 10 组以上明文，对比这些明文加密时间总和。）
5. RSA 算法正确性检验。

四、实验步骤

4.1 算法代码如下。

Prime.py 文件：

```
1 import math
2 import random
3
4
5 # 扩展欧几里得算法求模反元素
6 def ex_euclid(a, b, list):
7     if b == 0:
8         list[0] = 1
9         list[1] = 0
10        list[2] = a
11    else:
12        ex_euclid(b, a % b, list)
13        temp = list[0]
14        list[0] = list[1]
15        list[1] = temp - a // b * list[1]
```

天津大学本科生实验报告专用纸

```
18 # 求模反元素
19 def mod_inverse(a, b):
20     list = [0, 0, 0]
21     if a < b:
22         a, b = b, a
23     ex_euclid(a, b, list)
24     if list[1] < 0:
25         list[1] = a + list[1]
26     return list[1]
27
28
29 # 快速幂模运算，把b拆分为二进制，遍历b的二进制，当二进制位为0时不计入计算
30 def quick_pow_mod(a, b, c):
31     a = a % c
32     ans = 1
33     while b != 0:
34         if b & 1:
35             ans = (ans * a) % c
36             b >>= 1
37             a = (a % c) * (a % c)
38     return ans
39
40
41 # n为要检验的大数，a < n, k = n - 1
42 def miller_rabin_witness(a, n):
43     if n == 1:
44         return False
45     if n == 2:
46         return True
47     k = n - 1
48     q = int(math.floor(math.log(k, 2)))
49     while q > 0:
50         m = k // 2 ** q
51         if k % 2 ** q == 0 and m % 2 == 1:
52             break
53         q = q - 1
54     if quick_pow_mod(a, n - 1, n) != 1:
55         return False
56     b1 = quick_pow_mod(a, m, n)
57     for i in range(1, q + 1):
58         if b1 == n - 1 or b1 == 1:
59             return True
60         b2 = b1 ** 2 % n
61         b1 = b2
62     if b1 == 1:
63         return True
64     return False
```

教师签字：
年 月 日

```

67 # Miller-Rabin素性检验算法,检验8次
68 def prime_test_miller_rabin(p, k):
69     while k > 0:
70         a = random.randint(1, p - 1)
71         if not miller_rabin_witness(a, p):
72             return False
73         k = k - 1
74     return True
75
76 # 判断 num 是否与 prime_arr 中的每一个数都互质
77 def prime_each(num, prime_arr):
78     for prime in prime_arr:
79         remainder = num % prime
80         if remainder == 0:
81             return False
82     return True
83
84 # return a prime array from begin to end
85 def get_con_prime_array(begin, end):
86     array = []
87     for i in range(begin, end):
88         flag = judge_prime(i)
89         if flag:
90             array.append(i)
91     return array
92
93 # judge whether a number is prime
94 def judge_prime(number):
95     temp = int(math.sqrt(number))
96     for i in range(2, temp + 1):
97         if number % i == 0:
98             return False
99     return True
100
101 # 根据 count 的值生成若干个与质数数组都互质的大数
102 def get_rand_prime_arr(count):
103     arr = get_con_prime_array(2, 100000)
104     prime = []
105     while len(prime) < count:
106         num = random.randint(pow(2, 255), pow(2, 256))
107         if num % 2 == 0:
108             num = num + 1
109         while True:
110             if prime_each(num, arr) and prime_test_miller_rabin(num, 8):
111                 if num not in prime:
112                     prime.append(num)
113                     break
114             num = num + 2
115     return prime

```

RSA.py 文件:

```

1 import random
2 import Prime
3 import time
4
5 # encryption ,according to the formula:m^e = c (mod n) , calculate c ,c == secret,m == m
6 def encryption(plaintext, puk):
7     return Prime.quick_pow_mod(plaintext, puk[1], puk[0])
8
9 # decryption ,according to the formula:c^d = m (mod n) , calculate m ,
10 def decryption(ciphertext, prk):
11     return Prime.quick_pow_mod(ciphertext, prk[1], prk[0])

```

```

16 def get_RSAKey():
17     RSAKey = {}
18
19     start = time.perf_counter()
20     prime_arr = Prime.get_rand_prime_arr(2)
21     p = prime_arr[0]
22     q = prime_arr[1]
23     while p == q:
24         q = random.choice(prime_arr)
25     end = time.perf_counter()
26
27     n = p * q
28     s = (p - 1) * (q - 1)
29     a = 65537
30     b = Prime.mod_inverse(a, s)
31
32     print("随机生成的素数p = ", p)
33     print("随机生成的素数q = ", q)
34     print('素数生成的时间为: ', end - start, 's')
35
36     print("n = pq = ", n)
37     print("利用Euclidean算法生成的私钥a = ", a)
38     print("利用扩展Euclidean算法生成的公钥b = ", b)
39
40     puk = [n, a]
41     prk = [n, b]
42     RSAKey['puk'] = puk
43     RSAKey['prk'] = prk
44     return RSAKey

```

```

47 if __name__ == '__main__':
48     RSAKey = get_RSAKey()
49     # print("Enter a number less and shorter than ", len(str(RSAKey['puk'][0])), ", ", RSAKey['puk'][0])
50     print('请输入明文: ')
51     # only encrypt a number type
52     m = int(input())
53     c = encryption(m, RSAKey['puk'])
54     print("RSA加密后的密文为:", c)
55     # print(len(str(c)))
56     m1 = decryption(c, RSAKey['prk'])
57     print("RSA解密后的密文为:", m1)
58     # print(len(str(m1)))
59     # 检验解密的正确性
60     if m == m1:
61         print('解密成功')
62     # 记录10组明文的RSA加密和解密时间
63     sumEnTime = 0
64     sumDeTime = 0
65     for i in range(10):
66         print('请输入明文: ')
67         # only encrypt a number type
68         m = int(input())
69
70         # RSA加密
71         start = time.perf_counter()
72         c = encryption(m, RSAKey['puk'])
73         end = time.perf_counter()
74         sumEnTime = sumEnTime + (end - start)
75         print("RSA加密后的密文为:", c)
76         # print(len(str(c)))
77
78         # RSA解密
79         start = time.perf_counter()
80         m1 = decryption(c, RSAKey['prk'])
81         end = time.perf_counter()
82         sumDeTime = sumDeTime + (end - start)
83         print("RSA解密后的密文为:", m1)
84     print('10组明文的RSA加密时间为: ', sumEnTime)
85     print('10组明文的RSA解密时间为: ', sumDeTime)

```

4.2 利用 Miller-Rabin 测试生成两个 256（或 512）比特的素数，记录素数生成的时间。

测试结果如下：

```
随机生成的素数p = 95774910269881591502996635900512935208573012503071012106153507813367363892691
随机生成的素数q = 95351721313296420393677056213518656825529237863167943342184121657382959809863
素数生成的时间为： 0.22799048700000002 s
```

随机生成的素数 p =

95774910269881591502996635900512935208573012503071012106153507813367363892691

随机生成的素数 q =

95351721313296420393677056213518656825529237863167943342184121657382959809863

素数生成的时间为： 0.22799048700000002 s

4.3 计算 $n = pq$ 。利用 Euclidean 算法生成与 $\phi(n)$ 互素的整数 a 作为私钥，利用扩展 Euclidean 算法计算 $b = a^{-1}$ 作为公钥。

测试结果如下：

```
n = pq = 913230255285972076710922706627643906447814363844756644344001770567509024199135611980172139456244163
利用Euclidean算法生成的私钥a = 65537
利用扩展Euclidean算法生成的公钥b = 8285080380325318791675134865489696468509040485697803653319699295456365724980
```

n = pq =

913230255285972076710922706627643906447814363844756644344001770567509024199135611980172

1394562441636135110291587167413779920820227463887675101476895411333

利用 Euclidean 算法生成的私钥 a = 65537

利用扩展 Euclidean 算法生成的公钥 b =

828508038032531879167513486548969646850904048569780365331969929545636572498083063000926

0538954266320276686888788480420275763717269484849154875354529107053

4.4 随机生成明文 $m \in \mathbb{Z}_n$ ，利用平方乘积（Square and Multiply）算法计算密文 $c = m^b \bmod n$ ，并进行解密运算 $m' = c^a \bmod n$ ，检验解密的正确性，即 $m = m'$ 是否成立。

测试结果如下：

```
请输入明文：
7345874874086706
RSA加密后的密文为： 3899893164165662650305062235305811581694247742952136769146324534199029087754317272019098344
RSA解密后的密文为： 7345874874086706
解密成功
```

请输入明文：

7345874874086706

RSA 加密后的密文为:

389989316416566265030506223530581158169424774295213676914632453419902908775431727201909

8344422525827117140346252602799872341380795588940691844422411975878

RSA 解密后的密文为: 7345874874086706

解密成功

4.5. 记录一组明文的 RSA 加密和解密时间。使用 DES 对相同的明文进行加解密，记录 DES 加密和解密时间。（建议选取 10 组以上明文，对比这些明文加密时间总和。）

选取的 10 组明文为：

0123456789012345

0123456789123456

0123456789234567

0123456789345678

0123456789456789

0123456789567890

1234567890123456

1234567891234567

1234567892345678

1234567893456789

测试结果如下：

RSA 加密时间总和为: 0.0006894260000258612 s

RSA 解密时间总和为: 0.03216246800001521 s

DES 加密时间总和为: 0.10484764400002611 s

RSA 测试结果:

```
RSA x
请输入明文：
0123456789012345
RSA加密后的密文为： 114687876363034004782790208369984919381391465477906530540153853899499939823145618809479623
RSA解密后的密文为： 123456789012345
请输入明文：
01234567890123456
RSA加密后的密文为： 233129785045206376108098106721957645709968148666646168796436942470077142228913412080292141
RSA解密后的密文为： 123456789123456
请输入明文：
012345678901234567
RSA加密后的密文为： 153944996793389058384647425339826631745616746582892599866445483162971900293671427166505494
RSA解密后的密文为： 123456789234567
请输入明文：
0123456789012345678
RSA加密后的密文为： 266324515956654687453185823685921874286582041418054277828365979508972344201228371862359312
RSA解密后的密文为： 123456789345678
请输入明文：
01234567890123456789
RSA加密后的密文为： 269614864471275877527526332108982517517488529858251328794594988340706523845035455989822814
RSA解密后的密文为： 123456789456789
请输入明文：
012345678901234567890
RSA加密后的密文为： 336845832790293312973424803146439221598177303773365398321845755840680874691687404813001354
RSA解密后的密文为： 123456789567890
请输入明文：
1234567890123456
RSA加密后的密文为： 113353300233906447222030742950472994191957007198350799902655878835983378629853228599525402
RSA解密后的密文为： 1234567890123456
请输入明文：
12345678901234567
RSA加密后的密文为： 155088960386067425256517494471492560155119577321204150637235333707086180039695686618312243
RSA解密后的密文为： 1234567891234567
请输入明文：
123456789012345678
RSA加密后的密文为： 134965145424296257101378169776885596156774587036977773108979301188089168990312097870930232
RSA解密后的密文为： 1234567892345678
请输入明文：
1234567890123456789
RSA加密后的密文为： 382418567424360779409891214790885546252352869000361759872801021578046202295178710960006331
RSA解密后的密文为： 1234567893456789
10组明文的RSA加密时间为： 0.0006894260000258612
10组明文的RSA解密时间为： 0.03216246800001521
Process finished with exit code 0
```

DES 测试结果:

```
/Users/yuqianhou/Documents/Software_Engineering/信息安全技术/实验/DES/DES_py/venv/bin/python /Users/yuqianhou/Documents/Sof
密钥k = 4F6556C45580DAD9
请输入16进制明文x:
0123456789012345
加密后的密文z为: 73E22465B140101F
请输入16进制明文x:
0123456789123456
加密后的密文z为: 4FD52DD09A56793F
请输入16进制明文x:
012345678901234567
加密后的密文z为: E7222D1B4BC3BC76
请输入16进制明文x:
0123456789012345678
加密后的密文z为: F1BB9D9102F67A0D
请输入16进制明文x:
01234567890123456789
加密后的密文z为: 124D54093BDD32C3
请输入16进制明文x:
012345678901234567890
加密后的密文z为: 10FB0E0950FC8CDF
请输入16进制明文x:
1234567890123456
加密后的密文z为: CA11669E214605AE
请输入16进制明文x:
12345678901234567
加密后的密文z为: 79193ADAE2200C6E
请输入16进制明文x:
123456789012345678
加密后的密文z为: 5290163AA7B0C6DD
请输入16进制明文x:
1234567890123456789
加密后的密文z为: 01A977237230520E
加密10组明文所用时间为: 0.10484764400002611
Process finished with exit code 0
```

4.6. RSA 算法正确性检验。

检验通过。

五、结论

通过本次实验的实践和学习，我掌握了使用 python 语言编写 RSA 加密算法的方法。同时，我还在对算法的测试过程中发现,RSA 算法生成素数和加密的速度都是非常快的,加密一组数据仅需要不到 $7 * 10^{-5}$ 秒的时间。在与 DES 加密算法的对比过程中发现，虽然 DES 加密算法的速度也很快，加密一组数据仅需要 0.01s，但是这还远远比 RSA 算法慢了许多。