

# readme

倪浚桐

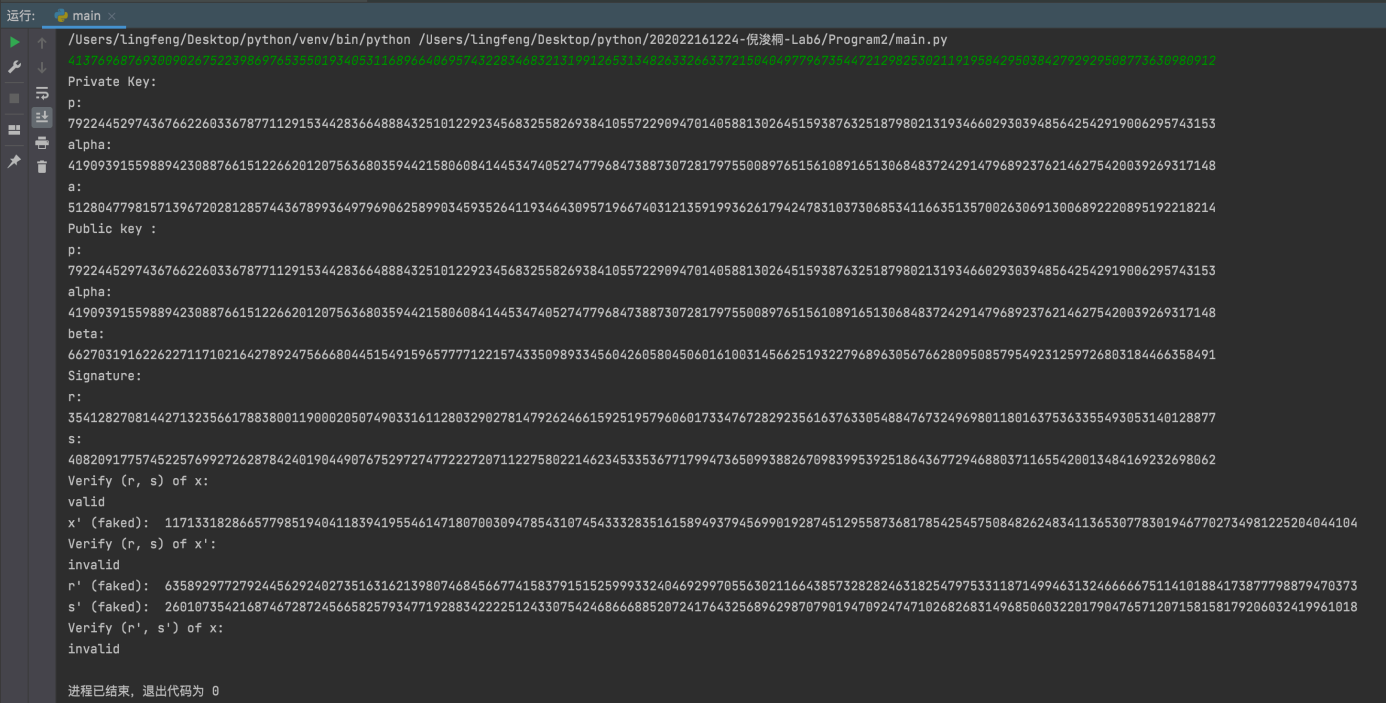
202022161224

Lab6-progarm2

macOS Monterey 12.0.1

Pycharm 11.0.12 x86-64

Python 3.9.5



```
1 import random
2
3
4 # 求最大公约数
5 def gcd(a, b):
6     if a < b:
7         return gcd(b, a)
8     elif a % b == 0:
9         return b
10    else:
11        return gcd(b, a % b)
12
13
14 # 快速幂+取模
```

```

15 def power(a, b, c):
16     ans = 1
17     while b != 0:
18         if b & 1:
19             ans = (ans * a) % c
20             b >>= 1
21             a = (a * a) % c
22     return ans
23
24
25 # 卢卡斯-莱墨素性检验
26 def Lucas_Lehmer(num) -> bool: # 快速检验 $\text{pow}(2, m) - 1$ 是不是素数
27     if num == 2:
28         return True
29     if num % 2 == 0:
30         return False
31     s = 4
32     Mersenne = pow(2, num) - 1 #  $\text{pow}(2, \text{num}) - 1$ 是梅森数
33     for x in range(1, (num - 2) + 1): # num-2是循环次数, +1表示右区间开
34         s = ((s * s) - 2) % Mersenne
35     if s == 0:
36         return True
37     else:
38         return False
39
40
41 # 大素数检测
42 def Miller_Rabin(n):
43     a = random.randint(2, n - 2) # 随机第选取一个 $a \in [2, n-2]$ 
44     # print("随机选取的a=%lld\n"%a)
45     s = 0 # s为d中的因子2的幂次数。
46     d = n - 1
47     while (d & 1) == 0: # 将d中因子2全部提取出来。
48         s += 1
49         d >>= 1
50
51     x = power(a, d, n)
52     for i in range(s): # 进行s次二次探测
53         newX = power(x, 2, n)
54         if newX == 1 and x != 1 and x != n - 1:
55             return False # 用二次定理的逆否命题, 此时n确定为合数。
56         x = newX
57
58     if x != 1: # 用费马小定理的逆否命题判断, 此时 $x = a^{(n-1)} \pmod{n}$ , 那么n确定为合数。
59         return False
60
61     return True # 用费马小定理的逆命题判断。能经受住考验至此的数, 大概率为素数。
62
63

```

```

64 # 扩展的欧几里得算法,  $ab=1 \pmod m$ , 得到a在模m下的乘法逆元b
65 def Extended_Eulid(a: int, m: int) -> int:
66     def extended_eulid(a: int, m: int):
67         if a == 0: # 边界条件
68             return 1, 0, m
69         else:
70             x, y, gcd = extended_eulid(m % a, a) # 递归
71             x, y = y, (x - (m // a) * y) # 递推关系, 左端为上层
72             return x, y, gcd # 返回第一层的计算结果。
73     # 最终返回的y值即为b在模a下的乘法逆元
74     # 若y为复数, 则y+a为相应的正数逆元
75
76     n = extended_eulid(a, m)
77     if n[1] < 0:
78         return n[1] + m
79     else:
80         return n[1]
81
82
83 # 生成域参数p, 长度大约为512bits
84 def Make_p() -> int:
85     a = random.randint(10 ** 150, 10 ** 160)
86     while gcd(a, 2) != 1:
87         a = random.randint(10 ** 150, 10 ** 160)
88     return a
89
90
91 # 生成域参数alpha
92 def Make_alpha(p: int) -> int:
93     return random.randint(2, p)
94
95
96 # 生成一个小于p的素数作为私钥, 长度大约为512bits
97 def Make_private_key(p: int) -> int:
98     pri = random.randint(2, p - 2)
99     while gcd(pri, p) != 1:
100         pri = random.randint(2, p - 2)
101     return pri
102
103
104 # 快速幂
105 def qpow(a: int, b: int) -> int:
106     ans = 1
107     while b != 0:
108         if b & 1:
109             ans = ans * a
110             b >>= 1
111         a = a * a
112     return ans

```

```

113
114
115 def Make_prime(key_size: int) -> int:
116     while True:
117         num = random.randrange(qpow(2, key_size - 1), qpow(2, key_size))
118         if Miller_Rabin(num):
119             return num
120
121
122 # 计算签名
123 def Sign(x, p, alpha, d) -> []:
124     temp_key = random.randint(0, p - 2)
125     while gcd(temp_key, p - 1) != 1:
126         temp_key = random.randint(0, p - 2)
127     r = power(alpha, temp_key, p)
128     s = (x - d * r) * Extended_Eulid(temp_key, p - 1) % (p - 1)
129     return r, s
130
131
132 # 签名验证
133 def Verify(x, p, alpha, beta, r, s):
134     t = (power(beta, r, p) * power(r, s, p)) % p
135     if t == power(alpha, x, p):
136         return True
137     else:
138         return False
139
140
141 if __name__ == '__main__':
142     x = int(input())
143     if type(x) != int:
144         raise ValueError("Must be an integer!")
145
146     p = Make_prime(512)
147     alpha = Make_alpha(p)
148     a = Make_private_key(p)
149     beta = power(alpha, a, p)
150
151     r, s = Sign(x, p, alpha, a)
152     Valid = Verify(x, p, alpha, beta, r, s)
153
154     r_fake = random.randint(10 ** 150, 10 ** 160)
155     s_fake = random.randint(10 ** 150, 10 ** 160)
156     x_fake = random.randint(10 ** 150, 10 ** 160)
157
158     print("Private Key: ")
159     print("p:")
160     print(p)
161     print("alpha:")

```

```
162     print(alpha)
163     print("a:")
164     print(a)
165     print("Public key : ")
166     print("p:")
167     print(p)
168     print("alpha:")
169     print(alpha)
170     print("beta:")
171     print(beta)
172     print("Signature: ")
173     print("r:")
174     print(r)
175     print("s:")
176     print(s)
177     print("Verify (r, s) of x: ")
178     if Verify(x, p, alpha, beta, r, s):
179         print("valid")
180     else:
181         print("invalid")
182     print("x' (faked): ", x_fake)
183     print("Verify (r, s) of x': ")
184     if Verify(x_fake, p, alpha, beta, r_fake, s_fake):
185         print("valid")
186     else:
187         print("invalid")
188
189     print("r' (faked): ", r_fake)
190     print("s' (faked): ", s_fake)
191     print("Verify (r', s') of x: ")
192     if Verify(x, p, alpha, beta, r_fake, s_fake):
193         print("valid")
194     else:
195         print("invalid")
196
```