**广州大学学生实验报告**

**开课学院及实验室：**网络空间安全学院 **2023年 6月 6日**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **学院** | 网络空间安全学院 | **年级/专业/班** | 网安211 | **姓名** |  | **学号** |  |
| **实验课程名称** | 密码技术及应用 | | | | | **成绩** |  |
| **实验项目名称** | 安全多方计算实验 | | | | | **指导**  **老师** | 李树栋 |

1. 实验目的
2. 用Shamir门限方案实现秘密共享
3. 百万富翁问题**（选做）**
4. 实验内容
5. 在Shamir秘密共享门限方案中，假设秘密S=54321，选取7个人保管秘密，只要有4个人就能重构秘密。选取参数p=727，并随机生成S1~S3，私钥x1~x7分别为[628, 635, 55, 295, 502, 683, 105]。请计算他们各自的公钥yi，并分两次每次选取任意四个成员进行秘密还原。
6. 随机选取两个百万级别的数字，并使用多方安全计算方法，在不泄露原本数字的情况下比较他们的大小**（选做）**。
7. 实验过程及结果

3.1：

import random

def get\_inverse(value, p):  # 求逆

    for i in range(1, p):

        if (i \* value) % p == 1:

            return i

    return -1

s = 321

n = 7

t = 4

x = [628, 635, 55, 295, 502, 683, 105]

p = 727

sm = []

y = []

for i in range(t - 1):

    si = random.randint(1, p)

    sm.append(si)

for i in range(n):  # 求yi

    tmp = 1

    yi = s

    for j in range(t - 1):

        yi = (yi + sm[j] \* pow(x[i], tmp, p)) % p

        tmp += 1

    y.append(yi)

print("x:", x)

print("y:", y)

peo = []

for i in range(n):

    peo.append([x[i], y[i]])

# 秘密恢复

for i in range(2):

    a = random.sample(peo, t)

    print("随机选取", t, "个人", a)

    st = 0

    for k in range(t):

        t1 = 1

        for j in range(t):

            if j != k:

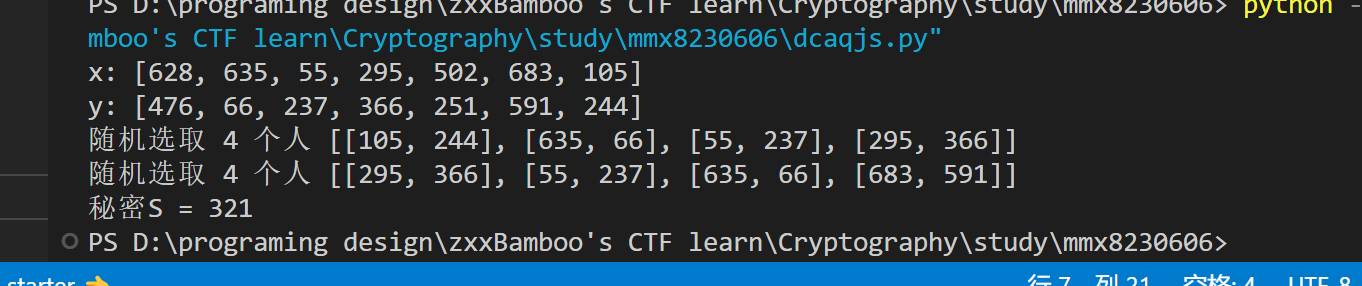
                t2 = get\_inverse(a[k][0] - a[j][0], p)

                t1 = (t1 \* (p - a[j][0]) \* t2) % p

        st += a[k][1] \* t1

    st = st % p

print("秘密S =", st)



百万富翁：

import random

import math

def is\_prime(x):  # 检测是否为素数

    for i in range(2, int(math.sqrt(x)) + 1):

        if x % i == 0:

            return False

    return True

def encry(x, p, n):  # RSA加密

    c = pow(x, p, n)

    return c

def decry(y, q, n):  # RSA解密

    m = pow(y, q, n)

    return m

N = 10

i = int(input("A的财富（1-9）百万："))

j = int(input("B的财富（1-9）百万："))

print("采用RSA加密算法")

pb = int(input("B的公钥："))

sb = int(input("B的私钥："))

n = int(input("n="))

y = []

while True:

    x = random.randint(10, 100)

    if is\_prime(x):

        break

print("随机整数x =", x)

c = decry(x, pb, n)

c = c - i

print("A发送给B { c-i =", c, "}")

for u in range(N):

    t = decry(c + u + 1, sb, n)

    y.append(t)

p = int(input("选取大素数p="))

for s in range(N):

    y[s] %= p

t2 = j

while t2 < N:

    y[t2] += 1

    t2 += 1

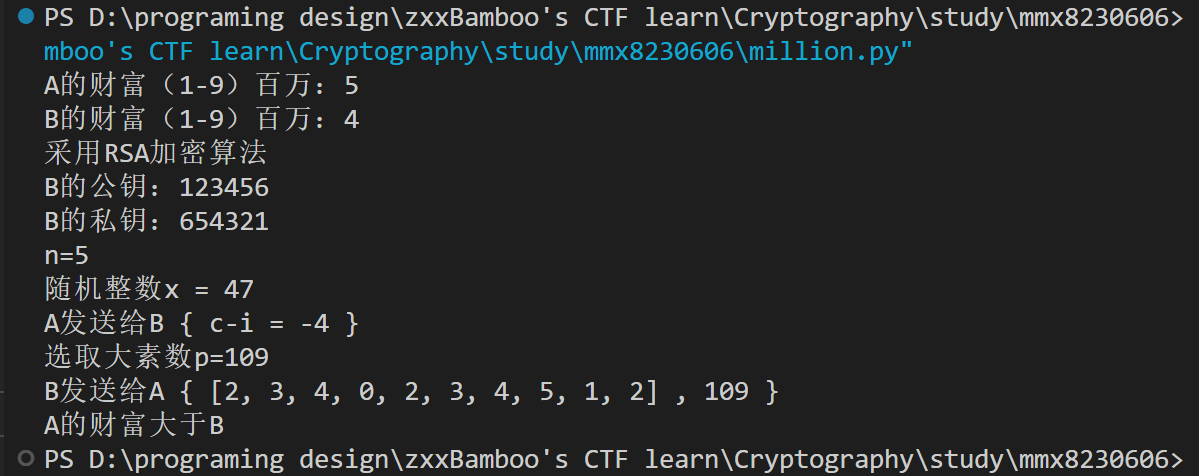
print("B发送给A {", y, ",",p, "}")

if y[i - 1] == x:

    print("B的财富大于等于A")

else:

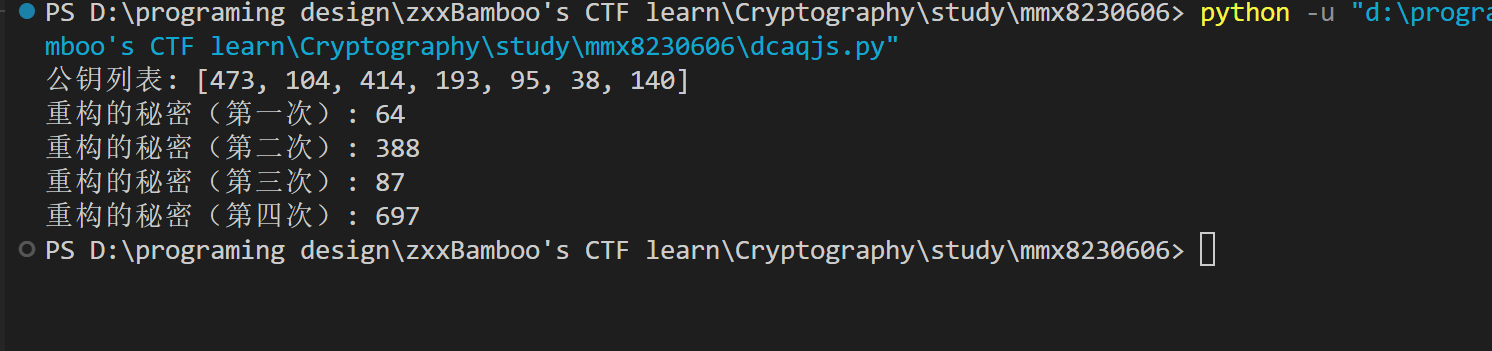
    print("A的财富大于B")



1. 实验总结

有bug,找bug

1. 公钥生成，设S1=3,S2=4,S3=5,验证结果：无问题



那么问题应该在拉格朗日插值运算那

1. import random
2. # def check\_inputs(S, p, x, k):
3. #     """
4. #     检查输入是否满足Shamir秘密共享方案的条件。
5. #     S: 秘密
6. #     p: 一个素数
7. #     x: 私钥列表
8. #     k: 重构秘密所需的最小共享数量
9. #     """
10. #     if not isinstance(S, int) or not isinstance(p, int)
11. #   or not isinstance(k, int):
12. #         return False
13. #     if not all(isinstance(xi, int) for xi in x):
14. #         return False
15. #     if S >= p or any(xi >= p for xi in x):
16. #         return False
17. #     if k > len(x):
18. #         return False
19. #     return True
20. def shamir\_share(S, p, x, k):
21. """
22. S: 秘密
23. p: 一个素数
24. x: 私钥列表
25. k: 重构秘密所需的最小共享数量
26. """
27. # # 检查输入的有效性
28. # if not check\_inputs(S, p, x, k):
29. #     raise ValueError("无效的输入。")
30. # 随机生成系数 S1 ,S2和S3
31. S1 = random.randint(1, p-1)
32. S2 = random.randint(1, p-1)
33. S3 = random.randint(1, p-1)
34. # 计算公钥
35. y = [(S + S1\*xi + S2\*xi\*\*2+S3\*xi\*\*3) % p for xi in x]
36. return y, S1, S2, S3
37. def lagrange\_interpolation(x, y, p):
38. """
39. x: x坐标列表
40. y: y坐标列表
41. p: 一个素数
42. """
43. n = len(x)
44. total = 0
45. for i in range(n):
46. xi, yi = x[i], y[i]
47. prod = 1
48. for j in range(n):
49. if i != j:
50. xj = x[j]
51. prod = (prod \* (0 - xj) \* pow(xi - xj, p-2, p)) % p
52. total = (total + yi\*prod) % p
53. return total
54. def shamir\_reconstruct(y, x, p, k):
55. """
56. y: 公钥列表
57. x: 与公钥对应的私钥列表
58. p: 一个素数
59. k: 重构秘密所需的最小共享数量
60. """
61. # 随机选择 k 个共享
62. shares = random.sample(list(zip(y, x)), k)
63. # 使用Lagrange插值求解秘密 S
64. x\_coords, y\_coords = zip(\*shares)
65. secret = lagrange\_interpolation(x\_coords, y\_coords, p)
66. return secret
67. # 参数
68. S = 54321
69. p = 727
70. x = [628, 635, 55, 295, 502, 683, 105]
71. k = 4
72. # 生成共享
73. y, S1, S2, S3 = shamir\_share(S, p, x, k)
74. print("公钥列表:", y)
75. # 两次重构秘密
76. secret1 = shamir\_reconstruct(y, x, p, k)
77. print("重构的秘密（第一次）:", secret1)
78. secret2 = shamir\_reconstruct(y, x, p, k)
79. print("重构的秘密（第二次）:", secret2)

不太稳定：

