**海南大学网络空间安全学院**

**实 验 报 告**

|  |  |
| --- | --- |
| **评定成绩** | **指导教师** |
|  | 王文丽 |

|  |  |
| --- | --- |
| **实验课程** | 安全协议分析与设计 |
| **实验名称** | 实现 Shamir秘密共享方案 |
| **学 号** | 20213006839 |
| **姓 名** | 甄五四 |
| **班 级** | 信息安全(密码学方向)理科实验班 |
| **完成日期** | 2024年5月25日 |

1. **实验目的**

1.熟练掌握Shamir秘密共享算法的秘密分配的步骤。

2.熟练掌握恢复秘密消息的机制(矩阵或者拉格朗日插值二选一)。

1. **实验任务**

1. 配置C或Python环境。

2. 任选一种语言实现(t,n)Shamir秘密共享方案，主要步骤要有标注。

3. 令秘密消息是11，构造(3,5)门限方案，选取多项式为F(x)=(7x^2+9x+11)mod13

1) 计算5个影子

2) 从任意3个影子中恢复出消息(矩阵或者拉格朗日插值二选一)

1. **实验环境**

**Windows 10,Pycharm 2023,python 3.12.2**

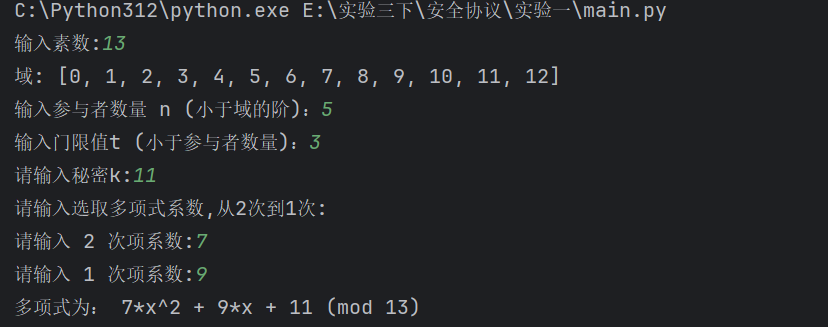
1. **实验内容及步骤(题目、相应程序代码和运行效果截图)**

第一步：首先根据输入q,(t,n)，以及k，系数构造出多项式。这里秘密消息是11，构造(3,5)门限方案，选取多项式为F(x)=(7x^2+9x+11)mod13。

# (t,n)Shamir秘密共享方案  
q = int(input("输入素数:"))  
while not(is\_prime(q)):  
 print("输入错误，请输入素数！")  
 q = int(input("输入素数:"))  
Field =list(range(q))  
print("域:", Field)  
n = int(input("输入参与者数量 n (小于域的阶)："))  
t = int(input("输入门限值t (小于参与者数量)："))  
k =int(input("请输入秘密k:"))  
a = [0] \* t  
print(f"请输入选取多项式系数,从{t-1}次到1次:")  
for i in range(t-1):  
 temp = int(input(f"请输入 {t-1-i} 次项系数:"))  
 a[i] = temp  
print("多项式为：",construct\_polynomial(t, q, a, k))

def construct\_polynomial(t, q, a, k):  
 *"""  
 构造并输出多项式 F(x) = a[0]\*x^(t-1) + a[1]\*x^(t-2) + ... + a[t-2]\*x + k (mod q)  
 """* polynomial\_terms = []  
 # 构造多项式项  
 for i in range(t - 1):  
 coefficient = a[i]  
 exponent = t - 1 - i  
 if coefficient != 0:  
 if exponent == 0:  
 term = f"{coefficient}"  
 elif exponent == 1:  
 term = f"{coefficient}\*x"  
 else:  
 term = f"{coefficient}\*x^{exponent}"  
 polynomial\_terms.append(term)  
 # 添加常数项 k  
 if k != 0:  
 polynomial\_terms.append(f"{k}")  
 # 组合所有项，形成多项式字符串  
 polynomial\_str = " + ".join(polynomial\_terms) + f" (mod {q})"  
 return polynomial\_str

运行结果：

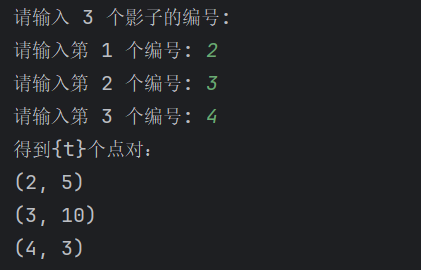


第二步：任意选取5个不同的x,计算5个影子

def F(x,t,q,a,k):  
 # 计算多项式 F(x) = a[0]\*x^(t-1) + a[1]\*x^(t-2) + ... + a[t-2]\*x + k (mod q)  
 result = 0  
 for i in range(t-1):  
 result = (result + a[i] \* pow(x, t - 1 - i, q)) % q  
 result = (result+k)%q  
 return result

# 计算多项式在不同影子值处的结果  
x\_values = []  
print(f"请输入 {n} 个影子的取值 x:")  
while len(x\_values) < n:  
 x = int(input(f"请输入第 {len(x\_values) + 1} 个 x 值: "))  
 if x in x\_values:  
 print(f"值 {x} 已存在，请输入不同的值。")  
 else:  
 x\_values.append(x)  
print(f"{n}个影子为：")  
Fx=[]  
for x in x\_values:  
 fx = F(x, t, q, a,k)  
 Fx.append(fx)  
 print(f"F({x}) = {fx}")

运行结果：



第三步：从任意3个影子中恢复出消息(矩阵或者拉格朗日插值二选一)

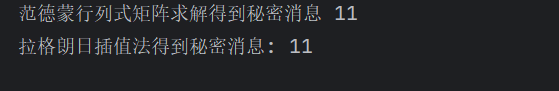
矩阵方法：

def matrix(x\_t,x\_values,Fx,q,t):  
 *'''矩阵求解法'''* '''  
 x\_t 选择的影子编号  
 x\_values，x的取值矩阵，x\_values[x\_t-1]则为选择编号对应x取值  
 Fx为对应同余方程fx的取值，Fx[x\_t-1]为选择x对应fx取值  
 t为门限值，t-1为最高次次数  
 q为同余方程组的模长  
 '''  
 x\_t = np.array([x - 1 for x in x\_t])  
 V = np.vander([x\_values[i] for i in x\_t], t, increasing=True)  
 V\_inv = np.linalg.inv(V)  
 coefficients = np.dot(V\_inv, [Fx[i] for i in x\_t]) % q  
 return coefficients[::-1] # 反转系数数组

拉格朗日插值法：

def lagrange\_interpolation(x\_t, x\_values,Fx, q):  
 *"""  
 使用拉格朗日插值法恢复秘密消息。  
 """* t = len(x\_t)  
 result = 0  
 for i in range(t):  
 term = Fx[x\_t[i]-1]  
 for j in range(t):  
 if i != j:  
 term \*= x\_values[x\_t[j]-1] \* pow(x\_values[x\_t[j]-1] - x\_values[x\_t[i]-1], -1, q)%q  
 term %= q  
 result += term  
 result %= q  
 return result

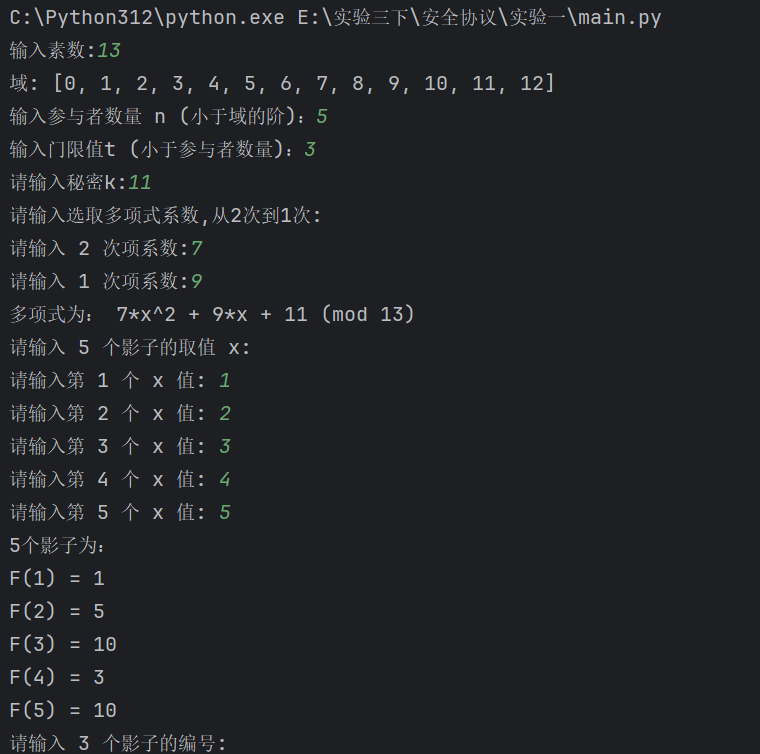
运行结果：

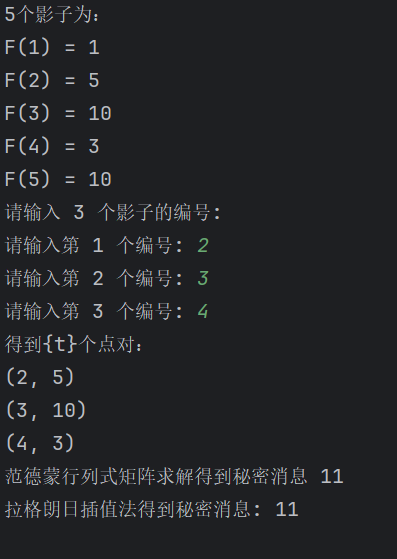


**完整代码：**

import math  
import numpy as np  
def is\_prime(num):  
 *"""判断一个数是否为素数"""* if num <= 1:  
 return False  
 if num == 2:  
 return True # 2 是素数  
 if num % 2 == 0:  
 return False # 排除所有偶数  
 max\_divisor = int(math.sqrt(num)) + 1  
 for divisor in range(3, max\_divisor, 2):  
 if num % divisor == 0:  
 return False  
 return True  
def construct\_polynomial(t, q, a, k):  
 *"""  
 构造并输出多项式 F(x) = a[0]\*x^(t-1) + a[1]\*x^(t-2) + ... + a[t-2]\*x + k (mod q)  
 """* polynomial\_terms = []  
 # 构造多项式项  
 for i in range(t - 1):  
 coefficient = a[i]  
 exponent = t - 1 - i  
 if coefficient != 0:  
 if exponent == 0:  
 term = f"{coefficient}"  
 elif exponent == 1:  
 term = f"{coefficient}\*x"  
 else:  
 term = f"{coefficient}\*x^{exponent}"  
 polynomial\_terms.append(term)  
 # 添加常数项 k  
 if k != 0:  
 polynomial\_terms.append(f"{k}")  
 # 组合所有项，形成多项式字符串  
 polynomial\_str = " + ".join(polynomial\_terms) + f" (mod {q})"  
 return polynomial\_str  
def F(x,t,q,a,k):  
 # 计算多项式 F(x) = a[0]\*x^(t-1) + a[1]\*x^(t-2) + ... + a[t-2]\*x + k (mod q)  
 result = 0  
 for i in range(t-1):  
 result = (result + a[i] \* pow(x, t - 1 - i, q)) % q  
 result = (result+k)%q  
 return result  
def matrix(x\_t,x\_values,Fx,q,t):  
 *'''矩阵求解法'''* '''  
 x\_t 选择的影子编号  
 x\_values，x的取值矩阵，x\_values[x\_t-1]则为选择编号对应x取值  
 Fx为对应同余方程fx的取值，Fx[x\_t-1]为选择x对应fx取值  
 t为门限值，t-1为最高次次数  
 q为同余方程组的模长  
 '''  
 x\_t = np.array([x - 1 for x in x\_t])  
 V = np.vander([x\_values[i] for i in x\_t], t, increasing=True)  
 V\_inv = np.linalg.inv(V)  
 coefficients = np.dot(V\_inv, [Fx[i] for i in x\_t]) % q  
 return coefficients[::-1] # 反转系数数组  
def lagrange\_interpolation(x\_t, x\_values,Fx, q):  
 *"""  
 使用拉格朗日插值法恢复秘密消息。  
 """* t = len(x\_t)  
 result = 0  
 for i in range(t):  
 term = Fx[x\_t[i]-1]  
 for j in range(t):  
 if i != j:  
 term \*= x\_values[x\_t[j]-1] \* pow(x\_values[x\_t[j]-1] - x\_values[x\_t[i]-1], -1, q)%q  
 term %= q  
 result += term  
 result %= q  
 return result  
if \_\_name\_\_ == "\_\_main\_\_":  
 # (t,n)Shamir秘密共享方案  
 q = int(input("输入素数:"))  
 while not(is\_prime(q)):  
 print("输入错误，请输入素数！")  
 q = int(input("输入素数:"))  
 Field =list(range(q))  
 print("域:", Field)  
 n = int(input("输入参与者数量 n (小于域的阶)："))  
 t = int(input("输入门限值t (小于参与者数量)："))  
 k =int(input("请输入秘密k:"))  
 a = [0] \* t  
 print(f"请输入选取多项式系数,从{t-1}次到1次:")  
 for i in range(t-1):  
 temp = int(input(f"请输入 {t-1-i} 次项系数:"))  
 a[i] = temp  
 print("多项式为：",construct\_polynomial(t, q, a, k))  
 # 计算多项式在不同影子值处的结果  
 x\_values = []  
 print(f"请输入 {n} 个影子的取值 x:")  
 while len(x\_values) < n:  
 x = int(input(f"请输入第 {len(x\_values) + 1} 个 x 值: "))  
 if x in x\_values:  
 print(f"值 {x} 已存在，请输入不同的值。")  
 else:  
 x\_values.append(x)  
 print(f"{n}个影子为：")  
 Fx=[]  
 for x in x\_values:  
 fx = F(x, t, q, a,k)  
 Fx.append(fx)  
 print(f"F({x}) = {fx}")  
  
 x\_t = []  
 print(f"请输入 {t} 个影子的编号:")  
 while len(x\_t) < t:  
 x = int(input(f"请输入第 {len(x\_t) + 1} 个编号: "))  
 if x in x\_t:  
 print(f"编号{x} 已存在，请输入不同的编号。")  
 else:  
 x\_t.append(x)  
 print("得到{t}个点对：")  
 for i in range(t):  
 print(f"{x\_values[x\_t[i]-1],Fx[x\_t[i]-1]}")  
 # 恢复出秘密消息  
 msg1 = matrix(x\_t,x\_values,Fx,q,t)  
 print("范德蒙行列式矩阵求解得到秘密消息",int(msg1[t-1]))  
 msg2 = lagrange\_interpolation(x\_t,x\_values,Fx,q)  
 print("拉格朗日插值法得到秘密消息:",msg2)

运行结果：

****

****

1. **实验总结**

通过这次实验，我深入理解了Shamir秘密共享算法的工作原理，包括如何分配秘密和如何从影子中恢复秘密，对秘密共享，门限秘密共享有了更深的理解

。我也学习了如何在Python环境中实现这个算法。这次实验让我对安全协议有了更深的理解，特别是对秘密共享方案的理解。我也提高了我的编程技能，特别是在处理数学问题和实现复杂算法方面。尽管我成功地完成了实验，但在实现过程中还是遇到了一些困难。例如，我在理解拉格朗日插值和矩阵恢复机制时花了很多时间。我认为我需要进一步提高我的数学技能和编程技能。总之，这次实验非常对我而言非常棒，我也收获满满，巩固了课上所学，对Shamir秘密共享有了更深的理解和掌握，但仍需继续学习强化。