南開大學
Nankai University

计算机学院和网络空间安全学院

数据安全实验报告

秘密共享实践

姓名：魏伯繁

学号：2011395

专业：信息安全

2023 年 4 月 24 日

目录

1 实验要求

复现课本中的实验 3-2，课本中的代码已经提供了一个实现求出三人投票数目综合的程序，根据课本提供的代码框架实现对于三个人所拥有的数据的平均值的计算。

2 实验过程

2.1 背景知识

秘密共享（Secret Sharing）是一种将秘密分割存储的密码技术，目的是阻止秘密过于集中，以达到分散风险和容忍入侵的目的，是信息安全和数据保密中的重要手段。目前，秘密共享已成为一种重要密码学工具，在诸多多方安全计算协议中被使用，例如拜占庭协议、多方隐私集合求交协议、阈值密码学等。

基于秘密分配和运算的形式，我们将秘密共享分为基于位运算的加性秘密共享和基于线性代数的线性秘密共享。依据秘密重构的条件或者秘密分享的份额数量等，又可以将秘密共享分为如下类别:

- 门限秘密共享：任意大于等于阈值的参与方集合可重构出秘密。
- 多重秘密共享：参与方的子秘密可以多次使用，分别恢复多个共享秘密
- 多秘密共享：一次共享，共享多个秘密，且子秘密可以重复使用
- 可验证秘密共享：可通过公共变量验证自己子秘密的正确性
- 动态秘密共享：允许添加或删除参与方，定期或不定期更新参与方的子秘密，还允许在不同的时间恢复不同的秘密。

设 t 和 n 为两个正整数，且 t t，n 个需要共享秘密的参与者集合为 P = P1,⋯, Pn 。一个 (t, n) 门限秘密共享体制是指：假设 P1,⋯, Pn 要共享同一个秘密 s，将 s 称为主秘密，至少 t 个参与者才可以共同恢复主秘密 s。有一个秘密管理中心 P0 来负责对 s 进行管理和分配，P0 掌握有秘密分配算法和秘密重构算法，这两个算法均满足重构要求和安全性要求。

> **秘密分配。**秘密管理中心P_0首先通过将主秘密s输入秘密分配算法，生成n个值，分别为$s_1, \dots, s_n$，称$s_1, \dots, s_n$为子秘密。然后秘密管理中心P_0分别将秘密分配算法产生的子秘密$s_1, \dots, s_n$通过P_0与P_i之间的安全通信信道秘密地传送给参与者P_i，参与者P_i不得向任何人泄露自己所收到的子秘密s_i。

图 2.1: 秘密分配定义

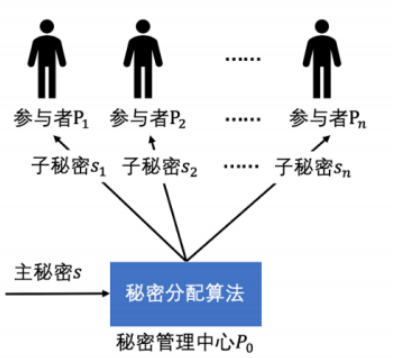


图 2.2: 秘密分配图示

秘密重构。门限值 t 指的是任意大于或等于 t 个参与者 Pi，将各自掌握的子秘密 si 进行共享，任意的一个参与者 Pi 在获得其余 i− 1 个参与者所掌握的子秘密后，都可独立地通过秘密重构算法恢复出主秘密 s。而即使有任意的 n-t 个参与者丢失了各自所掌握的子秘密，剩下的 t 个参与者依旧可以通过将各自掌握的子秘密与其他参与者共享，再使用秘密重构算法来重构出主秘密 s。安全性要求任意攻击者通过收买等手段获取了少于 t 个的子秘密，或者任意少于 t 个参与者串通都无法恢复出主秘密 s，也无法得到主秘密 s 的信息

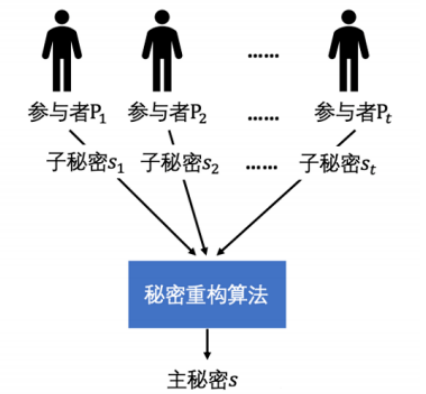
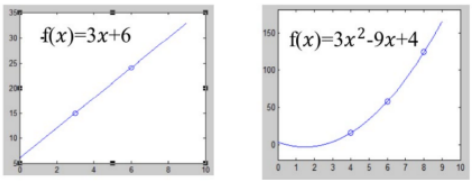


图 2.3: 秘密重构图示

经过前面的一些介绍，有一种很自然的实现方式就是构造函数的方法。

1. Shamir 方案



Shamir 于 1979 年，基于多项式插值算法设计了 Shamir(t,n)门限秘密共享体制。构造思路：

- 平面上不同的两个点唯一的确定平面上的一条直线，即一次多项式。
- 平面上不同的三个点唯一的确定平面上的一个二次多项式。
- 一般的，设$\{(x_1,y_1),\cdots(x_k,y_k)\}$是平面上 k 个不同的点构成的点集，那么在平面上存在唯一的$k-1$次多项式$f(x)=a_0+a_0x+\cdots a_{k-1}x^{k-1}$通过这 k 个点。
- 若把秘密 s 取做$f(0)$，n 个份额取做$f(i)(i=1,\cdots,n)$，那么利用其中任意 k 个份额就可以重构$f(x)$，从而得到秘密$s=f(0)$。

图 2.4: shamir 方案

而恢复函数的方式我们可以使用拉格朗日插值法进行恢复

假设这 t 个子秘密分别为(x_i,y_i)，其中$y_i=f(x_i),i=1,\cdots,t$且$i\neq j$时$x_i\neq x_j$。参与者$P_1,\ldots,P_t$共同计算：

$$h(x)=y_1\frac{(x-x_2)(x-x_3)\cdots(x-x_t)}{(x_1-x_2)(x_1-x_3)\cdots(x_1-x_t)}+y_2\frac{(x-x_1)(x-x_3)\cdots(x-x_t)}{(x_2-x_1)(x_2-x_3)\cdots(x_2-x_t)}+\cdots$$
$$+y_t\frac{(x-x_1)(x-x_2)\cdots(x-x_{t-1})}{(x_t-x_1)(x_t-x_2)\cdots(x_t-x_{t-1})}$$

显然，$h(x)$是一个$t-1$次的多项式，且因为$i\neq j$时$x_i\neq x_j$，每个加式的分母均不为零，因此对于$i=1,\ldots,t$，$y_i=h(x_i)=f(x_i)$。

又根据多项式的性质，如果存在两个最高次均为$t-1$次的多项式，这两个多项式在t个互不相同的点所取的值均相同，那么这两个多项式相同，即$h(x)=f(x)$。

因此，参与者P_i通过分享各自秘密，共同计算$h(0)=f(0)=s$，即可恢复主秘密s。

图 2.5: 拉格朗日插值法公式

于是，我们可以有这样的一种密钥分配方式，将秘密信息隐藏在 f(0) 中，如果要构造一个 (t,n) 的 shamir 门，思路就是构造一个 t-1 次的多项式函数，取这个 t-1 次表达式上的 n 个点分配各 n 个不同的人即可，这样任意 t 个人都可以通过拉格朗日插值法确定 f(0) 的值以获得正确的秘密值。

2.2 环境配置

本次实验的环境不需要做过多的特殊配置，只需要一个 python 的编译环境即可，一般 ubuntu 下都会自带 python3 的编译环境，所以我们只需要将代码拷贝到对应的 py 文件下再使用 python 命令就可以对指定的文件进行执行。

2.3 代码实现

其中具体的交互流程在前面的原理介绍中已经详细交流过，在代码实现过程中着重介绍数学计算的部分：这段代码用于生成一个多项式。在函数中，x0 是常数项系数，T 是多项式的最高次数，p 是取模的值，fname 是多项式的名字，用于输出显示。

首先，生成一个空的列表 f，用于存储生成的多项式系数。将常数项系数 x0 添加到 f 列表的第一个位置。之后，使用循环语句生成多项式中余下的 T 个系数，这些系数是在 0 到 p 之间的随机整数。

接下来，将生成的多项式系数输出到屏幕上，以便进行检查和调试。根据生成的多项式系数 f，通过循环语句构建多项式 f_print，并将其输出到屏幕上。多项式的输出形式为：

$$f_{fname} = x0 + \ 1x^1 + \ 2x^2 + ... + \ Tx^T$$

最后，返回生成的多项式系数 f。

```
# 构建多项式：x0 为常数项系数，T 为最高次项次数，p 为模数,fname 为多项式名
def get_polynomial(x0,T,p,fname):
    f=[]
    f.append(x0)
    for i in range(0,T):
        f.append(random.randrange(0,p))
    # 输出多项式
    f_print='f'+fname+'='+str(f[0])
    for i in range(1,T+1):
        f_print+='+'+str(f[i])+'x^'+str(i)
    print(f_print)
    return f
```

这段代码用于实现重构多项式并返回 x=0 时的多项式值。传入参数包括 x，fx，t，p，其中 x 是多项式中每个系数对应的 x 值，fx 是多项式中每个系数对应的函数值，t 是多项式的系数数，p 是取模的值。

首先初始化 ans=0，用于存储计算结果。然后使用 for 循环遍历多项式 fx 中每一个系数的位置 i，将 fx[i] 和 t 对 p 取模，确保其在模 p 意义下的值。接着，使用多项式插值法计算出 x=0 时多项式的值。具体而言，内层的 for 循环中，首先从 x 和 fx 中取出与 fx[i] 对应的 x 和 fx 值，然后计算出 i 不在其中的每个 x 值的积，并利用快速幂算法计算出对应的逆元。最后，将这些积和 fx[i] 相乘，再对 p 取模，这样就得到了插值后多项式在每一个位置处的值，最后将其累加到 ans 中。

函数最后返回 ans，也就是插值后多项式在 x=0 时的值。

```
# 重构函数 f 并返回 f(0)
def restructure_polynomial(x,fx,t,p):
    ans=0
    # 利用多项式插值法计算出 x=0 时多项式的值
    for i in range(0,t):
        fx[i]=fx[i]%p
```

```
            fxi=1
            # 在模 p 下，(a/b)%p=(a*c)%p，其中 c 为 b 在模 p 下的逆元，c=b^(p-2)%p
            for j in range(0,t):
                if j !=i:
                    fxi=(-1*fxi*x[j]*quickpower(x[i]-x[j],p-2,p))%p
            fxi=(fxi*fx[i])%p
            ans=(ans+fxi)%p
        return ans
```

对于最终想达到实验目的，我们可以在 vote_counter.py 文件中对最后的求和后的 d 进行除三处理，这样就可以获得最终的三个人所拥有的平均数。

```
    import ss_function as ss_f
    # 设置模数 p
    p=1000000007
    # 随机选取两个参与方，例如 student2 和 student3，获得 d2,d3，从而恢复出 d=a+b+c
    # 读取 d2,d3
    d_23=[]
    for i in range(2,4):
        with open(f'd_{i}.txt', "r") as f: # 打开文本
            d_23.append(int(f.read())) # 读取文本
    # 加法重构获得 d
    d=ss_f.restructure_polynomial([2,3],d_23,2,p)
    d=d/3
    print(f'三人拥有数值的平均数为：{d}')
```

3 心得体会

在进行秘密共享实验之前，我对于 shamir 门方法和拉格朗日插值法并不是很了解。但通过这次实验，我对于这两种方法的原理和实现有了更深入的认识。

在实验中，我们主要使用了 shamir 门方法和拉格朗日插值法来实现秘密共享。shamir 门方法是一种公开密文系统，可以通过把信息分为不同的部分，将其多份分散到不同的容器中来保护秘密信息，而拉格朗日插值法则是一种数学方法，可用于恢复丢失数据。通过将 shamir 门方法和拉格朗日插值法结合起来，我们实现了将秘密信息分为多份，并分散存储到不同容器中的共享过程。

在实验过程中，我意识到了秘密共享的重要性。无论是对于个人还是对于企业，秘密信息的重要性都不言而喻。而通过秘密共享，我们可以让秘密信息更加安全地存储。同时，shamir 门方法和拉格朗日插值法也在保护数据的完整性和可靠性方面发挥了作用。这种方法能够有效避免因个别容器损坏或数据丢失而导致信息泄露或无法恢复数据。

除此之外，我还从这次实验中学到了一些密码学技术。比如多项式插值、有限域运算和离散对数等概念。这些知识可以进一步应用于加密算法和安全通信领域，提高我们对于数据安全的保护能力。

总体而言，这次实验让我对于秘密共享的原理和实现有了更加深入的认识。我也意识到了数据保护在现今互联网时代的重要性，并对密码学技术有了更为深刻的理解。