

数据安全 -- 秘密共享实践

学号：2212452 姓名：孟启轩 专业：计算机科学与技术

一、实验要求

借鉴实验3.2，实现三个人对于他们拥有的数据的平均值的计算。

实验3.2：假设有三个同学需要对班里的优秀干部 Alice、Bob、Charles、Douglas 进行投票，最后统计各个班干部获得的票数。这个时候就可以利用 Shamir 秘密共享将各个投票方的投票分享出去并进行隐私求和计算。

二、实验思路

基于 Shamir 门限秘密共享的协议设计如下：

目标：计算 Alice 的票数总和与平均值。三位学生 Student1、Student2 和 Student3 分别持有隐私输入 a、b、c（取值为 0 或 1，表示是否投票给 Alice）。

协议步骤

1. 秘密共享阶段

- 每位学生使用 Shamir 的 (2,3) 门限秘密共享方案，将自身隐私输入加密并分发给其他两位学生。
例如：
 - Student1 将 a 分割为三个份额 a_1, a_2, a_3 ，并将 a_2, a_3 分别发送给 Student2 和 Student3；
 - Student2 将 b 分割为 b_1, b_2, b_3 ，并发送 b_1, b_3 给 Student1 和 Student3；
 - Student3 将 c 分割为 c_1, c_2, c_3 ，并发送 c_1, c_2 给 Student1 和 Student2。
- 最终，每位学生持有其他两人的秘密份额：
 - Student1 持有 a_1 （自身 a 的份额）、 b_1 （Student2 的 b 份额）、 c_1 （Student3 的 c 份额）；
 - Student2 持有 a_2, b_2, c_2 ；
 - Student3 持有 a_3, b_3, c_3 。

2. 本地计算阶段

- 每位学生将持有的三个份额相加，生成本地聚合值：
 - Student1 计算 $d_1 = a_1 + b_1 + c_1$ ；
 - Student2 计算 $d_2 = a_2 + b_2 + c_2$ ；
 - Student3 计算 $d_3 = a_3 + b_3 + c_3$ 。

3. 重构与计算阶段

- 计票员从三位学生中任意选取两人（如 Student2 和 Student3），获取其聚合值 d_2 和 d_3 。
- 利用 Shamir 门限方案的线性性质，通过 d_2 和 d_3 重构出总和 $d = a + b + c$ 。
- 最终计算 Alice 的平均票数：平均值 = $d / 3$ 。

关键性质

- **隐私性**：每位学生仅能获取其他两人隐私输入的加密份额，无法单独推断原始值。
- **安全性**：Shamir 的 (2,3) 门限机制确保任意两个份额可重构秘密，但单个份额无意义。
- **效率**：通过线性共享的叠加性质，避免了复杂的同态加密操作，仅需简单加法即可完成聚合。

三、实验过程

1、创建文件

新建一个文件夹vote，在 vote 下打开终端，执行如下命令，创建文件 `ss_function.py`、`ss_student.py`、`count_student.py` 和 `vote_counter.py`

```
touch ss_function.py
touch ss_student.py
touch count_student.py
touch vote_counter.py
```

2、定义函数

`ss_function.py` 代码里面定义了一些秘密共享过程中三个学生以及计票员会用到的函数。

```
import random

def quick_power(a: int, b: int, p: int) -> int:
    """
    快速幂算法：计算 (a^b) % p
    参数：
        a (int): 底数
        b (int): 指数
        p (int): 模数
    返回：
        int: 计算结果
    """
    a = a % p
    result = 1
    while b != 0:
        if b & 1:
            result = (result * a) % p
        b >>= 1
        a = (a * a) % p
    return result

def generate_polynomial(constant_term: int, degree: int, mod: int, name: str) -> list:
    """
    构建多项式：形式为 f(x) = a0 + a1x + a2x^2 + ... + adx^d
    参数：
        constant_term (int): 常数项系数
        degree (int): 多项式次数
    """
```

```

        mod (int): 模数
        name (str): 多项式名称标识符
    返回:
        list: 多项式系数列表 [a0, a1, ..., ad]
    """
    coefficients = [constant_term]
    for _ in range(degree):
        coefficients.append(random.randint(0, mod - 1))

    # 输出多项式表达式
    polynomial_str = f'f{name} = {coefficients[0]}'
    for i in range(1, degree + 1):
        polynomial_str += f' + {coefficients[i]}x^{i}'
    print(polynomial_str)

    return coefficients

def evaluate_polynomial(coefficients: list, x: int, mod: int) -> int:
    """
    计算多项式在 x 处的值: f(x) % mod
    参数:
        coefficients (list): 多项式系数列表
        x (int): 输入值
        mod (int): 模数
    返回:
        int: 计算结果
    """
    result = coefficients[0]
    for i in range(1, len(coefficients)):
        result = (result + coefficients[i] * quick_power(x, i, mod)) % mod
    return result

def reconstruct_polynomial(shares_x: list, shares_y: list, threshold: int, mod:
int) -> int:
    """
    使用拉格朗日插值法重构多项式并计算 f(0)
    参数:
        shares_x (list): 已知点的 x 坐标
        shares_y (list): 对应的 y 值
        threshold (int): 需要使用的点数量
        mod (int): 模数 (需为质数)
    返回:
        int: 重构后的 f(0) 值
    """
    result = 0
    for i in range(threshold):
        yi = shares_y[i] % mod
        xi = shares_x[i]

        # 计算拉格朗日基多项式 L_i(0)
        lagrange_term = 1
        for j in range(threshold):

```

```

        if i != j:
            xj = shares_x[j]
            denominator = quick_power(xi - xj, mod - 2, mod) # 费马小定理求逆

            lagrange_term = (-lagrange_term * xj * denominator) % mod

        result = (result + yi * lagrange_term) % mod
    return result

```

3、秘密共享

三个学生分别运行 `ss_student.py`，将自己的秘密投票值共享给另外两个学生。`ss_student.py`代码如下：

```

import ss_function as ss_f

def main():
    # 设置模数
    MODULUS = 1000000007
    print(f'模数 p: {MODULUS}')

    # 获取用户输入
    participant_id = int(input("请输入参与方 id:"))
    secret_value = int(input(f'请输入 student_{participant_id}的投票值 s:'))

    # 初始化秘密份额
    shares_x = [1, 2, 3]
    shares_y = []

    # 生成多项式和秘密份额
    print(f'Student_{participant_id}的投票值的多项式及秘密份额: ')
    polynomial = generate_and_display_polynomial(secret_value, shares_x, MODULUS,
participant_id)
    generate_shares(polynomial, shares_x, shares_y, MODULUS)

    # 分发秘密份额到文件
    distribute_shares(participant_id, shares_y)

def generate_and_display_polynomial(s, x_values, p, id_suffix):
    """生成多项式并显示表达式"""
    polynomial = ss_f.generate_polynomial(s, 1, p, str(id_suffix))
    for x in x_values:
        fx = ss_f.evaluate_polynomial(polynomial, x, p)
        print(f'({x},{fx})')
    return polynomial

def generate_shares(polynomial, x_values, shares_y, p):
    """计算所有秘密份额"""
    for x in x_values:
        shares_y.append(ss_f.evaluate_polynomial(polynomial, x, p))

def distribute_shares(participant_id, shares_y):

```

```

"""将份额写入对应文件"""
for i in range(1, 4):
    filename = f'student_{participant_id}_{i}.txt'
    with open(filename, 'w') as f:
        f.write(str(shares_y[i-1]))

if __name__ == "__main__":
    main()

```

3个Student 分别执行如下命令，输入 id 和投票值 1

```

python3 ss_student.py
1
1

python3 ss_student.py
2
1

python3 ss_student.py
3
1










```

```

mqx@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab4/vote$ python3 ss_student.py
模数 p: 1000000007
请输入参与方 id:1
请输入 student_1的投票值 s:1
Student_1的投票值的多项式及秘密份额:
f1=1+534862552x^1
(1,534862553)
(2,69725098)
(3,604587650)
mqx@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab4/vote$ python3 ss_student.py
模数 p: 1000000007
请输入参与方 id:2
请输入 student_2的投票值 s:1
Student_2的投票值的多项式及秘密份额:
f2=1+496667876x^1
(1,496667877)
(2,993335753)
(3,490003622)
mqx@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab4/vote$ python3 ss_student.py
模数 p: 1000000007
请输入参与方 id:3
请输入 student_3的投票值 s:1
Student_3的投票值的多项式及秘密份额:
f3=1+547407132x^1
(1,547407133)
(2,94814258)
(3,642221390)
mqx@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab4/vote$

```

在文件夹下会生成 9 个 txt 文件，分别保存三个秘密值的秘密份额：

 student_1_1.txt	2025/5/5 14:41	文本文档	1 KB
 student_1_2.txt	2025/5/5 14:41	文本文档	1 KB
 student_1_3.txt	2025/5/5 14:41	文本文档	1 KB
 student_2_1.txt	2025/5/5 14:42	文本文档	1 KB
 student_2_2.txt	2025/5/5 14:42	文本文档	1 KB
 student_2_3.txt	2025/5/5 14:42	文本文档	1 KB
 student_3_1.txt	2025/5/5 14:42	文本文档	1 KB
 student_3_2.txt	2025/5/5 14:42	文本文档	1 KB
 student 3 3.txt	2025/5/5 14:42	文本文档	1 KB

4、处理秘密数据

三个学生分别执行 count_student.py，获取另外两个学生的投票值的秘密份额，并将三个投票值的秘密份额相加，count_student.py代码如下：

```
# 设置模数 p (需为大素数，用于模运算保证安全性)
p = 1000000007
# 输入当前参与方的 id (需与生成阶段保持一致)
id = int(input("请输入参与方 id:"))

# 读取其他参与方分享给当前参与方的秘密份额
# 每个文件 student_i_id.txt 存储了由第 i 个参与方生成的份额
data = []
for i in range(1, 4):
    # 打开并读取存储在对应文件中的秘密份额值
    with open(f'student_{i}_{id}.txt', "r") as f:
        data.append(int(f.read())) # 将文本内容转换为整数并存入列表

# 计算三个秘密份额的和 (模 p 运算)，每个数据项对应一个秘密份额，总共有 3 个份额
d = 0
for i in range(0, 3):
    d = (d + data[i]) % p # 累加并保持模 p 运算

# 将计算结果保存到 d_id.txt 文件，该文件将用于后续的秘密重构阶段
with open(f'd_{id}.txt', 'w') as f:
    f.write(str(d)) # 写入计算后的秘密份额值
```

3个Student 分别执行如下命令，获得三个投票值的秘密份额相加的结果分别保存到 d_1.txt、d_2.txt 和 d_3.txt。




```
python3 count_student.py
1

python3 count_student.py
2

python3 count_student.py
3
```

```
mqx@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab4/vote$ python3 count_student.py
请输入参与方 id:1
mqx@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab4/vote$ python3 count_student.py
请输入参与方 id:2
mqx@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab4/vote$ python3 count_student.py
请输入参与方 id:3
mqx@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab4/vote$
```

在文件夹下会生成 3 个 txt 文件，分别为 d_1.txt、d_2.txt 和 d_3.txt:

 d_1.txt	2025/5/5 14:44	文本文档	1 KB
 d_2.txt	2025/5/5 14:44	文本文档	1 KB
 d_3.txt	2025/5/5 14:44	文本文档	1 KB

5、重构秘密

vote_counter.py代码如下：

```
import ss_function as ss_f
# 设置模数 p (需为大素数，用于模运算保证安全性)
p = 1000000007

# 随机选取两个参与方 (如 student2 和 student3) 进行秘密重构
# 读取其他参与方存储的秘密份额 d2 和 d3
d_23 = []
for i in range(2, 4):
    # 打开并读取存储在对应文件中的秘密份额值
    with open(f'd_{i}.txt', "r") as f:
        d_23.append(int(f.read())) # 将文本内容转换为整数并存入列表

# 使用拉格朗日插值法重构原始秘密值 d = a + b + c
d = ss_f.reconstruct_polynomial([2, 3], d_23, 2, p)
d = d / 3 # 计算平均值
print(f'得票结果为: {d}')
```

执行python3 vote_counter.py，得到三个人拥有数据的平均值的计算。

结果为 1，说明实验正确：

```
mqx@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab4/vote$ python3 vote_counter.py
得票结果为: 1.0
mqx@LAPTOP-2BCGD4JI:/mnt/f/study/datasecurity/lab4/vote$
```

四、心得体会

通过本次实验，我深入学习了秘密共享的核心概念，并掌握了 Shamir 门限秘密共享方案的原理与实现方法。该方案通过多项式插值技术实现了对隐私数据的安全分割与重构，为多方安全计算提供了理论基础。这一实践让我直观感受到密码协议如何通过数学工具平衡隐私保护与协作效率。