Exp5 同态加密匿名投票

环境设置

本实验的环境设置为macOS 13.6.7, python 3.10.5

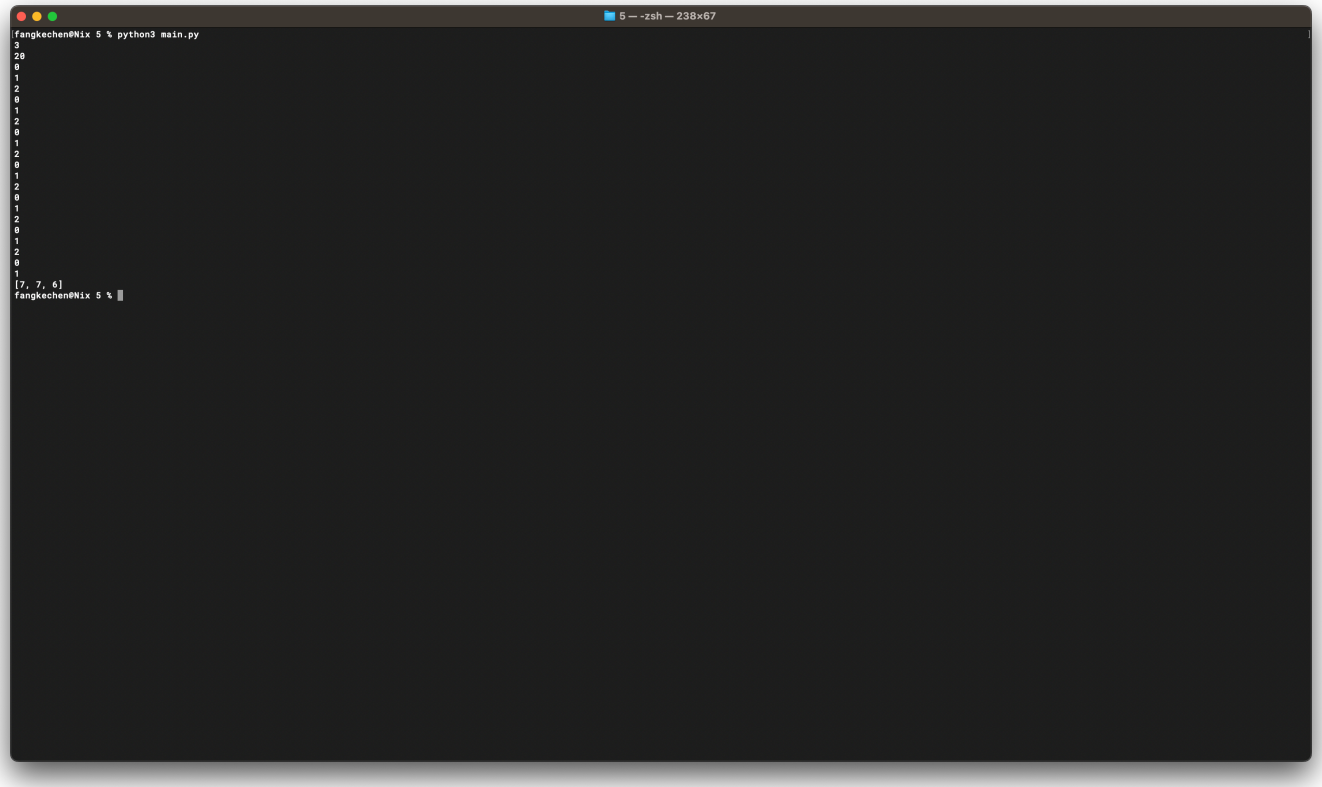
关键步骤

1. 在主程序代码 `main.py` 中，首先实现了Paillier算法。其中 `generate` 函数用于生成公钥和私钥。 `encode` 用来对明文进行加密， `decode` 用于对密文进行解密。
2. 参考书中匿名电子投票的过程，代码中实现了三个类： `Voter, Tally, Show` 分别表示投票方、计票方和公布方。其中，私钥只有在公布方处拥有。在电子投票过程中，投票方首先使用公钥将其对每个候选者的投票数进行加密，并返回加密后的结果。计票方得到所有投票方的结果，然后对于每一位候选者，将所有投票方对其加密后的投票进行相乘，全部运算结束后返回每个候选者的加密投票结果。公布方获得计票方的结果后，使用私钥将其解密。
3. 在实现中，需要依次每行输入候选者人数、投票方人数。然后对于每位投票方每行输入投给的候选者编号（从 0 开始计），最终代码会输出计票结果。
4. 代码运行方式：直接在命令行运行 `python main.py` 即可。

影响因素分析

本实验中由于每位投票方投的票都是一个 1 其余为 0 ，所以计票方还是可以知道每个投票方投了谁，依旧有风险。一种不完全的解决方案是对于每位候选者都生成一对公私钥，投票时使用候选者对应的公钥加密。但实际上计票方还是可以对所有人加密后的投票进行统计来获取信息。

实验现象



在该次测试中，共有 3 位候选者， 20 位投票方且依次投给 $0, 1, 2, \cdots$ 号候选者，可以发现三位候选者的票数应当分别为 $7, 7, 6$ 票。程序最终的输出结果符合预期。

关键源代码

本实验的关键源代码为：

```
import random
from math import gcd

def lcm(a, b):
    return a * b // gcd(a, b)

def gen_prime():
    while True:
        p = random.randint(1 << 15, 1 << 16)
        if is_prime(p):
            return p

def is_prime(n):
    if n <= 1:
        return False
    for i in range(2, int(n ** 0.5) + 1):
        if n % i == 0:
            return False
    return True

def L(x, n):
```

```
    return (x - 1) // n

def generate():
    while True:
        p = gen_prime()
        q = gen_prime()
        if gcd(p * q, (p - 1) * (q - 1)) == 1:
            break
    n = p * q
    lamb = lcm(p - 1, q - 1)
    while True:
        g = random.randint(1, n ** 2)
        if gcd(L(g, n), n) == 1:
            break

    mu = pow(L(pow(g, lamb, n ** 2), n), -1, n)
    pub = (n, g)
    priv = (lamb, mu)
    return pub, priv

def encode(pub, m):
    n, g = pub
    r = random.randint(1, n)
    return pow(g, m, n ** 2) * pow(r, n, n ** 2) % (n ** 2)

def decode(priv, pub, c):
    n, _ = pub
    lamb, mu = priv
    return L(pow(c, lamb, n ** 2), n) * mu % n

class Voter:
    def __init__(self, pub):
        self.pub = pub

    def vote(self, idx, n):
        return [encode(pub, 0) if i != idx else encode(pub, 1) for i in range(n)]

class Tally:
    def __init__(self, pub):
        self.pub = pub

    def count(self, votes, n):
        res = [1 for _ in range(n)]
        for vote in votes:
            for i in range(n):
                res[i] *= vote[i]
                res[i] %= pub[0] ** 2
        return res
```

```
class Show:
    def __init__(self, pub, priv):
        self.pub = pub
        self.priv = priv

    def show(self, res):
        return [decode(priv, pub, r) for r in res]

if __name__ == '__main__':
    pub, priv = generate()
    n = int(input())
    m = int(input())
    voters = [Voter(pub) for _ in range(m)]
    tally = Tally(pub)
    show = Show(pub, priv)
    votes = []
    for i in range(m):
        votes.append(voters[i].vote(int(input()), n))
    res = tally.count(votes, n)
    print(show.show(res))
```