

《数据安全》实验报告

姓名： 费泽锟 学号： 2014074 班级： 信安班

实验名称：

秘密共享实践

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。

实验过程：

1. 复现实验 3.2，也就是投票方求和示例，理解各部分代码

我们可以直接在 Ubuntu 18.04 虚拟机上直接根据实验指导书的步骤，一步一步地进行实验，就可以得到最后的结果，这里我们根据源代码进行逐个部分的理解与复现。

首先第一部分就是理解 `ss_function.py` 到底提供了怎样的函数以及函数的具体内容，

```
def quickpower(a,b,p):  
    a=a%p  
    ans=1  
    while b!=0:  
        if b&1:  
            ans=(ans*a)%p  
        b>>=1  
        a=(a*a)%p  
    return ans
```

对于快速模幂运算的过程，大致即为相乘取模右移的过程。该部分可以参考密码学之中使用到的快速模幂运算。构建多项式的函数 `get_polynomial`，即为使用 `random` 函数为多项式函数的系数随机构造使用 `list` 数据结构进行存储。

```

#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

```

对于重构函数我们可以参考计算方法课程之中讲授的拉格朗日插值方法,该函数就是使用拉格朗日插值的方法对多项式函数进行重构的,因为我们需要构造的是(2,3)的门陷方案,也就是说只需要两个人就可以重构这个函数,所以这个函数的最高次数为 1 次,也就是随机线性函数。对于参与计算的每一方而言,他的秘密值也就是存储在常数项之中。

接着我们来对 ss_student.py 中的内容进行进一步的理解。

```

#计算多项式及秘密份额(t=2,n=3)
print(f'Student_{id}的投票值的多项式及秘密份额: ')
f=ss_f.get_polynomial(s,1,p,str(id))
temp=[]
for j in range(0,3):
    temp.append(ss_f.count_polynomial(f,shares_x[j],p))
    print(f'({shares_x[j]}, {temp[j]})')
    shares_y.append(temp[j])

```

我们最需要理解的就是上述这一部分的代码,对于参与计算的每一方按照 ID 的顺序,分别对于三个随机的线性函数计算存储在三个参与方的秘密值,分别计算线性函数时使用的 x 值也就是 0, 1, 2...这种顺序,然后分别存储 9 个文件之中。

接着我们对于 count_student.py 的重要部分进行理解:

```

#计算三个秘密份额的和
d=0
for i in range(0,3):
    d=(d+data[i])%p

```

我们需要对于这一部分有一定的理解,这里其实与我们讲述的多项式函数的加法同态性有一定的关系,这里计算本地的存储的三个秘密分享值,其实相当于将三个随机线性函数进行了相加的操作,所以最后的秘密值就是我们所希望的投票之和。

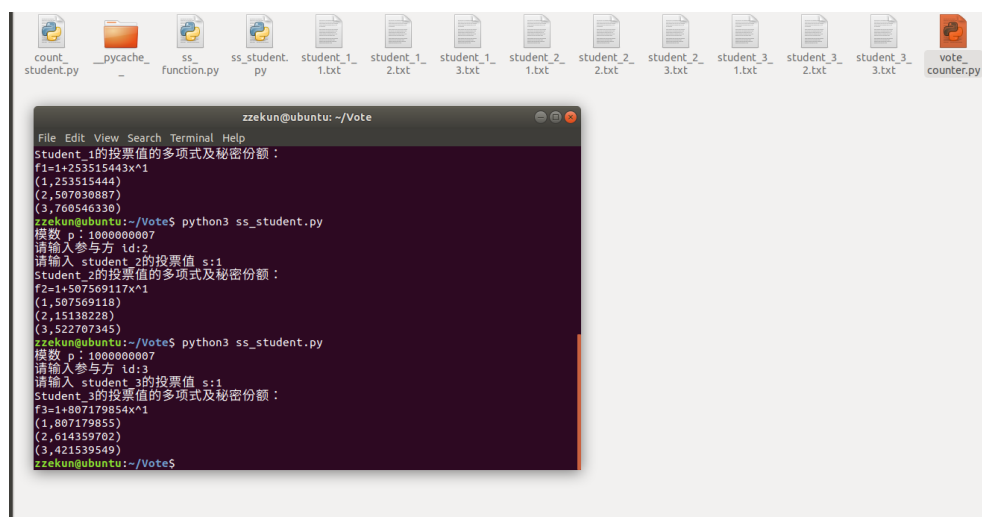
最后我们就是去理解 `vote_counter.py` 中的内容,这一部分没有很重要的内容。

```
#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
```

我们选择了第二方和第三方计算得到的秘密值之和进行计算,重构出三个随机线性函数相加之后的结果,所得的常数项也就是最后的投票结果了。

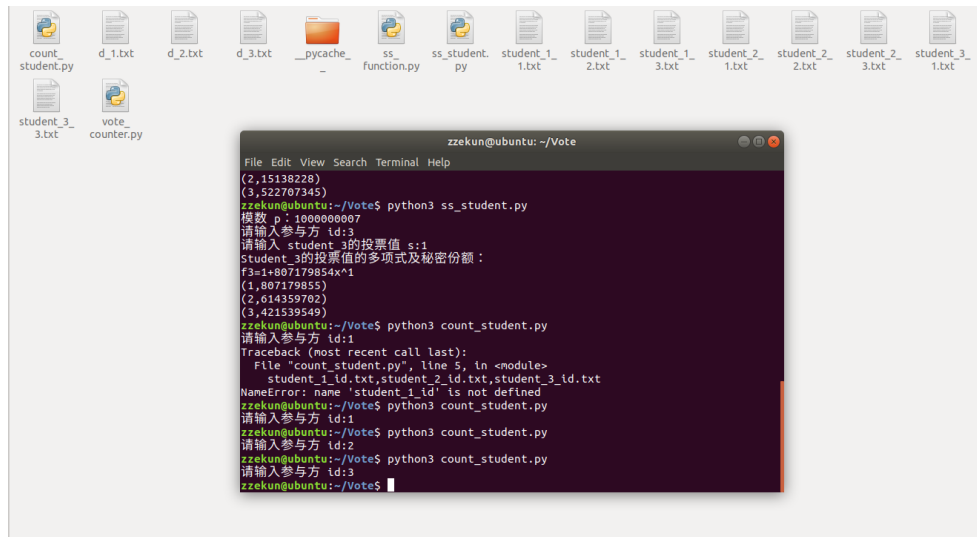
我们根据如上的理解以及实验指导书对应的 `bash` 命令,可以复现得到的结果如下:

(1) 输入参与方 ID 与投票值生成秘密分享文件



```
File Edit View Search Terminal Help
Student_1的投票值的多项式及秘密份额:
f1=1+253515443x^1
(1,253515444)
(2,507030887)
(3,760546330)
zzekun@ubuntu:~/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+507569117x^1
(1,507569118)
(2,151382228)
(3,522707345)
zzekun@ubuntu:~/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+807179854x^1
(1,807179855)
(2,614359702)
(3,421539549)
zzekun@ubuntu:~/Vote$
```

(2) 计算各自拥有的秘密值的和(因为对于多项式秘密共享而言是同态的)



(3) 三方输入值均为 1，得到的结果为 3，验证正确

```
zzekun@ubuntu: ~/Vote
File Edit View Search Terminal Help
zzekun@ubuntu:~/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+807179854x^1
(1,807179855)
(2,614359702)
(3,421539549)
zzekun@ubuntu:~/Vote$ python3 count_student.py
请输入参与方 id:1
Traceback (most recent call last):
  File "count_student.py", line 5, in <module>
    student_1_id.txt,student_2_id.txt,student_3_id.txt
NameError: name 'student_1_id' is not defined
zzekun@ubuntu:~/Vote$ python3 count_student.py
请输入参与方 id:1
zzekun@ubuntu:~/Vote$ python3 count_student.py
请输入参与方 id:2
zzekun@ubuntu:~/Vote$ python3 count_student.py
请输入参与方 id:3
zzekun@ubuntu:~/Vote$ python3 vote_counter.py
得票结果为: 3
zzekun@ubuntu:~/Vote$
```

至此，实验 3.2 已经完成了基本的理解与复现。

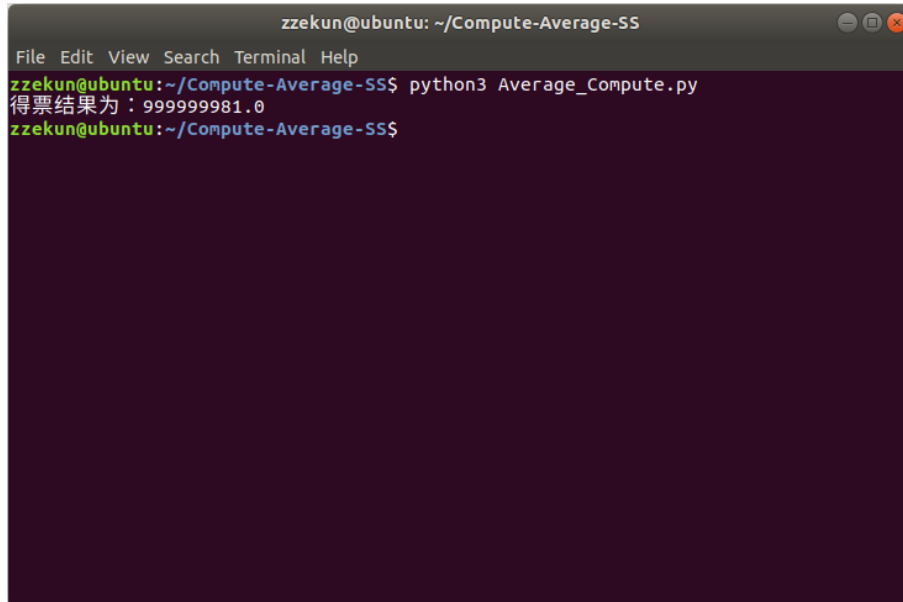
2. 实现计算三个人的平均值

对于计算三个人本地私有数据的平均值这个问题，在实验二的基础之上，我们三个策略。

- (1) 策略一：在最后解密得到的三个人之和的结果之后，直接将这个结果除以 3 即可得到这个问题的答案。
- (2) 策略二：在本机计算秘密值之时，直接将运算规则运用到本地秘密计算的过程也就是将本地的数据除以 3 之后再使用多项式函数封存。

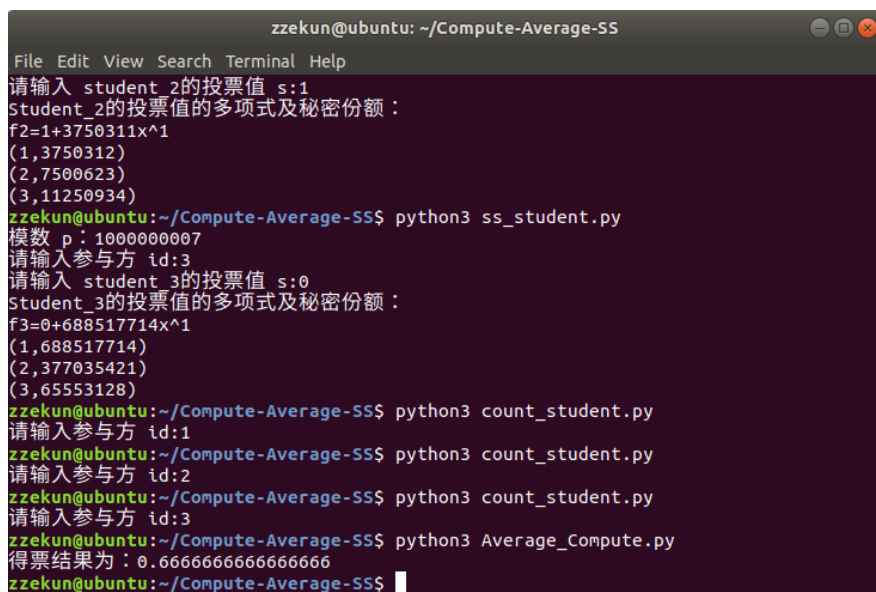
- (3) 策略三：在本地计算三方的秘密值之和时，应用运算规则除以 3，然后存储最终本地的 d 文件之中。

对于策略二和策略三而言，有一个避免不了的问题就是多项式重构计算的过程之中，无法避免的浮点数的精度不够的问题，这个问题会导致最后的解密的秘密值错误，所以除非是增加精度没有什么好的方法处理这个问题，最后的错误结果如下：



```
zzekun@ubuntu: ~/Compute-Average-SS
File Edit View Search Terminal Help
zzekun@ubuntu:~/Compute-Average-SS$ python3 Average_Compute.py
得票结果为：999999981.0
zzekun@ubuntu:~/Compute-Average-SS$
```

所以为了得到最正确的结果，应当使用策略一，直接在最后的求和结果之上除以 3 即可，实现的结果图如下：



```
zzekun@ubuntu: ~/Compute-Average-SS
File Edit View Search Terminal Help
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+3750311x^1
(1,3750312)
(2,7500623)
(3,11250934)
zzekun@ubuntu:~/Compute-Average-SS$ 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:0
Student_3 的投票值的多项式及秘密份额：
f3=0+688517714x^1
(1,688517714)
(2,377035421)
(3,65553128)
zzekun@ubuntu:~/Compute-Average-SS$ python3 count_student.py
请输入参与方 id:1
zzekun@ubuntu:~/Compute-Average-SS$ python3 count_student.py
请输入参与方 id:2
zzekun@ubuntu:~/Compute-Average-SS$ python3 count_student.py
请输入参与方 id:3
zzekun@ubuntu:~/Compute-Average-SS$ python3 Average_Compute.py
得票结果为：0.6666666666666666
zzekun@ubuntu:~/Compute-Average-SS$
```

3. 使用最新的 secretflow 框架实现本机三方计算平均值模拟

在 3 月底公布了最新的安全多方计算框架 secretflow，使用该框架可以十分简便的在本

机模拟，并且选择相应的秘密共享协议，具体实现的代码如下：

```
import secretflow as sf

# In case you have a running secretflow runtime already.
sf.shutdown()

# 本机初始化三方
sf.init(['Alice', 'Bob', 'Charles'], address='local')

# 创建基于 aby3 的 spu 单元
aby3_config = sf.utils.testing.cluster_def(parties=['Alice', 'Bob', 'Charles'])
spu_device = sf.SPU(aby3_config)

# 获取 num 的函数
def get_Alice_num():
    num = int(input("Alice input: "))
    return num

def get_Bob_num():
    num = int(input("Bob input: "))
    return num

def get_Charles_num():
    num = int(input("Charles input: "))
    return num

def average_compute(Alice, Bob, Charles):
    return (Alice + Bob + Charles) / 3

# 创建各方的本地处理单元
Alice, Bob, Charles = sf.PYU('Alice'), sf.PYU('Bob'), sf.PYU('Charles')
# 获取每一方对应的数值
Alice_num = Alice(get_Alice_num)()
Bob_num = Bob(get_Bob_num)()
Charles_num = Charles(get_Charles_num)()

average = spu_device(average_compute)(Alice_num, Bob_num, Charles_num)
print('Average: ', average)
average = sf.reveal(average)
print('Average: ', average)
```

- (1) 我们首先在本地创建参与计算的三方，使用 init 函数，也就是创建了 Alice, Bob, Charles 这三方，并且使用 ABY3 协议创建了秘密处理计算单元 SPU

- (2) 接着是为三方分别初始化对应的模拟本地计算单元 PYU，并且为每一方对应的数值定义输入函数，获取每一方输入的数值。
- (3) 最后只需要简单的调用 SPU 单元，输入计算的函数以及对应的参数，即可计算得到三方的平均值的秘密值，该秘密共享存储在 SPU 的三方计算方之中，只有当我们使用 reveal 函数后，即可得知最后的秘密值。

实现的结果如下：

```
问题 输出 调试控制台 终端 窗口
[Not connected to 127.0.0.1:40475 yet, server_id=0 [R3][E112]Not connected to 127.0.0.1:40475 yet, server_id=0
[SPURuntime pid=11347] 2023-04-12 12:56:43.393 [error] [context.cc:operator():132] connect to rank=1 failed with error [external/yacl/yacl/link/
c.cc:368] send, rpc failed=112, message=[E112]Not connected to 127.0.0.1:40475 yet, server_id=0 [R1][E112]Not connected to 127.0.0.1:40475 yet,
[Not connected to 127.0.0.1:40475 yet, server_id=0 [R3][E112]Not connected to 127.0.0.1:40475 yet, server_id=0
1
(_run pid=11247) INFO:jax_src.xla_bridge:Unable to initialize backend 'cuda': module 'jaxlib.xla_extension' has no attribute 'GpuAllocatorConf
(_run pid=11247) INFO:jax_src.xla_bridge:Unable to initialize backend 'rocm': module 'jaxlib.xla_extension' has no attribute 'GpuAllocatorConf
(_run pid=11247) INFO:jax_src.xla_bridge:Unable to initialize backend 'tpu': INVALID ARGUMENT: TpuPlatform is not available.
(_run pid=11247) INFO:jax_src.xla_bridge:Unable to initialize backend 'plugin': xla extension has no attributes named get_plugin_device_client
ith //tensorflow/compiler/xla/python:enable_plugin_device set to true (defaults to false) to enable this.
(_run pid=11247) WARNING:jax_src.xla_bridge:No GPU/TPU found, falling back to CPU. (Set TF_CPP_MIN_LOG_LEVEL=0 and rerun for more info.)
2
(_run pid=11248) INFO:jax_src.xla_bridge:Unable to initialize backend 'cuda': module 'jaxlib.xla_extension' has no attribute 'GpuAllocatorConf
(_run pid=11248) INFO:jax_src.xla_bridge:Unable to initialize backend 'rocm': module 'jaxlib.xla_extension' has no attribute 'GpuAllocatorConf
(_run pid=11248) INFO:jax_src.xla_bridge:Unable to initialize backend 'tpu': INVALID ARGUMENT: TpuPlatform is not available.
(_run pid=11248) INFO:jax_src.xla_bridge:Unable to initialize backend 'plugin': xla extension has no attributes named get_plugin_device_client
ith //tensorflow/compiler/xla/python:enable_plugin_device set to true (defaults to false) to enable this.
(_run pid=11248) WARNING:jax_src.xla_bridge:No GPU/TPU found, falling back to CPU. (Set TF_CPP_MIN_LOG_LEVEL=0 and rerun for more info.)
3
(_run pid=11421) INFO:jax_src.xla_bridge:Unable to initialize backend 'cuda': module 'jaxlib.xla_extension' has no attribute 'GpuAllocatorConf
(_run pid=11421) INFO:jax_src.xla_bridge:Unable to initialize backend 'rocm': module 'jaxlib.xla_extension' has no attribute 'GpuAllocatorConf
(_run pid=11421) INFO:jax_src.xla_bridge:Unable to initialize backend 'tpu': INVALID ARGUMENT: TpuPlatform is not available.
(_run pid=11421) INFO:jax_src.xla_bridge:Unable to initialize backend 'plugin': xla extension has no attributes named get_plugin_device_client
ith //tensorflow/compiler/xla/python:enable_plugin_device set to true (defaults to false) to enable this.
(_run pid=11421) WARNING:jax_src.xla_bridge:No GPU/TPU found, falling back to CPU. (Set TF_CPP_MIN_LOG_LEVEL=0 and rerun for more info.)
Average: <secretflow.device.device.spu.SPUObject object at 0x7f6f3add040>
Average: 2.0
[root@36a2019fb356 test]#
```

对于 warning 可以不予理会，这里是因为虚拟机上的 GPU 环境问题导致的。我们可以看到三方对应输入 1,2,3，最终可以得到结果在没有 reveal 之前显示为 SPUObject，在 reveal 之后得到平均值结果为 2.0

心得体会

通过本次实验，我们学习了秘密共享的原理，实验中给出了一个非常贴近现实的使用 (2,3)门陷方案的场景，我们完成了最终秘密值的 reveal，以及秘密值的计算与共享过程。

经过学习与实践，将在密码学课上学习的知识和数据安全学习的内容结合在了一起，更好的理解了秘密共享和 shamir 秘密共享方案的特性及其适合的应用场景。