南開大學

Nankai University

半同态加密应用实践实验报告

- 姓名：陈睿颖
- 学号：2013544
- 专业：计算机科学与技术

1. 实验要求

基于Paillier算法实现隐私信息获取:从服务器给定的m个消息中获取其中一个，不得向服务器泄露获取了哪一个消息，同时客户端能完成获取消息的解密。

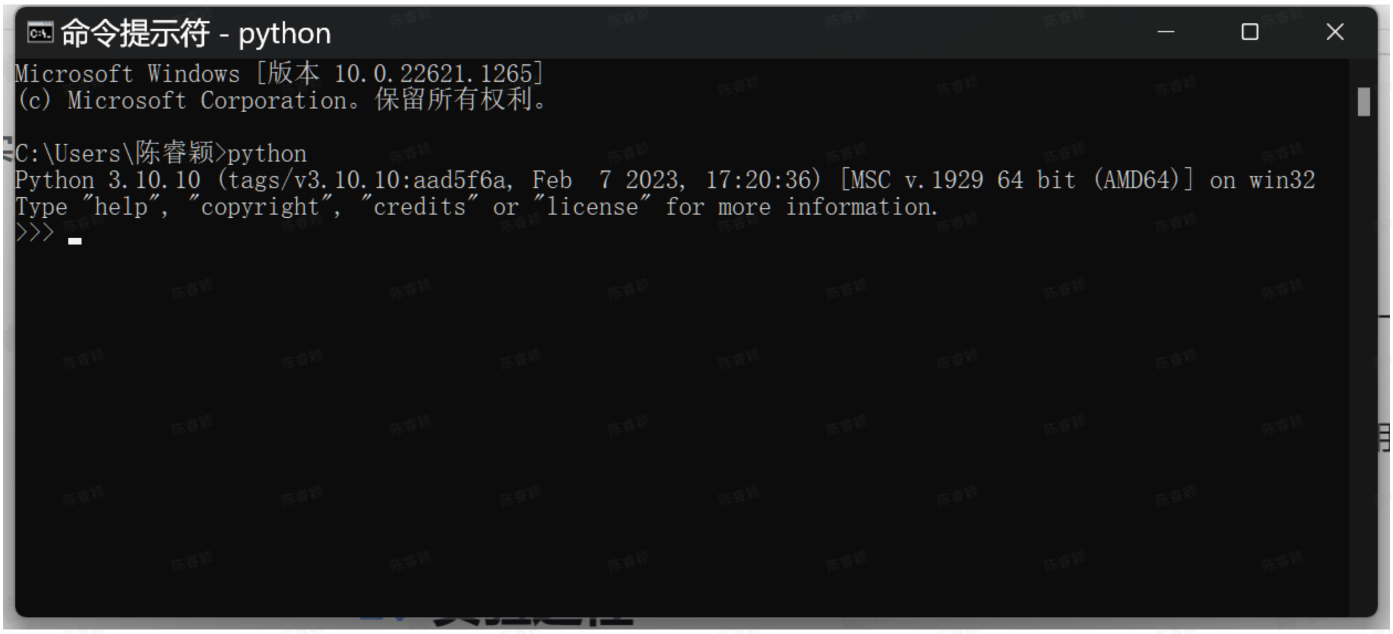
扩展实验:有能力的同学可以在客户端保存对称密钥k，在服务器端存储m个用对称密钥k加密的密文，通过隐私信息获取方法得到指定密文后能解密得到对应的明文。

2. 实验过程

2.1 安装环境

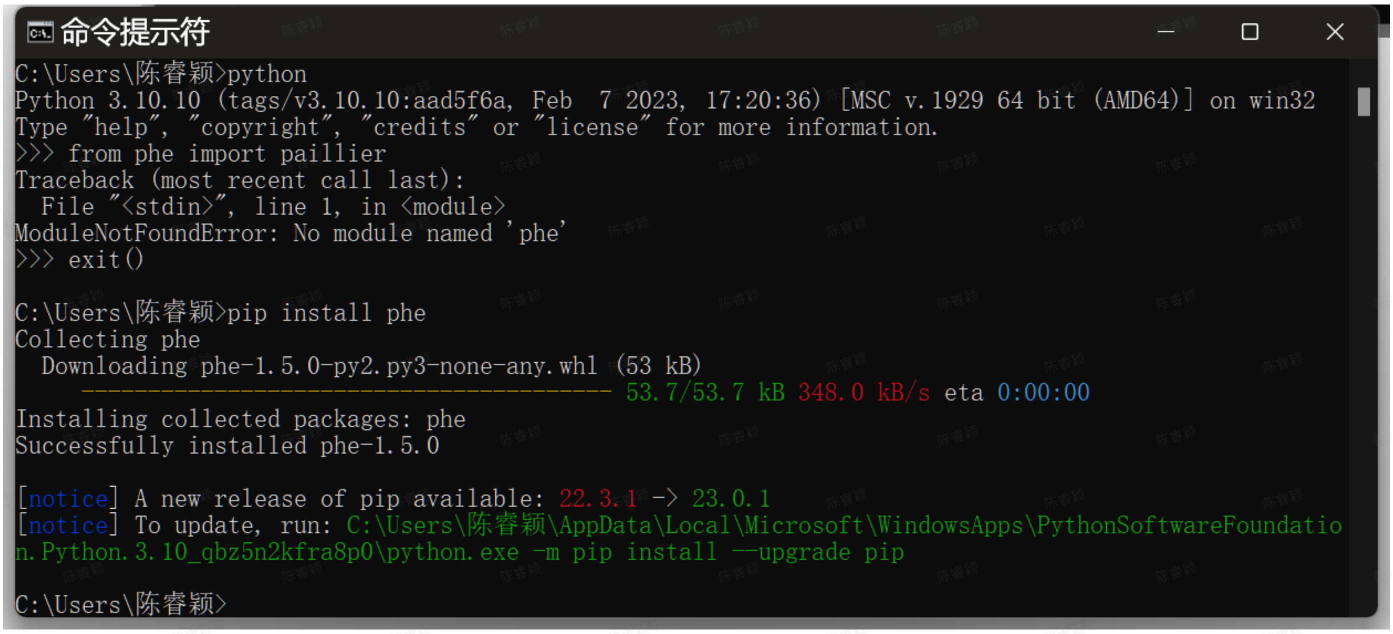
1. 安装python环境在 Windows 下安装 python 开发环境。到官方网站 https://www.python.org/downloads/下 载 windows 版本的 python 安装包。下载后双击安装即可。 提示：安装过程一定要勾选“Add python.exe to PATH”，这样会使得安装后的 python 程序路径直接加入到系统的环境变量中，在控制台可以直接使用 python 命令。如果忘记勾选，则需要通过“我的电脑”->右键“属性”->“高级系统设置”->“环境变量”的 path 中将安装的路径手动填入。

   安装完毕，打开控制台，输入 `python` 命令，会显示:



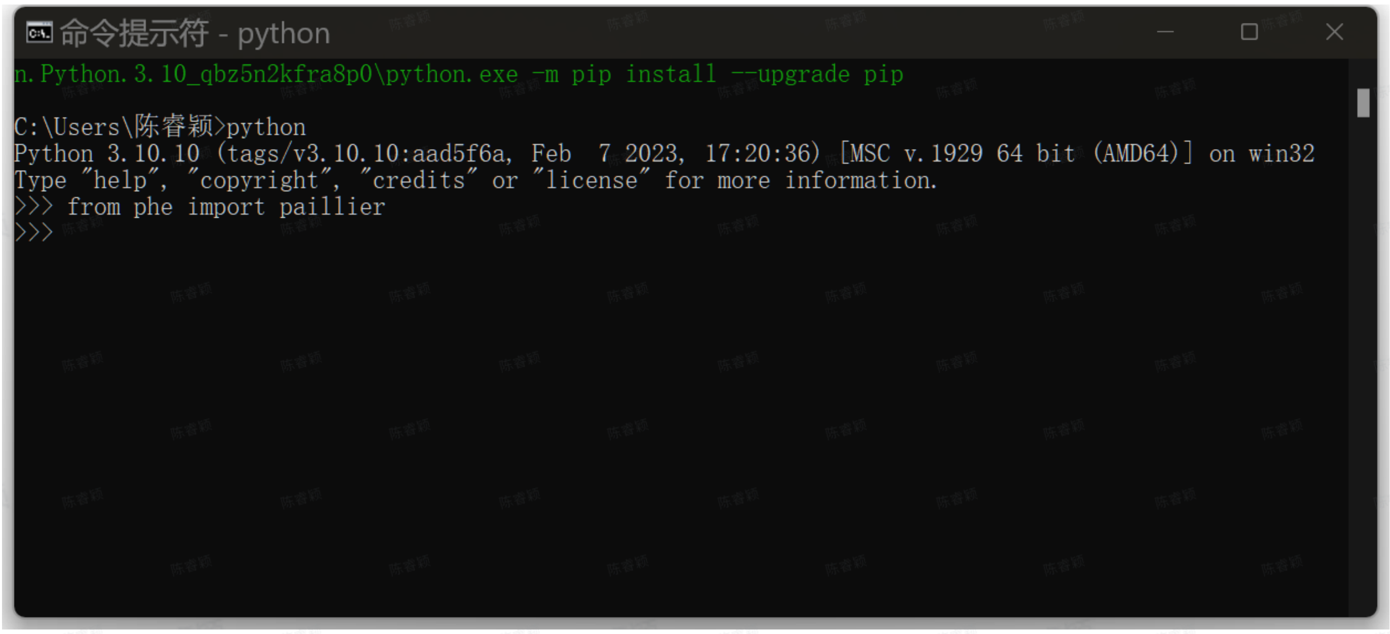
```
Microsoft Windows [版本 10.0.22621.1265]
(c) Microsoft Corporation。保留所有权利。

C:\Users\陈睿颖>python
Python 3.10.10 (tags/v3.10.10:aad5f6a, Feb  7 2023, 17:20:36) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

2. 安装 phe 库 输入命令： `pip install phe` 完成 phe 库的安装。



```
C:\Users\陈睿颖>python
Python 3.10.10 (tags/v3.10.10:aad5f6a, Feb  7 2023, 17:20:36) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from phe import paillier
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ModuleNotFoundError: No module named 'phe'
>>> exit()

C:\Users\陈睿颖>pip install phe
Collecting phe
  Downloading phe-1.5.0-py2.py3-none-any.whl (53 kB)
     ---------------------------------------- 53.7/53.7 kB 348.0 kB/s eta 0:00:00
Installing collected packages: phe
Successfully installed phe-1.5.0

[notice] A new release of pip available: 22.3.1 -> 23.0.1
[notice] To update, run: C:\Users\陈睿颖\AppData\Local\Microsoft\WindowsApps\PythonSoftwareFoundatio
n.Python.3.10_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip

C:\Users\陈睿颖>
```

3. 验证环境的正确性



```
n.Python.3.10_qbz5n2kfra8p0\python.exe -m pip install --upgrade pip

C:\Users\陈睿颖>python
Python 3.10.10 (tags/v3.10.10:aad5f6a, Feb  7 2023, 17:20:36) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from phe import paillier
>>>
```

没有报错，正确！

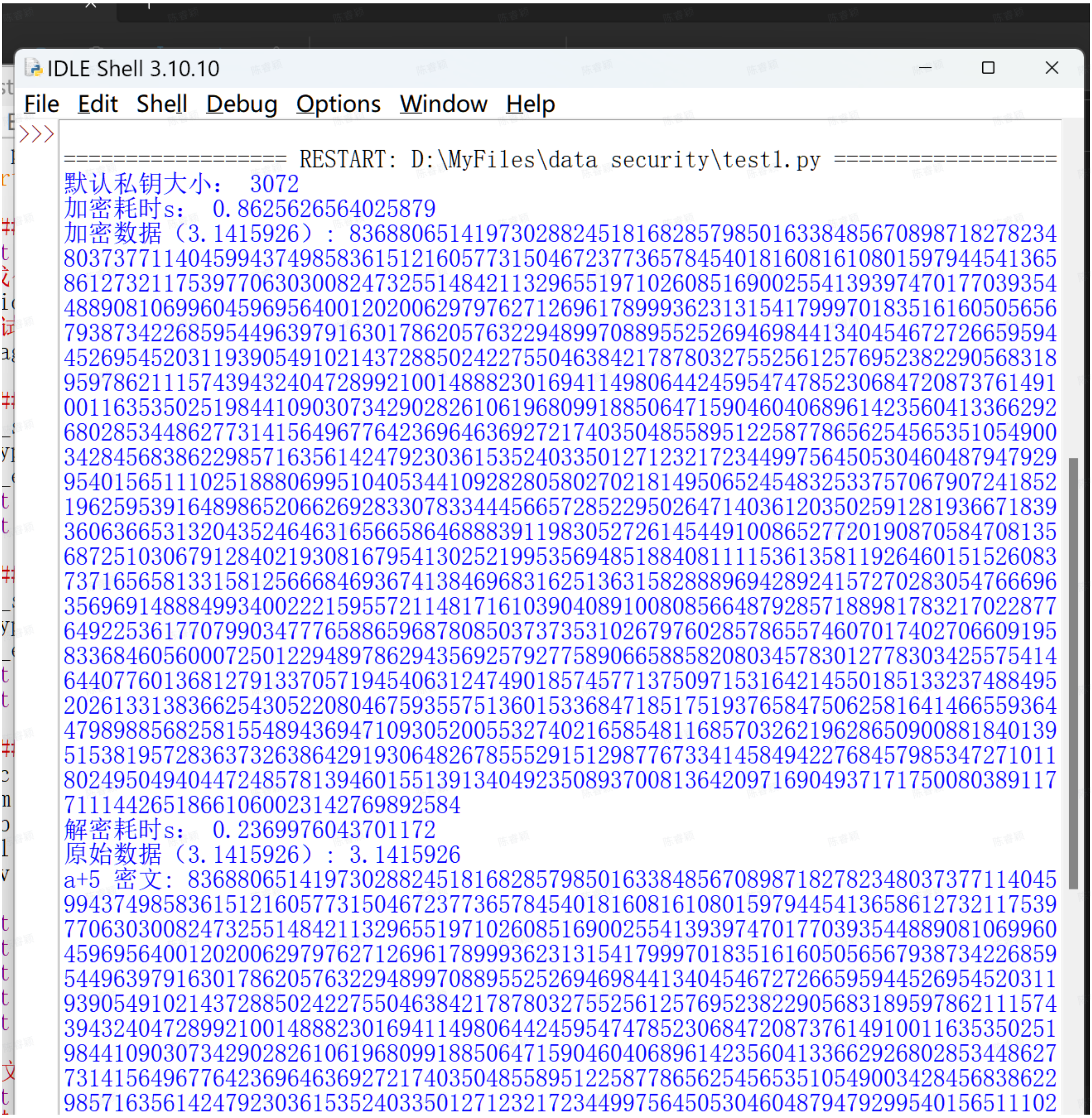
2.2 基于 Python 的 phe 库完成加法和标量乘法的验证

使用的代码如下：

```python
from phe import paillier # 开源库
import time # 做性能测试

##################### 设置参数
print("默认私钥大小：", paillier.DEFAULT_KEYSIZE)
#生成公私钥
public_key, private_key = paillier.generate_paillier_keypair()
# 测试需要加密的数据
message_list = [3.1415926,100,-4.6e-12]

##################### 加密操作
time_start_enc = time.time()
encrypted_message_list = [public_key.encrypt(m) for m in message_list]
time_end_enc = time.time()
print("加密耗时：",time_end_enc-time_start_enc)
print("加密数据（3.1415926）：",encrypted_message_list[0].ciphertext())

##################### 解密操作
time_start_dec = time.time()
decrypted_message_list = [private_key.decrypt(c) for c in encrypted_message_list
time_end_dec = time.time()
print("解密耗时：",time_end_dec-time_start_dec)
print("原始数据（3.1415926）：",decrypted_message_list[0])

##################### 测试加法和乘法同态
```

```python
a,b,c = encrypted_message_list # a,b,c分别为对应密文
a_sum = a + 5 # 密文加明文，已经重载了+运算符
a_sub = a - 3 # 密文加明文的相反数，已经重载了-运算符
b_mul = b * 6 # 密文乘明文,数乘
c_div = c / -10.0 # 密文乘明文的倒数

print("a+5 密文:",a.ciphertext()) # 密文纯文本形式
print("a+5=",private_key.decrypt(a_sum))
print("a-3",private_key.decrypt(a_sub))
print("b*6=",private_key.decrypt(b_mul))
print("c/-10.0=",private_key.decrypt(c_div))

##密文加密文
print((private_key.decrypt(a)+private_key.decrypt(b))==private_key.decrypt(a+b))
#报错，不支持a*b，即两个密文直接相乘
#print((private_key.decrypt(a)+private_key.decrypt(b))==private_key.decrypt(a*b)

```

运行结果如图所示：



```
>>>
================== RESTART: D:\MyFiles\data security\test1.py ==================
默认私钥大小:  3072
加密耗时s:  0.8625626564025879
加密数据（3.1415926）: 836880651419730288245181682857985016338485670898718278234
80373771140459943749858361512160577315046723773657845401816081610801597944541365
86127321175397706303008247325514842113296551971026085169002554139397470177039354
48890810699604596956400120200629797627126961789993623131541799970183516160505656
79387342268595449639791630178620576322948997088955252694698441340454672726659594
45269545203119390549102143728850242275504638421787803275525612576952382290568318
95978621115743943240472899210014888230169411498064424595474785230684720873761491
00116353502519844109030734290282610619680991885064715904604068961423560413366292
68028534486277314156496776423696463692721740350485589512258778656254565351054900
34284568386229857163561424792303615352403350127123217234499756450530460487947929
95401565111025188806995104053441092828058027021814950652454832533757067907241852
19625953916489865206626928330783344456657285229502647140361203502591281936671839
36063665313204352464631656658646888391198305272614544910086527720190870584708135
68725103067912840219308167954130252199535694851884081111536135811926460151526083
73716565813315812566684693674138469683162513631582888969428924157270283054766696
35696914888499340022215955721148171610390408910080856648792857188981783217022877
64922536177079903477765886596878085037373531026797602857865574607017402706609195
83368460560007250122948978629435692579277589066588582080345783012778303425575414
64407760136812791337057194540631247490185745771375097153164214550185133237488495
20261331383662543052208046759355751360153368471851751937658475062581641466559364
47989885682581554894369471093052005532740216585481168570326219628650900881840139
51538195728363732638642919306482678555291512987767334145849422768457985347271011
80249504940447248578139460155139134049235089370081364209716904937171750080389117
71114426518661060023142769892584
解密耗时s:  0.2369976043701172
原始数据（3.1415926）: 3.1415926
a+5 密文: 8368806514197302882451816828579850163384856708987182782348037377114045
99437498583615121605773150467237736578454018160816108015979445413658612732117539
77063030082473255148421132965519710260851690025541393974701770393544889081069960
45969564001202006297976271269617899936231315417999701835161605056567938734226859
54496397916301786205763229489970889552526946984413404546727266595944526954520311
93905491021437288502422755046384217878032755256125769523822905683189597862111574
39432404728992100148882301694114980644245954747852306847208737614910011635350251
98441090307342902826106196809918850647159046040689614235604133662926802853448627
73141564967764236964636927217403504855895122587786562545653510549003428456838622
98571635614247923036153524033501271232172344997564505304604879479299540156511102
```

2.3 基于 Python 的 phe 库完成隐私信息获取的功能

服务器端拥有多个数值，要求客户端能基于 Paillier 实现从服务器读取一个指定的数值并正确解密，但服务器不知道所读取的是哪一个。

基于 Paillier 协议进行设计：由于数值“0”的密文与任意数值的标量乘也是 0，数值“1”的密文与任意数值的标量乘将是数值本身。

设计如下：

> 服务器端：产生数据列表 data_list={m1, m2, ⋯, mn}
>
> 客户端：

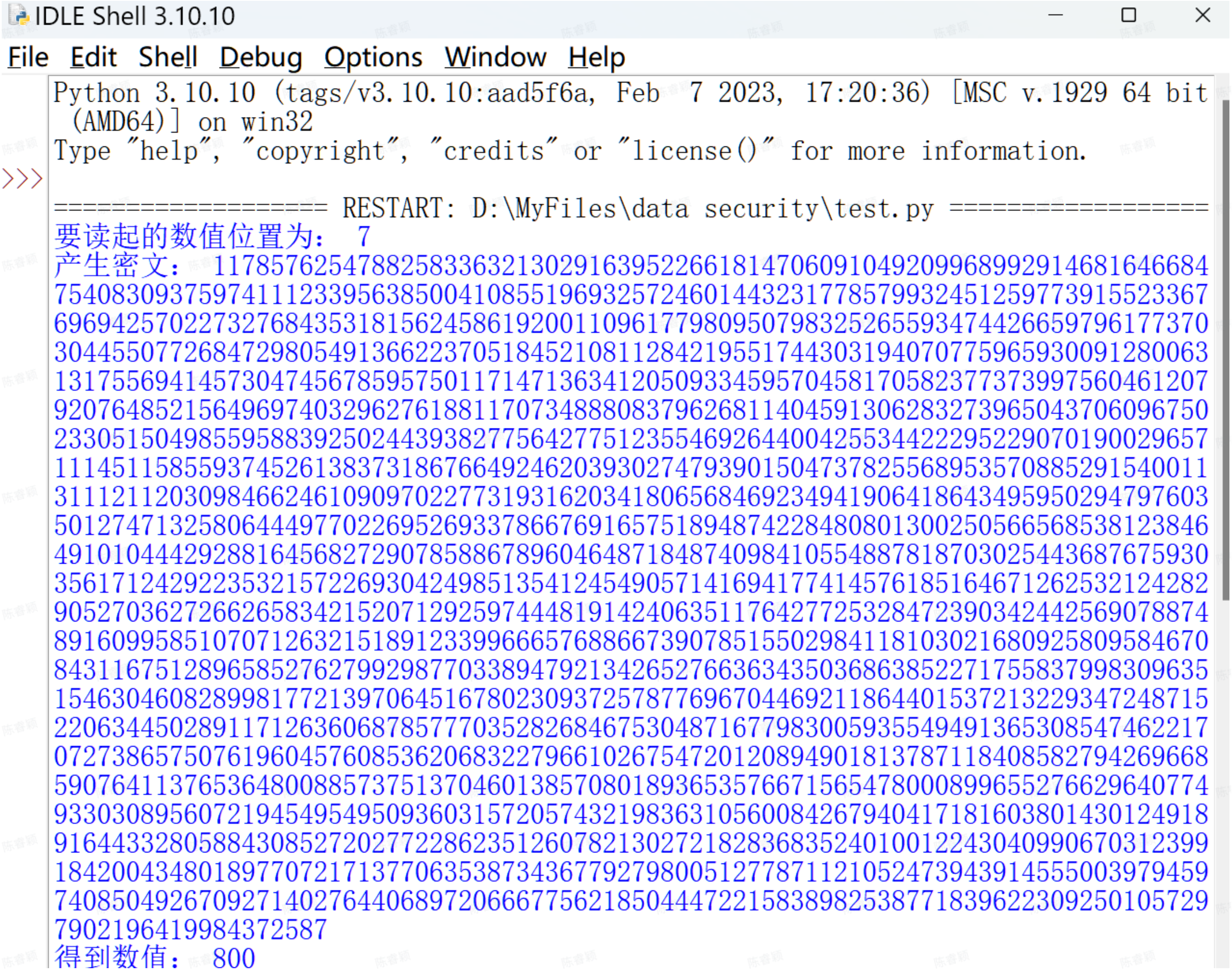
> - 设置要选择的数据位置为 pos
> - 生成选择向量 select_list={0,…,1,.., 0}，其中，仅有 pos 的位置为 1
> - 生成密文向量 enc_list={E(0),…, E(1),.., E(0)}
> - 发送密文向量 enc_list 给服务器
>
> 服务器端：
>
> - 将数据与对应的向量相乘后累加得到密文 c= m1*enc_list[1]+…+ mn*enc_list[n]
> - 返回密文 c 给客户端
>
> 客户端：解密密文 c 得到想要的结果

具体代码实现如下：

```python
from phe import paillier # 开源库
import random # 选择随机数

#################### 设置参数
# 服务器端保存的数值
message_list = [100,200,300,400,500,600,700,800,900,1000]
length = len(message_list)
# 客户端生成公私钥
public_key, private_key = paillier.generate_paillier_keypair()
# 客户端随机选择一个要读的位置
pos = random.randint(0,length-1)
print("要读起的数值位置为：",pos)

#################### 客户端生成密文选择向量
select_list=[]
enc_list=[]
for i in range(length):
    select_list.append(  i == pos )
    enc_list.append( public_key.encrypt(select_list[i]) )

# for element in select_list:
#     print(element)
# for element in enc_list:
#     print(private_key.decrypt(element))

#################### 服务器端进行运算
c=0
for i in range(length):
    c = c + message_list[i] * enc_list[i]
print("产生密文：",c.ciphertext())
```

```

##################### 客户端进行解密
m=private_key.decrypt(c)
print("得到数值：",m)

```

如图所示：



```
Python 3.10.10 (tags/v3.10.10:aad5f6a, Feb  7 2023, 17:20:36) [MSC v.1929 64 bit
 (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:\MyFiles\data security\test.py ==================
要读起的数值位置为：  7
产生密文： 117857625478825833632130291639522661814706091049209968992914681646684
754083093759741112339563850041085519693257246014432317785799324512597739155233676
969425702273276843531815624586192001109617798095079832526559347442665979617737030
445507726847298054913662237051845210811284219551744303194070775965930091280063
131755694145730474567859575011714713634120509334595704581705823773739975604612079
207648521564969740329627618811707348880837962681140459130628327396504370609675023
305150498559588392502443938277564277512355469264400425534422295229070190029657
111451158559374526138373186766492462039302747939015047378255689535708852915400113
111211203098466246109097022773193162034180656846923494190641864349595029479760350
127471325806444977022695269337866769165751894874228480801300250566568538123846
491010444292881645682729078588678960464871848740984105548878187030254436876759303
561712429223532157226930424985135412454905714169417741457618516467126253212428290
527036272662658342152071292597444819142406351176427725328472390342442569078874
891609958510707126321518912339966657688667390785155029841181030216809258095846708
431167512896585276279929877033894792134265276636343503686385227175583799830963515
463046082899817721397064516780230937257877696704469211864401537213229347248715
220634450289117126360687857770352826846753048716779830059355494913653085474622170
727386575076196045760853620683227966102675472012089490181378711840858279426966859
076411376536480088573751370460138570801893653576671565478000899655276629640774
933030895607219454954950936031572057432198363105600842679404171816038014301249189
164433280588430852720277228623512607821302721828368352401001224304099067031239918
420043480189770721713770635387343677927980051277871121052473943914555003979459
740850492670927140276440689720666775621850444722158389825387718396223092501057297
902196419984372587
得到数值： 800
```

2.4 扩展实验

在客户端保存对称密钥k，在服务器端存储m个用对称密钥k加密的密文，通过隐私信息获取方法得到指定密文后能解密得到对应的明文。

首先要使用如下命令安装：

```
pip3 install pycryptodome

pip3 install crypto

```

```
pip3 install pycrypto

```

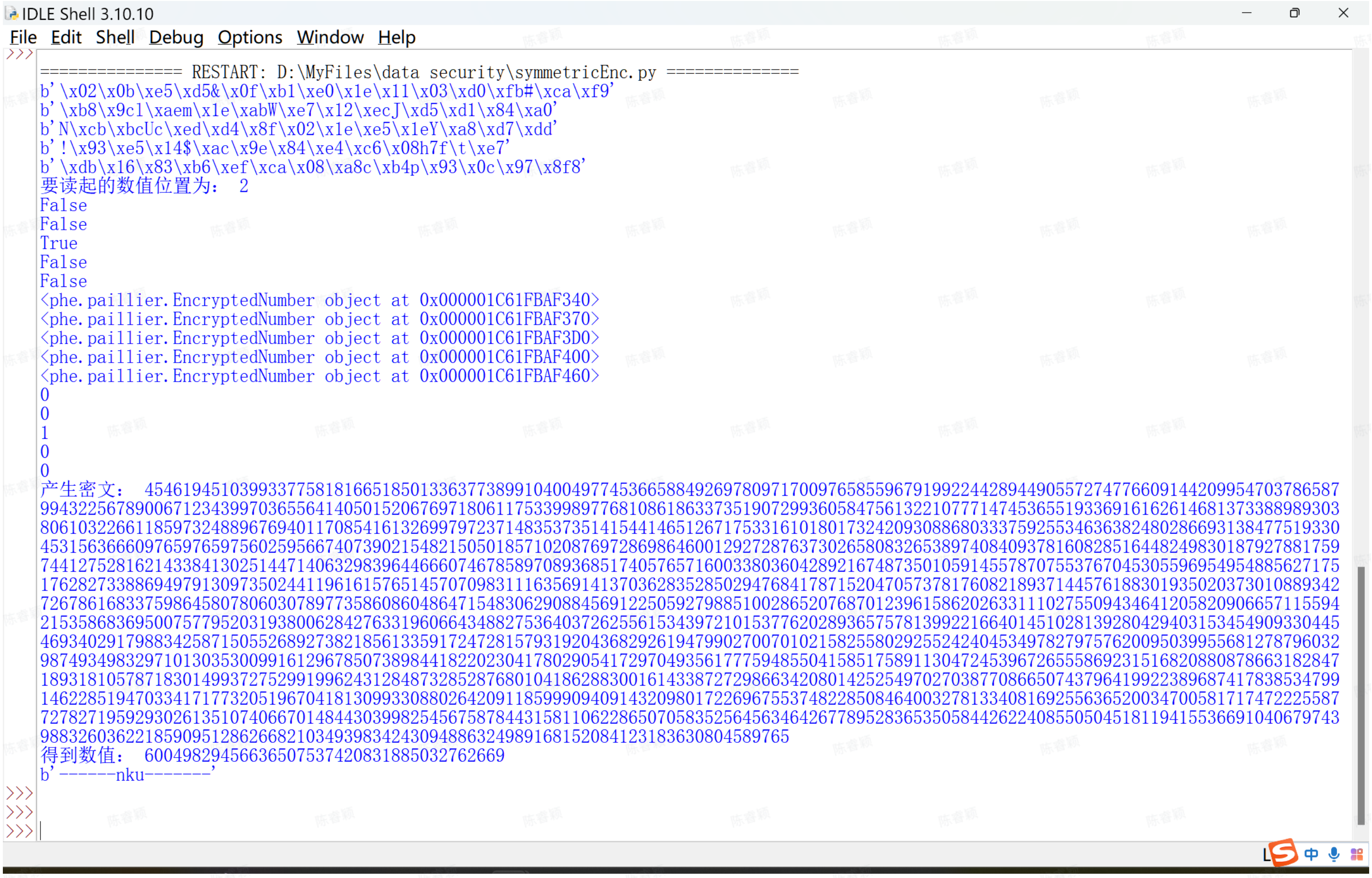
python3的PyCryptodome库用于密码学，是一个低级密码基元的独立Python包，常见对称密码在Crypto.Cipher 库下，主要有：DES 3DES AES RC4 Salsa20，这次我们使用的是AES。

可参照前面两个实验，仿照编写代码如下：

```python
from phe import paillier  # 开源库
from Crypto.Cipher import AES
import random  # 选择随机数

##################### 设置参数
# 服务器端保存的数值
plaintext_list = [
    b'0123456789abcdef',
    b'qwertyuiopasdfgh',
    b'------nku-------',
    b'thisisalabreport',
    b'**chenruiying** '
]
ciphertext_list = []
ciphernum_list = []
symmetric_key = b'------nku-------'
aes = AES.new(symmetric_key, AES.MODE_ECB)  # 创建一个aes对象
for text in plaintext_list:
    ciphertext_list.append(aes.encrypt(text))  # 加密明文
    ciphernum_list.append(int.from_bytes(text, byteorder='big'))

for element in ciphertext_list:
    print(element)

length = len(ciphertext_list)
# 客户端生成公私钥
public_key, private_key = paillier.generate_paillier_keypair()
# 客户端随机选择一个要读的位置
pos = random.randint(0, length - 1)
print("要读起的数值位置为：", pos)

##################### 客户端生成密文选择向量
select_list = []
enc_list = []
for i in range(length):
    select_list.append(i == pos)
    enc_list.append(public_key.encrypt(select_list[i]))
```

```python

for element in select_list:
    print(element)
for element in enc_list:
    print(element)
for element in enc_list:
    print(private_key.decrypt(element))

##################### 服务器端进行运算
c = 0
for i in range(length):
    c = c + ciphernum_list[i] * enc_list[i]
print("产生密文: ", c.ciphertext())

##################### 客户端进行解密
m = private_key.decrypt(c)
print("得到数值: ", m)

message = m.to_bytes(16, byteorder='big')
x = aes.decrypt(message)
print(message)

```

运行结果如下：



```
>>>
=============== RESTART: D:\MyFiles\data security\symmetricEnc.py ==============
b'\x02\x0b\xe5\xd5&\x0f\xb1\xe0\x1e\x11\x03\xd0\xfb#\xca\xf9'
b'\xb8\x9c1\xaem\x1e\xabW\xe7\x12\xecJ\xd5\xd1\x84\xa0'
b'N\xcb\xbcUc\xed\xd4\x8f\x02\x1e\xe5\x1eY\xa8\xd7\xdd'
b'!\x93\xe5\x14$\xac\x9e\x84\xe4\xc6\x08h7f\t\xe7'
b'\xdb\x16\x83\xb6\xef\xca\x08\xa8c\xb4p\x93\x0c\x97\x8f8'
要读起的数值位置为：  2
False
False
True
False
False
<phe.paillier.EncryptedNumber object at 0x000001C61FBAF340>
<phe.paillier.EncryptedNumber object at 0x000001C61FBAF370>
<phe.paillier.EncryptedNumber object at 0x000001C61FBAF3D0>
<phe.paillier.EncryptedNumber object at 0x000001C61FBAF400>
<phe.paillier.EncryptedNumber object at 0x000001C61FBAF460>
0
0
1
0
0
产生密文：  454619451039933775818166518501336377389910400497745366588492697809717009765855967919922442894490557274776609144209954703786587
99432256789006712343997036556414050152067697180611753399897768108618633735190729936058475613221077714745365519336916162614681373388989303
80610322661185973248896769401170854161326997972371483537351415441465126717533161018017324209308868033375925534636382480286693138477519330
45315636660976597659756025956674073902154821505018571020876972869864600129272876373026580832653897408409378160828516448249830187927881759
74412752816214338413025144714063298396446660746785897089368517405765716003380360428921674873501059145578707553767045305596954954885627175
17628273388694979130973502441196161576514570709831116356914137036283528502947684178715204705737817608218937144576188301935020373010889342
72678616833759864580780603078977358608604864715483062908845691225059279885100286520768701239615862026331110275509434641205820906657115594
21535868369500757795203193800628427633196066434882753640372625561534397210153776202893657578139922166401451028139280429403153454909330445
46934029179883425871505526892738218561335917247281579319204368292619479902700701021582558029255242404534978279757620095039955681278796032
98749349832971013035300991612967850738984418220230417802905417297049356177759485504158517589113047245396726555869231516820880878663182847
18931810578718301499372752991996243128487328528768010418628830016143387272986634208014252549702703877086650743796419922389687417838534799
14622851947033417177320519670418130993308802642091185999094091432098017226967553748228508464003278133408169255636520034700581717472225587
72782719592930261351074066701484430399825456758784431581106228650705835256456346426778952836535058442622408550504518119415536691040679743
9883260362218590951286266821034939834243094886324989168152084123183630804589765
得到数值：  60049829456636507537420831885032762669
b'------nku-------'
>>>
>>>
>>>
```

解密完成！

3. 心得体会

- 复习了python环境的安装，学习了phe库的安装
- 在实验2.2中，学会了基于 Paillier 协议进行设计：由于数值“0”的密文与任意数值的标量乘也是0，数值“1”的密文与任意数值的标量乘将是数值本身
- 尝试完成了扩展实验，了解了包PyCryptodome，学会了其安装及使用方法