

# 数据安全 -- 半同态加密应用实践

学号：2212452 姓名：孟启轩 专业：计算机科学与技术

## 一、实验要求

- 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。

## 二、实验原理

### （一）半同态加密

半同态加密基于公钥密码体制，允许第三方对密文直接进行特定类型的运算（如加法或乘法），且运算结果解密后与明文直接运算的结果一致。其核心是仅支持单一类型的同态运算（如Paillier加密仅支持加法同态和标量乘法），通过加密后的密文保留运算特性，实现“数据可用不可见”。例如，Paillier加密的密文支持同态加法（ $Enc(a) + Enc(b) = Enc(a+b)$ ）和标量乘法（ $Enc(a) * k = Enc(a*k)$ ），但无法直接进行密文相乘，这使其在隐私保护场景中既能支持基础计算，又能避免密钥泄露风险。

### （二）隐私信息获取

隐私信息获取通过Paillier的标量乘特性实现：数值“0”的密文与任意数值的标量乘结果仍为0，数值“1”的密文与任意数值的标量乘结果为该数值本身。客户端将数据加密后发送至服务器，服务器仅对密文执行同态运算（如加法或标量乘），无需接触明文；最终结果返回客户端解密，确保服务器无法获取原始数据。例如，若客户端需获取某数值，可让服务器对加密后的数据执行同态加法或标量乘，结果解密后即所需信息，全程保护客户端隐私，符合“计算在密文上，结果由密钥方控制”的安全模型。

### （三）移位密码

移位密码（凯撒密码）是一种简单的对称加密算法，通过固定位移量（密钥）对明文进行偏移加密。加密时，明文字符按密钥值向后移动（如密钥为3，字符'A'变为'D'）；解密时，密文字符向前移动相同位移量还原明文。其原理基于明文与密钥的简单加减运算，但由于密钥空间小、模式易被破解，安全性极低。在实验中，移位密码通常用于演示基础加密思想，或作为对比案例，说明现代同态加密（如Paillier）在隐私保护与计算能力上的优势。

## 三、实验过程

### （一）实验环境安装

- 已提前安装好 Python 环境，此处仅展示安装后的运行效果。
- 在进入 Python 环境后，执行 `from phe import paillier` 命令，报错提示缺少 phe 库。退出python环境，在命令行中执行 `pip install phe` 命令完成 phe 库的安装。
- 导入 phe 库，环境安装成功。

```
C:\WINDOWS\system32\cmd. x + v
Microsoft Windows [版本 10.0.26100.3194]
(c) Microsoft Corporation。保留所有权利。

C:\Users\lenovo>python
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 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\lenovo>pip install phe
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple/
Collecting phe
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/53/7c/1c514f3e030ff69ee2a184fca3f1514c1d32653ca00869d884b4f981e564/phe-1.5.0-py2.py3-none-any.whl (53 kB)
Installing collected packages: phe
Successfully installed phe-1.5.0

[notice] A new release of pip is available: 24.3.1 -> 25.0.1
[notice] To update, run: python.exe -m pip install --upgrade pip

C:\Users\lenovo>python
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from phe import paillier
>>> |
```

## (二) 基于 Python 的 phe 库完成加法和标量乘法的验证

代码通过Paillier同态加密库（**phe**）验证了加密、解密及同态运算的核心功能，具体分析如下：

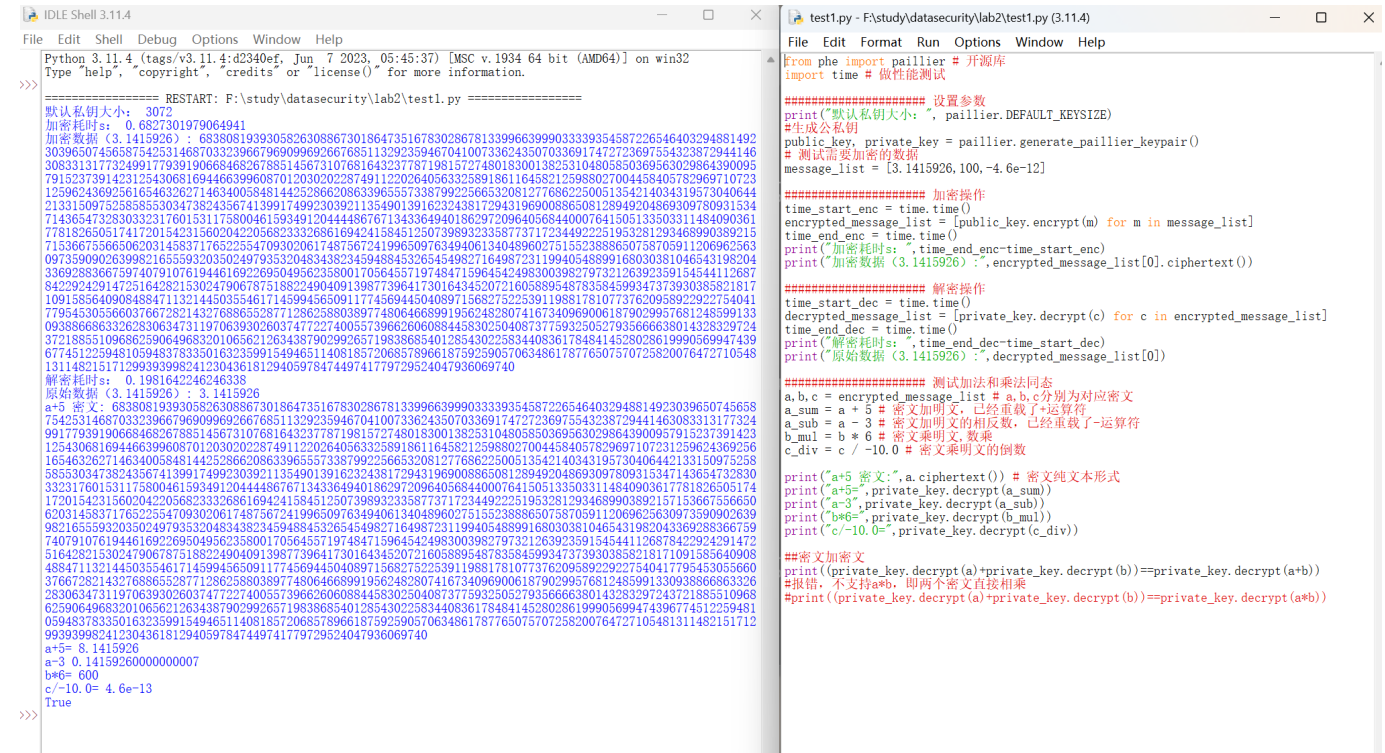
### 1. 密钥生成与加密解密验证

- **密钥生成**：使用`generate_paillier_keypair()`生成公私钥对，为后续加密和解密操作提供基础。
- **加密测试**：对包含浮点数（如`3.1415926`）、整数（如`100`）和科学计数法（如`-4.6e-12`）的明文列表进行加密，记录加密耗时，并输出密文的纯文本形式（如`encrypted_message_list[0].ciphertext()`）。
- **解密验证**：解密密文并验证结果是否与原始数据一致，同时记录解密耗时。例如，解密后的`3.1415926`需与加密前完全匹配，以确保加密-解密流程的正确性。

### 2. 同态运算测试

Paillier加密支持**加法同态**和**标量乘法**（明文与密文的运算），但不支持**密文与密文的乘法**。

- **密文与明文的运算**
  - **加法/减法**：如 $a + 5$ （密文 $a$ 对应明文`3.1415926`，解密后结果为`8.1415926`）； $a - 3$ 解密后为`0.1415926`。
  - **标量乘法/除法**：如 $b * 6$ （密文 $b$ 对应明文`100`，解密后结果为`600`）； $c / -10.0$ （密文 $c$ 对应明文`-4.6e-12`，解密后为`4.6e-13`）。
- **密文与密文的加法**
  - 验证 $a + b$ 的同态加法：解密后结果应为明文之和（`3.1415926 + 100`），代码输出`True`，表明同态加法有效。
- **密文与密文的乘法**
  - 尝试 $a * b$ 会引发错误，因Paillier加密仅支持加法同态，无法直接进行密文相乘（需更复杂的加密方案如BGN或BFV）。



```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type 'help', 'copyright', 'credits' or 'license()' for more information.

>>>
===== RESTART: F:\study\datsasecurity\lab2\test1.py =====
默认私钥大小: 3072
加密数据 (3, 1415926): 6838081939305826308867301864735167830286781339966399903339354587226546403294881492
3039650745658754253146870332396679690996926676851132923594670410073362435070336917472723697554323872944146
3083313177324991799190668468267885145673107681643237787198157274801830013825310480585036956302964390095
791523739142312543068169446639960870120302022874911202640563325891861164582125988027004458405782969710723
1259624369251654632627146340558414425286620633965557338799225665320812776862250051354214034319573040644
21331509752585853034738243567413991749923520313549013916232438172943196900886508128949204869309780931534
714365473283032317601531175800461593491204444867671343364940186297209640568440007641505133503311484903061
77818265051741720154231560242205682332686169424158451250739893233587737172344922251953281293468990389215
715366753665062031458371763225547093020617487367241996097634940613404899602751552388650758705911206962563
097359090263982165559320305024979353204834382345948843265454982716498723119940548899168030381046543198204
336928336675974079107619461692269504965235800170564557197484715964542498300398279732126392359154544112687
842924291472516428215302479067875188224904091398773964173016434520721605889548783584599347373930385821817
109158564090848471132144503554617145994565091177456944504089715682752253911988178107737620958922922754041
77954530556603766728214327688652871286258803897748064668991956248280741673409690061879029957681248599133
0938666633262830634731197063930260374722274005573966260608944583025040873775925052793566638014328329724
3721885510968625906496832010656212634387902992657198386854012854302258344083617848414328028619990569947439
6774512259481059483783350163235991549465114081857206857896618759259057063486178776507570725820076472710548
13114821517129939399824123043618129405978474974179729524047936069740
解密耗时: 0.1981642246246338
原始数据 (3, 1415926): 3, 1415926
a=5 密文: 68380819393058263088673018647351678302867813399663999033393545872265464032948814923039650745658
7542531468703323966796909969266768511329235946704100733624350703369174727236975543238729441463083313177324
991779391906684682678851456731076816432377871981572748018300138253104805850369563029643900957915237391423
1254306816944663996087012030202287491122026405633258918611645821259880270044584057829697107231259624369256
16546326271463405848144252866206339655573387992256653208127768622500513542140343195730406442133150975258
5855303473824356741399174992303921135490139162324381729431969008865081289492048693097809315347143654732830
3323176015311758004615934912044448676713433649401862972096405684400076415051335033114849030617781826505174
172015423156020422056823326861694241584512507398932335877371723449222519532812934689903892157153667556650
620314583717652255470930206174875672419965097634940613404896027515523886507587059112069625630973590902639
16255932035024979353204834382359488453265454982716498723119940548891680303810465431982043369288366759
7407910761944616922695049563358001705645571974847159645424983003982797321263923591545441126878422924291472
5164282153024790678751882249040913987739641730164345207216058895487835845993473739303858218171091585640908
4884711321445035546171459945650911774569445040897156827522539119881781077376209589229227540417795453055660
37667282143276886528712862588038977480646689919562482807416734096900618790299576812485991330938866863326
28306247311970639302603747727400557396620608844583025040873775932505279356666380143283297243721885510968
6259064968320106562126343879029926571983868540128543022583440836178484145280286199905699474396774512259481
0594837833501632359915494651140818572068578966187592590570634861787765075707258200764727105481311482151712
99393998241230436181294059784749741779729524047936069740
a=5 密文: 3, 1415926
a=3 0.14159260000000007
b*=600
c/-10.0 4.6e-13
True
>>>
```

### (三) 隐私信息获取

客户端通过加密的选择向量匿名指定数据位置，服务器对加密向量与数据同态运算后返回密文结果，客户端解密即可获取目标值，全程无需暴露选择位置或原始数据，实现隐私保护下的信息获取。具体分析如下：

#### 1. 参数设置与密钥生成

- 服务器端数据：预存数值列表`message_list`。
- 客户端操作：
  - 生成Paillier公私钥对 (`public_key, private_key`)。
  - 随机选择目标位置：`pos`为0到9之间的随机整数，表示客户端想读取的数值位置。

#### 2. 客户端生成密文选择向量

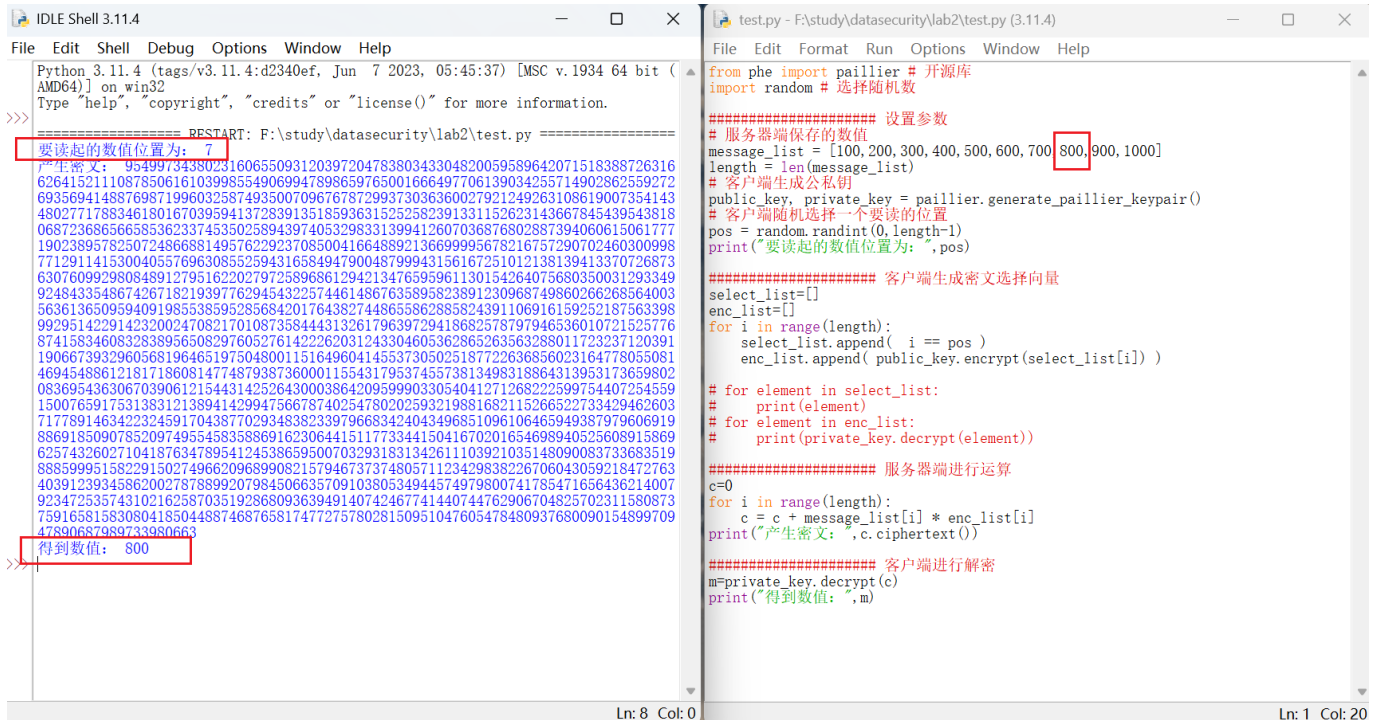
- 构造选择向量：创建布尔列表`select_list`，其中：
  - `select_list[i] = True`（即数值1）当且仅当`i == pos`；
  - 其他位置为`False`（即数值0）。
- 加密选择向量：将`select_list`中的每个元素（0或1）加密为密文列表`enc_list`，确保服务器无法直接读取选择的位置。

#### 3. 服务器端同态运算

- 加密的“选择性求和”：服务器端对每个元素执行`c += message_list[i] * enc_list[i]`
  - 数学本质：
    - 当`i == pos`时，`enc_list[i]`是加密的1，因此`message_list[i] * enc_list[i]`加密了`message_list[i]`本身。
    - 其他位置的`enc_list[i]`是加密的0，因此`message_list[i] * enc_list[i]`加密了0。
  - 结果：最终密文`c`加密了`message_list[pos]`（所有项的同态求和结果）。

#### 4. 客户端解密获取结果

- 客户端用私钥`private_key`解密密文`c`，得到明文结果`m = message_list[pos]`，从而在不暴露选择位置和原始数据的前提下，完成隐私保护的数据读取。



```
Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32
Type "help()", "copyright()", "credits()" or "license()" for more information.
>>>
===== RESTART: F:\study\datasecurity\lab2\test.py =====
要读取的数值位置为: 7
产生密文: 954997343802316065509312039720478380343304820059589642071518388726316
62641521110878506161039985549069947898659765001666497706139034255714902862559272
69356941488769871996032587493500709676787299373036360027921249263108619007354143
48027717883461801670395941372839135185936315252582391331152623143667845439543818
0687236865665853623745350258943974053298331399412607036876802887394060615061777
19023895782507248668814957622923708500416648892136699995678216757290702460300998
77129114153004055769630855259431658494790048799943156167251012138139413370726873
63076099298084891279516220279725896861294213476595961130154264075680350031293349
92484335486742671821939776294543225744614867635895823891230968749860266268564003
56361365095940919855385952856842017643827448655862885824391106916159252187563398
99295142291423200247082170108735844431326179639729418682578797946536010721525776
87415834608328389565082976052761422262031243304605362865263563288011723237120391
19066739329605681964651975048001151649604145537305025187722636856023164778055081
46945488612181718608147748793873600011554317953745573813498318864313953173659802
0836954363067039061215443142526430003864209599033054041271268222599754407254559
15007659175313831213894142994756678740254780202593219881682115266522733429462603
71778914634223245917043877029348382339796683424043496851096106465949387979606919
88691850907852097495545835886916230644151177334415041670201654698940525608915869
62574326027104187634789541245386595007032931831342611103921035148090083733683519
888599951582291502749662096899082157946773737480571123429838226706043059218472763
40391239345862002787889920798450663570910380534944574979800741785471656436214007
92347253574310216258703519286809363949140742467741440744762906704825702311580873
75916581583080418504488746876581747727578028150951047605478480937680090154899709
47890687989733950653
得到数值: 800

Ln: 8 Col: 0
Ln: 1 Col: 20
```

#### (四) 拓展实验

选择使用移位密码进行加解密，并设置密钥 `key = 123456`

- 首先，设置必要的参数，包括服务器端保存的数值、使用移位加密后得到的密文数值、客户端的公私钥以及要读取的信息的位置。
- 其次，客户端生成密文选择向量，并将该向量交由服务器端进行运算。
- 最后，客户端对服务器端返回的数值进行解密，从而获取所需的信息。

具体代码实现如下：

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

def shift_encrypt_decrypt(data, key, encrypt = True):
    data_bytes = data.to_bytes((data.bit_length() + 7) // 8, byteorder = 'big')
    key_bytes = key.to_bytes((key.bit_length() + 7) // 8, byteorder = 'big')
    data_bytearray = bytearray(data_bytes)
    key_bytearray = bytearray(key_bytes)

    if encrypt:
        for i in range(len(data_bytearray)):
            data_bytearray[i] = (data_bytearray[i] + key_bytearray[i % len(key_bytearray)]) % 256
    else:
        for i in range(len(data_bytearray)):
            data_bytearray[i] = (data_bytearray[i] - key_bytearray[i % len(key_bytearray)]) % 256
```

```
len(key_bytearray)]) % 256

    return int.from_bytes(data_bytearray, byteorder = 'big')

# key 为加解密密钥
key = 123456

##### 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]
# 服务器要保存的加密后数值
server_list = []
for i in message_list:
    encrypted_data = shift_encrypt_decrypt(i, key, encrypt = True)
    server_list.append(encrypted_data)
    # print(encrypted_data)
# 加密后的数组
print("密钥 k 加密后的数组:", server_list)
length = len(server_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]))

##### 服务器端进行运算
c = 0
for i in range(length):
    c = c + server_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("得到用密钥 k 加密后的数值:", m)
decrypted_data = shift_encrypt_decrypt(m, key, encrypt = False)
print("得到原始数值: ", decrypted_data)
```



Python 3.11.4 (tags/v3.11.4:d2340ef, Jun 7 2023, 05:45:37) [MSC v.1934 64 bit (AMD64)] on win32

Type "help", "copyright", "credits" or "license()" for more information.

>>>===== RESTART: F:\study\datasecurity\lab2\test2.py =====

密钥 k 加密后的数组: [101, 201, 526, 626, 726, 826, 926, 1026, 1126, 1226]

要读取的数值位置为: 3

产生密钥 X: 729839657228583796733009732416868067643418887205518377538180158974502

05872347621636319710612581336552786684307026983355190420494235089529905145376285

2011704977061250901022851177926734933105327992490772451955372513247649767587451

457771716025773431279286765212451853312081215208876380931821830578276498778972

94152564381515197685900216102903775883794301455544693794489283132491657030911105

97404897147031087632573137794670126413993188010505379023842333911044666689963816

2222174102758960553103943022557803734479905778172290761136938481182543506630987

65878126902148541105152330795029537537563059104568944229937640545390210578231850

62316727294543133692171418802579949765379315683776463970209950956883633392166404

84665002715135716584966118795550189533748546590239419002323304301817591510261544

67135608420556598373545518584968006340077409114550873590353897840191330954976785

60702739930160994436605039023414575661641283634459124540489085150344245154195282

44162421333458587054680321406052663813076324802150409246813057697179137974668098

56698738253708720069893952748058609385556686410496764688791767409828523268251156

18021709503571253756084612040904745653287487801327101541966531850331728768743728

77538429705734286695281076062276269233947194043764663287379500715268901561635927

26573345218816248787104792097880073796585781139111338317290289357530458438373597

13436963135515316290028917425178085560580065929638145712829446836287153499650382

31724000188032165086121251700009452358403045207193211633285154199170099559683738

9938130272399000946269616194300341048183774141994369986935738228425345996388280

4277179043658401643501320962458772222601520716016180539332068127147791403317414

5677050632391775910814312246237575161183505090159341693455931335520183552813314

53782395342943182515891267100822778522942304654497316316200250474690638637748648

80043781986659329946

得到用密钥 k 加密后的数值: 626

得到原始数值: 400

Ln: 10 Col: 0

test2.py - F:\study\datasecurity\lab2\test2.py (3.11.4)

File Edit Format Run Options Window Help

from phe import paillier # 开源库

import random # 选择随机数

def shift\_encrypt\_decrypt(data, key, encrypt = True):

data\_bytes = data.to\_bytes((data.bit\_length() + 7) // 8, byteorder = 'big')

key\_bytes = key.to\_bytes((key.bit\_length() + 7) // 8, byteorder = 'big')

data\_bytearray = bytearray(data\_bytes)

key\_bytearray = bytearray(key\_bytes)

if encrypt:

for i in range(len(data\_bytearray)):

data\_bytearray[i] = (data\_bytearray[i] + key\_bytearray[i % len(key\_bytearray)]) % 256

else:

for i in range(len(data\_bytearray)):

data\_bytearray[i] = (data\_bytearray[i] - key\_bytearray[i % len(key\_bytearray)]) % 256

return int.from\_bytes(data\_bytearray, byteorder = 'big')

# key 为加解密密钥

key = 123456

##### 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]

# 服务器需要保存的加密后数值

server\_list = []

for i in message\_list:

encrypted\_data = shift\_encrypt\_decrypt(i, key, encrypt = True)

server\_list.append(encrypted\_data)

# print(encrypted\_data)

# 加密后的数组

print("密钥 k 加密后的数组:", server\_list)

length = len(server\_list)

# 客户端生成公钥

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("要读取的数值位置为:", 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]))

##### 服务器端进行运算

c = 0

for i in range(length):

c = c + server\_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("得到用密钥 k 加密后的数值:", m)

decrypted\_data = shift\_encrypt\_decrypt(m, key, encrypt = False)

print("得到原始数值:", decrypted\_data)

Ln: 1 Col: 0

要读取的数值位置为3，加密后的数值为626，原数值为400，正确。

五、总结与感悟

在本次实验中，我首先学习了半同态加密中的具体函数和算法，例如卡迈尔克函数和判定复合剩余假设等，并将这些知识成功复现到实验中。此外，还了解了半同态加密在现实生活中的应用，如联邦学习、隐私集合求和和数据库查询统计等。最后，通过实验将所学的理论知识应用于实践，进一步提高了对半同态加密及隐私信息获取的理解。