

数据安全 -- 半同态加密应用实践

学号：2013921

姓名：周延霖

专业：信息安全

一、实验名称

半同态加密应用实践

二、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

三、实验过程

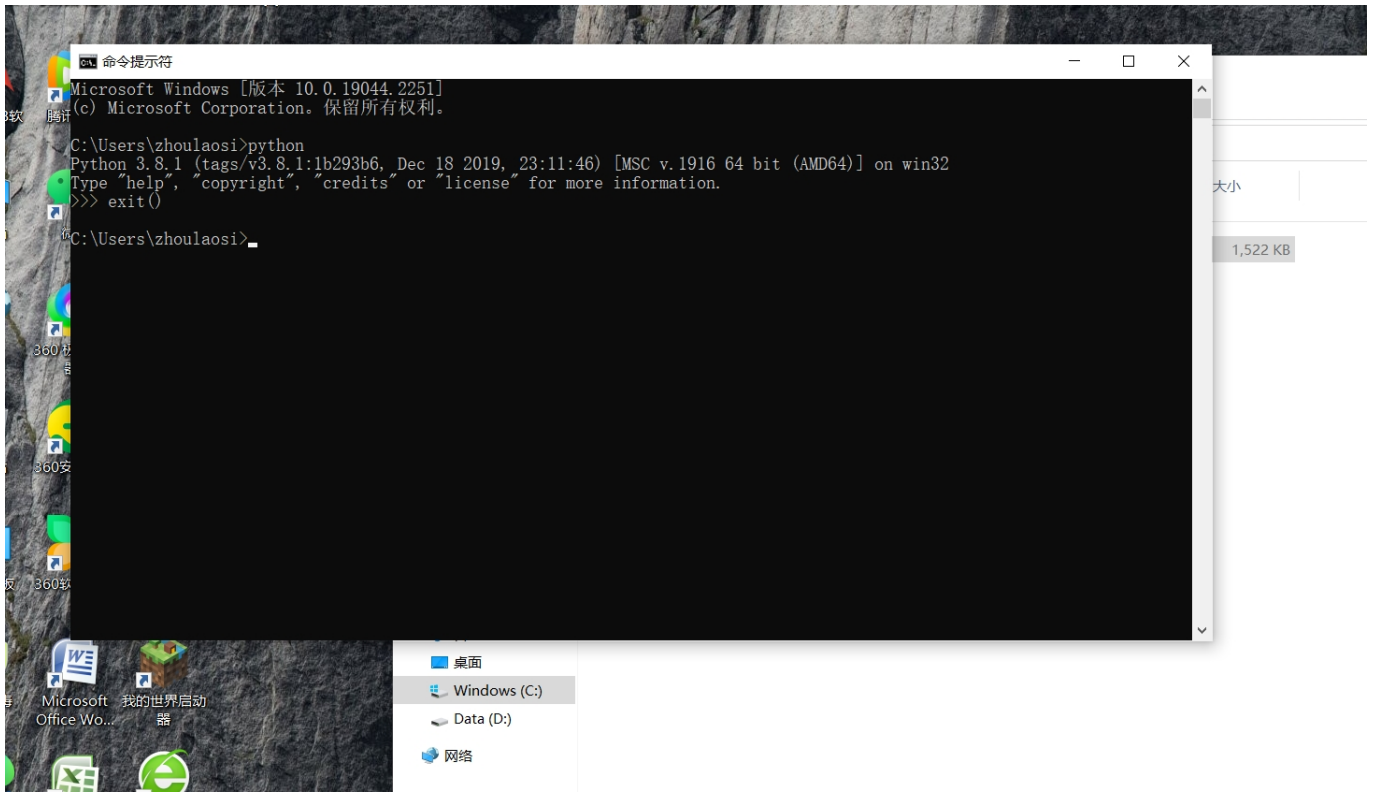
1、安装 python 环境

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：



代表已经安装成功，并且进入 python 运行环境。

输入 python 程序：`from phe import paillier`

该命令将导入 phe 库的 paillier 功能，第一次执行会提示“ModuleNotFoundError: No module named 'phe'”。因为，默认安装 python 后，并没有安装 phe 这个库。

输入 python 命令：`exit()`

该命令可以退出当前 python 环境，切回控制台模式

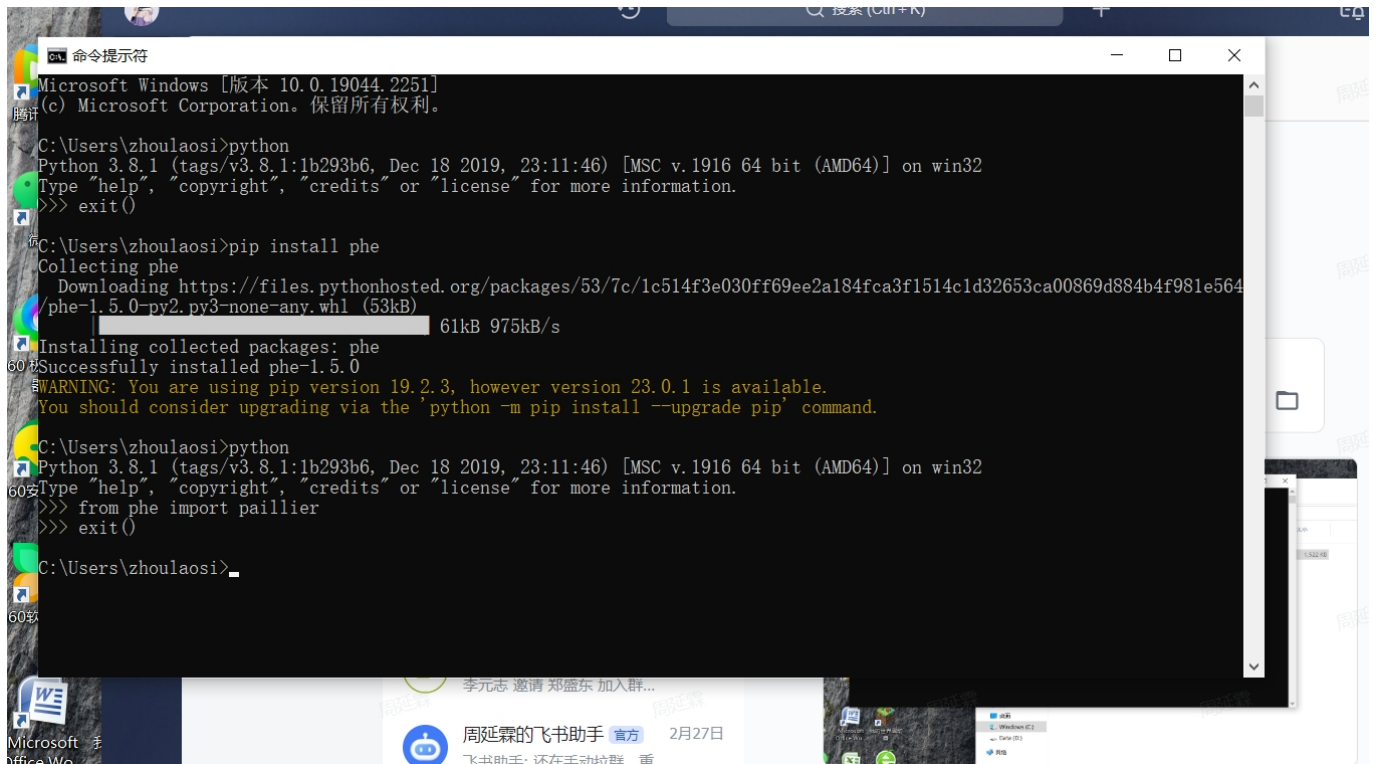
2、安装 phe 库

输入命令：`pip install phe` 完成 phe 库的安装

Pip 是 python 的一个安装库的工具，可执行文件在 python 安装目录下可以找到

3、验证环境正确性

再次进入 python 环境，输入 python 代码：`from phe import paillier`，如下图所示：



```
Microsoft Windows [版本 10.0.19044.2251]
(c) Microsoft Corporation。保留所有权利。

C:\Users\zhoulaosi>python
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> exit()

C:\Users\zhoulaosi>pip install phe
Collecting phe
  Downloading https://files.pythonhosted.org/packages/53/7c/1c514f3e030ff69ee2a184fca3f1514c1d32653ca00869d884b4f981e564/phe-1.5.0-py2.py3-none-any.whl (53kB)
    61kB 975kB/s
Installing collected packages: phe
Successfully installed phe-1.5.0
WARNING: You are using pip version 19.2.3, however version 23.0.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

C:\Users\zhoulaosi>python
Python 3.8.1 (tags/v3.8.1:1b293b6, Dec 18 2019, 23:11:46) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> from phe import paillier
>>> exit()

C:\Users\zhoulaosi>
```

发现不出现错误信息，说明环境安装成功

4、编写 python 程序并运行

可以有三种方式调试和编写 python 程序：

1. 在控制台运行 python 命令，逐行编写 python 程序并运行
2. 用文本编辑器编写完整的程序并保存为 x.py，通过控制台命令 `python x.py` 的方式完成整个程序的调用
3. 通过 IDLE 这个 python 的集成开发环境完成开发和调试运行。通过开始菜单，找到 IDLE 并打开，选择 File -> new file 可以新建一个文件，编辑程序并保存后，选择 Run -> Run Module 运行，会看到运行的结果

5、实验2.1

实验内容：

- 基于 Python 的 phe 库完成加法和标量乘法的验证

这里给出一个集成的演示代码如下：

```
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 s: ",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 s: ",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
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))
```

如上述代码所示:

1. python 程序对运算符进行了承载, 已经支持直接密文上的运算
2. 只支持明文的加法, 不支持明文的乘法, 最后一句如果将注释符去掉, 将报错

上述代码运行结果如下:

```
>>> from phe import paillier
>>> exit()
(wxnz) zhouyanlin@P:~$ python test1.py
默认私钥大小: 3072
加密耗时 s: 0.8368110656738281
加密数据 (3, 1415926) : 7712234118035430714054777414757916207615550278857378448133895896351055018262468233084210781613252068108487409613882779551539235013223727472172224466084938334663722666381226077904962
52980163558722780911434133001051408090387192905560587132010690338274979182383243445596428092554642866927282009919989622410850915985536710808798078905290613660354808850834534427526014130513224
367103803951372846424098262006389562113557188108029615492293680803307455378397856276076758495087312115401767184903408345231249216469790013968989795210184663253766221774964040267690260193092352885942561076
72082264391044291413057639565183155770049412206391635547946105882348422182785067841134010034186294141158390998988120596137163977605129825352247451828566936483278880869235341839444413588890717622761407143
87862268992936782135461229924668601735058848182002307928061309229984003449822537612227616698246753688058748689063344419243592670103849394157751899974227369803583298937827769235317757170748240270249798961
112660454884989832061317785166158728616317205341944222801870788046279231420706438887086980726483941076847888599520581711009964949160311035133738918540515665834682596538014716915051440883257751303230341227
12817339283682458245450138864628227478204317895051946239231204203901869488896477805250633907006747199367217024984631457247801334356552921827476781136436341312191857652780659950417140386424847458825148
2797590323845196332083334919232670028432095587482956536818777826210189837238559822872153732462338597304715490169384940116009517908680274164423573074636236968852719005923537062422627972568764242
8044211033178844180386852634382007789892892758013596027182241746831117998169106853761854945970029199653341353934680468022670235683167649403837491100581956340011444678778123822046992712892016290031172462
704630959533130479862813562640108874
解密耗时 s: 0.24268102645874023
原始数据 (3, 1415926) : 3, 1415926
a=5 密文 77122341180354307140547774147579162076155502788573784481338958963510550182624682330842107816132520681084874096138827795515392350132237274721722244660849383346637226663812260779049625298016355872
7284592906227009114341330010514080903871929055605871320106903382749791823832434455964280925546428669272820099199896224108509169855367108882980789852906126603548088508345344275260141305132243671038039513
728464240982920063895621135571881080296154922936808033074553783978562760767504950873121154017671849034083452312492164697900139689897952101846632537662217749640402676902601930923528859425610767208226439101
442914130576395651831557700494122063916355479461058823484221827850678411340100341862941411583909989881205961371639776051298253522474518285669364832788808692353418394444135888907176227614071438786226899293
67821354612299246686017350588481820023079280613092299840034498225376122276166982467536880587486890633444192435926701038493941577518999742273698035832989378277692353177571707482402702497989611126604548849
098320613177851661587286163172053419442228018707880462792314207064388870869807264839410768478885995205817110099649491603110351337389185405156658346825965380147169150514408832577513032303412271281733928368
2458245450138864628227478204317895051946239231204203901869488896477805250633907006747199367217024984631457247801334356552921827476781136436341312191857652780659950417140386424847458825148275903238451963320833349192326769028432095587482956536818777826210189837238559822872153732462338597304715490169384940116009517908680274164427375730746362369688527190059235370624226279725687642428044211033178
8441803868526343820077898928927580135960271822417468311179981691068537618549459700291996533413539346804680226702356831676494038374911005819563400114446787781238220469927128920162900311724627046309595331
30479862813562640108874
a+= 8, 1415926
a-3 0, 1415926000000007
b*=600
c/-10, 0= 4.6e-13
True
(wxnz) zhouyanlin@P:~$ python test1.py
```

6、实验 2.2

实验内容：

- 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。

首先，我们要基于 Paillier 协议进行设计

对 Paillier 的标量乘的性质进行扩展，我们知道：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

基于这个特性，我们可以如下巧妙的设计：

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

进而，开发具体代码如下：

```
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)
```

上述代码运行结果如下：

```
True
(wxn2) zhuyanlin@P_XOXZHOU-MB0 1 % python test2.py
要读起的数值位置为: 9
产生密文:  876746740982941800720459054154036621054275895847289267592460177139988761458656696586102429014647071327202279628986913083837760447281496113873674602029275011088486720825133949030977714961966681
520365213795614946567959169342820920120849064744147696627091327547840946202059628810522392010026061637508614184644561968914924630804873379398375392520037308355020473517386139234890148645834192913512004294
8403128331927618768422584676881751741901302789927729810260241212861668231798828531348450594141296706873041746703566075811346311247664710827213183279316357738631926128852540265439210117510856971937584
6278208798162000551077979459955108598576392470051322079879500354783399760303547253026043830320477394003111452110908116693134490420795017059620345735049136384150865772423558412673389811759119458092300
725512812365391379025767814108787270874048195808492024372639976718275153890950010192349429180710133294788644452851406889330144804834477698971792638063531871931173225800106570961312170936987550879004849028
24161273814553351055048132274219595589435233253218868241249579306907579830062588793738156584071459384612807924221095338222925835695925388692930102129020010639181192658470142073314325095440928194116337881
873881125610966869567403756164888365769113252697314952752850247046119980710966418190492495192315240157696673939311405846512727877919453326348393869347061004107694397739710658743303632439406431872988513065
371747224434227024582029452117788123133876320998356715853092753677945617609655137870745925246767067583689966713273041554302670973518439974434554007982360830733218528790729135631342134695255411497960482
664141905865149132446669576144035763663258979378941000690975992796357735590132472781334063888592949808658132699744758325867686343767649981388135172065731360192595981503502670361863805986433629850652379526
513014051701206700779891
得到数值: 1000
(wxn2) zhuyanlin@P_XOXZHOU-MB0 1 %
```

四、心得体会

在本次实验中，首先学习到了半同态加密中的具体的一些函数和算法，例如卡迈克尔函数和判定复合剩余假设等，并将其复现到实验中

还了解到了半同态加密在现实生活中的具体应用例如联邦学习、隐私集合求和、数据库查询统计等

最后通过两个实验对所学到的理论知识进行相应的应用，对python中的phe库的应用也更加的熟练，期待自己未来更好的发展，心想事成、万事胜意、未来可期