实验二 基于Paillier算法实现隐私信息获取

2112852 密码科学与技术　胡佳佳

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信息，同时客户端能完成获取信息的解密

理论知识

同态加密（HE）

HE是一种特殊的加密方法，它允许直接对加密数据执行计算，如加法和乘法，而计算过程不会泄露原文的任何信息。计算的结果仍然是加密的，拥有密钥的用户对处理过的密文数据进行解密后，得到的正好是处理后原文的结果。

根据支持的计算类型和支持程度，同态加密可以分为以下三种类型：

- **半同态加密**（Partially Homomorphic Encryption, **PHE**）：只支持加法或乘法中的一种运算。其中，只支持加法运算的又叫加法同态加密（Additive Homomorphic Encryption, AHE）；
- **部分同态加密**（Somewhat Homomorphic Encryption, **SWHE**）：可同时支持加法和乘法运算，但支持的计算次数有限；
- **全同态加密**（Fully Homomorphic Encryption, **FHE**）：支持任意次的加法和乘法运算。

PHE--Paillier【1】

Paillier是一个支持加法同态的公钥密码系统，由Paillier在1999年的欧密会（EUROCRYPT）上首次提出。此后，在PKC'01中提出了Paillier方案的简化版本26，是当前Paillier方案的最优方案。在众多PHE方案中，Paillier方案由于效率较高、安全性证明完备的特点，在各大顶会和实际应用中被广泛使用，是隐私计算场景中最常用的PHE实例化方案之一。

> `phe.paillier.``generate_paillier_keypair`(*private_keyring=None*, *n_length=2048*)[source]
>
> Return a new `PaillierPublicKey` and `PaillierPrivateKey`.
>
> Add the private key to *private_keyring* if given.

Parameters:	**private_keyring (*PaillierPrivateKeyring*) – a `PaillierPrivateKeyring` on which to store the private key.n_length – key size in bits.**
Returns:	The generated `PaillierPublicKey` and `PaillierPrivateKey`
Return type:	tuple

实验步骤

初始化

服务器端：产生一定长度的数据列表

```
#server's data
msg_list=[1,2,3,4,5,6,7,8,9,10]
msglist_len=len(msg_list)
```

客户端：生成公私钥对，并选择随机选择位置

```
#client op
public_key, private_key = paillier.generate_paillier_keypair()
pos=random.randint(0,msglist_len-1)
print(pos)
```

计算

客户端：加密选择向量

```
##select data
select_list=[]
encryt_list=[]
for i in range(msglist_len):
    select_list.append(i==pos)
    encryt_list.append(public_key.encrypt(select_list[i]))
    print(encryt_list[i].ciphertext())
```

服务器端：计算选择向量关于数据列表的数据线性组合（选择数据的密文）

```
#server calculates
c=0
for i in range(msglist_len):
    c=c+encryt_list[i]*msg_list[i]
```

客户端：解密得到选择的数据

```
m=private_key.decrypt(c)
print(m)
```

这样客户端就得到了服务器端无法具体知道的数据。

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

本次实验中，使用对称算法AES进行加密储存在服务器的消息

Fernet--AES

> Fernet is built on top of a number of standard cryptographic primitives. Specifically it uses:
>
> - `AES` in `CBC` mode with a 128-bit key for encryption; using `PKCS7` padding.
> - `HMAC` using `SHA256` for authentication.
> - Initialization vectors are generated using `os.urandom()`.

Fernet (symmetric encryption) — Cryptography 43.0.0.dev1 documentation

改进后步骤

利用python里的`cryptography.fernet`导入`Fernet`

```
from phe import paillier
from cryptography.fernet import Fernet
import random
import math
```

初始化

客户端：生成对称密钥并加密

```
#client init
key=Fernet.generate_key()
f=Fernet(key)

##client plain data
msg_list=[1,2,3,4,5,6,7,8,9,10]
msglist_len=len(msg_list)
enc_msglist=list(f.encrypt(x.to_bytes()) for x in msg_list)
a=1
```

计算

服务器端：利用存储密文进行线性组合

```
#server calculates
c=0
for i in range(msglist_len):

 c=c+encryt_list[i]*int.from_bytes(enc_msglist[i],byteorder='little',signed=False
)
```

客户端：先用paillier的私钥求出选择密文，再通过对称密钥进行解密

```
tmp=private_key.decrypt(c)
byte_length = math.ceil(tmp.bit_length() / 8)
m=f.decrypt(tmp.to_bytes(byte_length, byteorder='little'))
```

至此，服务器端对客户端存储和选择的数据未知，客户端可实现隐私获取。

总结

利用半同态加密可以实现客户端对服务器端的隐私获取。

参考文献

【1】API Documentation — python-paillier 1.4.0 documentation

【2】Fernet (symmetric encryption) — Cryptography 43.0.0.dev1 documentation

【3】Paillier半同态加密：原理、高效实现方法和应用 - 知乎 (zhihu.com)