



南開大學
Nankai University

计算机学院和网络空间安全学院
数据安全实验报告

半同态加密应用实践

姓名：魏伯繁

学号：2011395

专业：信息安全

2023 年 3 月 13 日

目录

1 实验要求	2
2 实验过程	2
2.1 阅读并复现已经给出的 test.py 代码	2
2.2 学习使用 python 提供的 AES 库	4
3 完成拓展实验	5
4 心得体会	7

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 阅读并复现已经给出的 test.py 代码

首先, 根据上课学习到的理论知识, 我们了解了同态加密 (homomorphic encryption) 是一种加密算法, 它可以通过对密文进行运算得到加密结果, 解密后与明文运算的结果一致, 这样我们就可以在不暴露原始数据的基础上将数据加密后给予合作伙伴, 使其能够完成对加密后数据的运算, 并且得到的结果与未加密前的结果一致。

根据支持的运算种类的不同以及运算次数的区别, 半同态加密 (partial homomorphic encryption) 仅支持单一类型的密文域同态运算 (加或乘同态)

而本次实验中所复现的 paillier 加密算法是 Paillier 等人 1999 年提出的一种基于判定 n^* 阶剩余类难题的典型密码学加密算法, 具有加法同态性, 是半同态加密方案。

总结 paillier 加密算法的特性其实有两点:

- (1) 密文乘等于明文加 (加法同态性)
- (2) a 个密文相加等于 a 个明文相加 (标量乘同态性)

paillier 的算法实现过程可以通过调用函数来完成计算, 其具体的过程如下所示:

(1) 密钥生成

- ✧ 随机选择两个质数 p 和 q , 尽可能地保证 p 和 q 的长度接近或相等 (安全性高);
- ✧ 计算 $n = pq$ 和 $\lambda = \text{lcm}(p-1, q-1)$, 其中 lcm 表示最小公倍数;
- ✧ 随机选择 $g \in \mathbb{Z}_{n^*}^*$, 考虑计算性能优化, 通常会选择 $g = n + 1$;
- ✧ 计算 $\mu = \left(L(g^\lambda \bmod n^2) \right)^{-1} \bmod n$, 其中 $L(x) = \frac{x-1}{n}$;
- ✧ 公钥为 (n, g) ;
- ✧ 私钥为 (λ, μ) 。

图 2.1: 密钥生成过程

(2) 加密算法

对于任意明文消息 $m \in \mathbb{Z}_n$, 任意选择一个随机数 $r \in \mathbb{Z}_n^*$, 计算得到密文 c :

$$c = E(m) = g^m r^n \bmod n^2$$

注意: 密文 c 要比明文 m 长度要长。

图 2.2: 加密算法

(3) 解密算法

对于密文 $c \in Z_n^*$, 计算得到明文 m :

$$m = D(c) = L(c^\lambda \bmod n^2) * \mu \bmod n$$

图 2.3: 解密算法

在介绍完 paillier 算法的基础知识后展示本次实验希望实现的任务:

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。

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。

服务器端: 产生数据列表 $\text{data_list} = m_1, m_2, \dots, m_n$

客户端

- 设置要选择的数据位置为 pos
- 生成选择向量 $\text{select_list} = 0, \dots, 1, \dots, 0$, 其中, 仅有 pos 的位置为 1
- 生成密文向量 $\text{enc_list} = E(0), \dots, E(1), \dots, E(0)$
- 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 $c = m_1 * \text{enc_list}[1] + \dots + m_n * \text{enc_list}[n]$
- 返回密文 c 给客户端
- 生成密文向量 $\text{enc_list} = E(0), \dots, E(1), \dots, E(0)$
- 发送密文向量 enc_list 给服务器

客户端: 解密密文 c 得到想要的结果

老师给出的示例代码就完成了下面的任务, 我将具体代码分块并解释其中的含义

下面这段代码完成的任务是: 展示服务端存储的数据, 并且在客户端挑选一个将要访问的数组下标

```

1
2 # 下面这个列表保存了服务端存储的数据
3 message_list = [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]
4 length = len(message_list)
5 # 客户端生成公私钥
6 public_key, private_key = paillier.generate_paillier_keypair()
7 # 客户端随机选择一个要读的位置, 接下来客户端将要读取对应位置的数据
8 pos = random.randint(0, length - 1)
9 print(" 要读起的数值位置为: ", pos)
10

```

下面这段代码的作用是：生成要去访问服务端数组的对应数据向量

```

1
2 ##### 客户端生成密文选择向量
3 # 该段代码的意图很明显，如果 i 等于要访问的下标，那么就在 select_list 中添加 1，否则就添加 0
4 # 然后 enc_list 来对 select_list 中的数字进行加密
5 select_list = []
6 enc_list = []
7 for i in range(length):
8     select_list.append(i == pos)
9     enc_list.append(public_key.encrypt(select_list[i]))
10

```

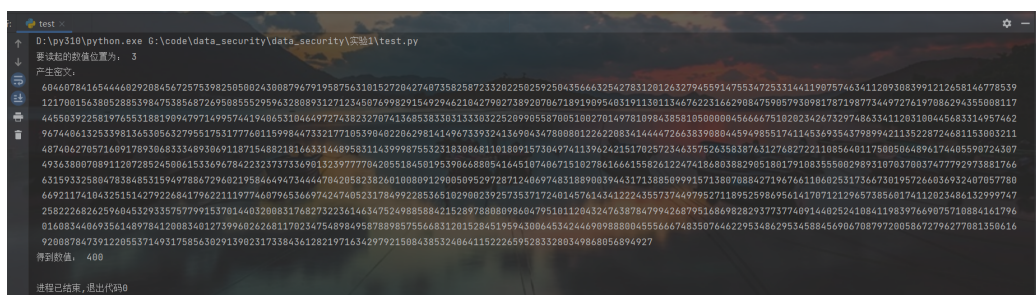
下面这段代码的意思是首先在服务端进行计算，也就是将服务端中保存的数据向量与客户端生成的选择向量对应相乘即可，然后客户端对返回的密文进行解密

```

1
2 ##### 服务器端进行运算
3 c = 0
4 for i in range(length):
5     c = c + message_list[i] * enc_list[i]
6 print(" 产生密文: ", c.ciphertext())
7 ##### 客户端进行解密
8 m = private_key.decrypt(c)
9 print(" 得到数值: ", m)
10

```

实验的结果截图为：可以看到上述代码成功完成了对应操作



```

D:\py310\python.exe G:\code\data_security\data_security\实验1\test.py
要读取的数值位置为: 3
产生密文:
684687841654446829288467257539825058024388876791958756318152720427487358258723202258259250435666325427831281263279455914755347253314411987574634112893883991212658146778539
121788156388528853984753856872695855529563288893127123450769829154929462184279827389287867189190954831911381134676223166298847590579389817871987734497276197886294355888117
445583922581976553188198947771499574419486531046497274382327874136853833831338322528990587005108278149781898438581050888845666751828234267329748633411283188445683314957462
96744861325398136538563279551753177681159984473521771855984022862981414967339324136984347808801226228834144472663839888459498551741145369354379899421135228724681153883211
487486270571691789386334893891187194862181263314489583114399875532133886421188915738974113962421517075723443575263838933177827221108544011758058648617440559072487
493438807889112078252580613336947842232373733509132397778428551845319864488516451874867151027841464515826124941888838839851881721083558892893107378837477792973861744
43159332588478384853159497884779462195844494734447842858232610188891290859539728712486974831889883944317138858999157138878842719476411868253173447381957246436162487857788
6492117418432515127226248417962211197746487945364974267488231784992285365182980239257357172401467414341222435373744879527118952598495414187821212963738560174112833486112999747
25822262262596045329335757799153781440328883176827122316463475249885884215289788809846479510112843247628784799424879516869828293773774891440252341884198397649075710884161794
81688340693561489784128083481275964826248117023475489849587889857554683128152845195943806451424469898888845556647483507644229534862953458845690678879728058672796277881358616
9288878473912205537149317585638291398231733843612821971634297921588438532486411522265952833280349868056894727
得到数值: 409
进程已结束,退出代码0

```

图 2.4: 复现结果

2.2 学习使用 python 提供的 AES 库

在进阶要求中，我希望使用 python 提供的 AES 库作为对明文加密的算法，AES 高级加密标准在密码学中又称 Rijndael 加密法，是美国联邦政府采用的一种区块加密标准。这个标准用来替代原先的 DES，已经被多方分析且广为全世界所使用。经过五年的甄选流程，高级加密标准由美国国家标准与

技术研究院 (NIST) 于 2001 年 11 月 26 日发布于 FIPS PUB 197, 并在 2002 年 5 月 26 日成为有效的标准。2006 年, 高级加密标准已然成为对称密钥加密中最流行的算法之一。

接下来, 我们学习在 python 中使用 AES 库, 为进阶实验做准备。

```

1
2 # 导入 AES 库
3 from Cryptodome.Cipher import AES
4 # 构造一个 AES 类来实现加密解密, 因为 AES 是对称加密算法, 所以他的加密解密使用一样的密钥,
5 其中参数 key 就是密钥,
6 需要用户提供 byte 类型的 16Byte 的密钥, AES_MODE 是加密模式, 例如可以选择 ECB 或者 CBC 等等,
7 aes = AES.new(key, AES.MODE_CBC)
8 # 这一步就是加密, 其中 content 是要加密的内容, 同样也需要传入 16byte 的以 byte 为类型的数据
9 content=aes.decrypt(content)
10

```

3 完成拓展实验

拓展实验的实现同样也是基于老师给出的 test.py 的基础上完成的, 完整的代码如下:

具体的思路是首先将服务端保存的数转换为 byte 类型存储并且使用 AES 加密后也是 byte 类型的, 当需要 paillier 时就将 byte 转换为 int 类型的进行操作。返回的也是一个 int 类型的数字, 接下来到客户端时需要把这个 int 类型的变量换成 byte 类型的做 AES 解密, 然后就是服务端中对应下标的结果啦。

```

1
2 from phe import paillier # 开源库
3 from Cryptodome.Cipher import AES # 使用 python 的 AES 库
4 import random # 选择随机数
5
6
7 # str 不是 16 的倍数那就补足为 16 的倍数
8 # AES 只能用 16 位的, 但这个没用了, 一开始我想把 int 转成 str 然后在弄成 byte, 但是这样会出现
9 乱码
10 # 然后上网一查发现 int 和 byte 可以直接相互转换, 小丑竟是我自己
11 def add_to_16(value):
12     while len(value) % 16 != 0:
13         value += '\0'
14     return str.encode(value) # 返回 bytes
15
16
17 # 对称密钥, 改成什么样子都行, 如果以字符串形式出现的话记得弄成 byte 类型, 16 位就好
18 password = b'1234567812345678'

```

```
19
20 # 服务器端保存的数值
21 message_list = [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000]
22 length = len(message_list)
23
24 # 服务器端对保存的内容进行加密
25 encrypt_message_list = []
26 aes = AES.new(password, AES.MODE_ECB) # 创建一个 aes 对象, 采用 ECB 加密模式
27
28 # 对每一个 message 都加密然后保存的一个 list 中
29 for i in range(length):
30     # 加密时需提供 byte, 于是将 int 转成 byte, 要提供的参数就是转换成的 byte 数,
31     # 大端还是小端序以及是否有符号
32     en_text = aes.encrypt(message_list[i].to_bytes(length=16, byteorder='big', signed=False))
33     encrypt_message_list.append(en_text)
34
35 # 客户端生成公私钥
36 public_key, private_key = paillier.generate_paillier_keypair()
37 # 客户端随机选择一个要读的位置
38 pos = random.randint(0, length - 1)
39 print(" 要读起的数值位置为: ", pos)
40
41 ##### 客户端生成密文选择向量
42 select_list = []
43 enc_list = []
44 for i in range(length):
45     select_list.append(i == pos)
46     enc_list.append(public_key.encrypt(select_list[i]))
47
48 # for element in select_list:
49 #     print(element)
50 # for element in enc_list:
51 #     print(private_key.decrypt(element))
52
53 ##### 服务器端进行运算
54 c = 0
55 for i in range(length):
56     # 根据 paillier 算法的要求把 byte 转成 int 进行计算, 因为 paillier 只能计算 int 型的值,
57     # 所以要把 byte 类型的转换为 int 类型
58     trans = int().from_bytes(encrypt_message_list[i], byteorder='big', signed=False)
59     c = c + trans * enc_list[i]
60 print(" 产生密文: ", c.ciphertext())
```

```

61
62 ##### 客户端进行解密
63 m = private_key.decrypt(c)
64 # 把拿回来的 byte 通过 AES 解密成需要的 int
65 print('未经过 AES 解密的密文为: ', m)
66 m = aes.decrypt(m.to_bytes(length=16, byteorder='big', signed=True))
67 m = int().from_bytes(m, byteorder='big', signed=True)
68 print(" 经过 AES 解密后得到数值: ", m)
69

```

实现效果如下：可以发现不管是 paillier 还是 AES 都顺利的成功解密

```

D:\py310\python.exe G:\code\data_security\data_security\实验1\extend_test.py
要加密的数据位数为: 8
产生密文:
276732258454286167629366348973104028086898795234766329274997173773655388549837992732518688156175739693257348587496585076168243648648441739828235528876068882544732208875763944
65284766354408312819294399801452162187772499825225883786536841502869038487526822683938773662991519271856487808793757587764653129216563252894821983428632367386894363689852293
5943489989283106691817372786038547960277519421207996497426519942913286232373769790524488323441478672438791488813011409315407556294231359446924530144716221571434981376298594
52188519869614719996264163823104083057647254681195711481619851486864213343988911311475745287079305678568142485766602398202066671455452454724922669386592272368726879627778516
2231661927319379459807841394680860538057784853461025139498539332803317871292756521320383026756424450622715511640146845780304990537026071884853193478483586178013047718428772
5928271964573285923678044884499424227523295715043748705720172313848276547172951045171246851572687036373578171891056891834758480653985941846336021582766255133917856575297355
619296149329288358320206458325452497134428560553280934608204623553726562822683402154353453854398656295402332549030167945288728776979712344110195584386558358966051815402175436
956565814756177310937593416109092813788392789753489995136936349764360375392762627787614662393305061540894321139231688379770516496617731023485204632795536384732284149898983799
15677809751983691984273081846410706709536349450766967813187562890294434386683392928208431990516751424973815188266239811155604925914207862997473713666275798453441908889687947
8238192120627334760936302771098961128346691550706867351773353808851694739375788408442980631098914456269151632056727592448979995421291723767958782733753459605618544950288711
758379391636196145490054943739112793885647069565376108429259440669734298767042988155252783760812778888884054
未经过AES解密的密文为: 73497722468718361663926393947363069432
经过AES解密后得到数值: 900
进程已结束, 退出代码0

```

图 3.5: Caption

4 心得体会

通过本次实验,我学习了半同态加密算法的应用实践,实验中给出了一个非常贴近现实的使用 paillier 的场景,在本地将要运算的内容加密好后发送给服务器端,服务器端进行解密后返回,并且在拓展实验中,我也动手实现了一个进阶版的要求,通过调用 python 库函数 AES 库来实现两次加密操作,也就是在服务器上存储的数据不是直接的明文数字,而是经过 AES 算法加密的数字作为 paillier 的明文进行计算,这样做可以进一步保护用户数据安全,体现加密算法的特性。

经过自己动手实践,我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,更好的理解了加密算法的特性及其适合的应用场景。