# 《数据安全》实验报告

- 实验一：基于Paillier算法算法实现隐私信息获取
- 姓名：李欣
- 学号：2011165
- 专业：计算机科学与技术

# 一、实验要求

1. 基础实验：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。

2. 拓展实验：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。

# 二、实验过程

## (一)基础实验

- 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）；

### 1. paillier算法

- 密钥生成

1. 随机选择两个大素数$p, q$满足$gcd(pq, (p-1)(q-1)) = 1$，且满足$p, q$长度相等
2. 计算$n = pq$以及$\lambda = lcm(p-1, q-1)$，这里$lcm$表示最小公倍数，$|n|$为$n$的长度，这里$lcm$表示最小公倍数，$|n|$为n的比特长度
3. 随机选择整数$|g \leftarrow Z_{n^2}^*|$
4. 定义$L$函数：$L(x) = \frac{x-1}{n}$ 计算$\mu = (L(g^\lambda \ mod \ n^2))^(-1) \ mod \ n$

公钥：$(n, g)$，私钥：$(\lambda, \mu)$

- 加密

1. 输入明文消息$m$，满足$0 \leq m < n$
2. 选择随机数$r$满足$0 \leq r < n$，且$r \in Z_{n^2}^*$
3. 计算密文$c = g^m r^n \ mod \ n^2$

- 解密

1. 输入密文$c$，满足$c \in Z_{n2}^*$
2. 计算明文消息$m = L(c^\lambda \ mod \ n^2)\mu \ mod \ n$

- 同态加

1. 对于密文$c_1$ 和$c_2$ ，计算$c = c_1 * c_2 \ mod \ n^2$

- 同态标量乘

1. 对于密文$c_1$和标量$a$，计算$c = c_1^a \ mod \ n^2$

## 2. 隐私信息获取原理

**服务器端**：产生数据列表 $data\_list = m1, m2, \ldots, mn$

**客户端**:

- 设置要选择的数据位置为$pos$
- 生成选择向量 $select\_list = 0, \ldots, 1, .., 0$，其中，仅有$pos$的位置为 1 • 生成密文向量 $enc\_list = E(0), \ldots, E(1), .., E(0)$
- 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 $c = m1 * enc\_list[1] + \ldots + mn * enc\_list[n]$
- 返回密文 $c$ 给客户端

**客户端**：解密密文 $c$ 得到想要的结果

## 3. 编程实现

- 需要的包

In [1]:
```
from phe import paillier # 开源库
import random # 选择随机数
```

- 设置参数

In [2]:
```
# 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("要读取的数值位置为：",pos)
```

```
要读取的数值位置为： 6
```

- 客户端生成密文选择向量

In [3]:
```
select_list=[]
enc_list=[]
for i in range(length):
    # i==pos结果为true or false
    select_list.append( i == pos )
    # public_key.encrypt(select_list[i])结果为 0 or 1
    enc_list.append( public_key.encrypt(select_list[i]) )
```

- 服务器端进行运算

In [4]:
```
c=0
for i in range(length):
    c = c + message_list[i] * enc_list[i]
    # print("产生密文：",c.ciphertext())

print("产生密文：",c.ciphertext())
```

```
产生密文： 10868621643188470547976428506756315736403127541400631836846762589273
46309835195872965064192360860169164411083281081769304429197026029122061768268316900149574914408114778774272173330040500219133208586660660887290316993961325004810290864168089957965603491733344683024871670494764481556488731458703816092546506419813626472881037275168095053719486148102517769016532541578030165620703793643665153082153792972949541706873459047029426085694393205322164003415636921654933846335664757898731429915518093934330484451226917937230286699111874037692067881906671001219096397846340250099175629308921645891598427839259148290262754999368687447040444047864624685137949477801776028868991890849408320257327272442517705121304027176322863159689404915562647836183692325603526279864308469697621918526755247775397360512179540665851572812711832733912776043070561433587472263632292735904420528599947990861222712322006556426306461507440887501498424497352647159970468814461711575183740271754959403953769560516841237966280556700213330829194838446817716685997699977657937613208730728341024574090488831250411475271968412004298749384834053167194053462178453501489265087865578415347316737096272769915970384872987372700854344960189310074624404188103317705717691775432582524559273030598628214167253618314302339478948597115747542650027285121311358656867947989249451276735789092108085885227702508418640008774533138112568919976521123942177825827666182704896202613552270016209701080358706972409692629471300731859815491806820480281068324581062497700835363406624501506439836462793341873197079408238075853536484828229762977274251247306225509281295475532543573093137833051904586150343365862998429199427940386530753448784952657242120817491212644240400298412984516270001004902388839835664646476279798292893007322151388370748337083779927423110067112042337463776622501810777979287010136333771064035 1
```

- 客户端进行解密

In [5]:
```
m=private_key.decrypt(c)
print("得到数值：",m)
```

```
得到数值： 700
```

可以看到，最终得到的结果为期望的结果

## (二)拓展实验

在拓展实验中，服务端存储$m$个用对称密钥$k$加密的密文（密钥k存储在客户端），使用$AES$对称加密算法对$m$个原始的数据进行加密。$AES$算法的实现可以直接调取python相关的包。由于$AES$以$16bit$为单位进行加密和解密操作操作，需要客户端提供一个$16bit$的对称密钥，且在运算时，需将整型消息转换为比特类型。在转换过程中，需要注意比特类型和整型类型的反复转换。

- 需要的包

In [6]:
```
from Cryptodome.Cipher import AES
# from Cryptodome.Cipher import DES
```

- 设置参数

In [7]:
```
# 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("要读取的数值位置为：", pos)
```

```
要读取的数值位置为： 2
```

- 对保存的内容进行加密

In [8]:
```
key = b'1234567891011121'
aes = AES.new(key, AES.MODE_ECB)  # 创建一个aes对象
aes_encrypt_list = []  # 保存使用aes加密后的消息

for i in range(length):
    temp = aes.encrypt(message_list[i].to_bytes(length=16,
                                                byteorder='big',
                                                signed=True))
    aes_encrypt_list.append(temp)
```

- 客户端生成密文选择向量

In [9]:
```
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

In [10]:
```
c = 0
for i in range(length):
```

```
    trans = int().from_bytes(aes_encrypt_list[i],
                             byteorder='big',
                             signed=True) #
    c = c + trans * enc_list[i]
print("产生密文：", c.ciphertext())
```

```
产生密文： 504053444684300894534197359704783625314360577559010566164484070221421516496060538669617357363947700610341364898035224111385590935718754415688349581841325827378401647350746343849356399027066415121050841866342461694146981651999182908967430191841451241110124669812203685327349706911777285200113675179414117309307223683434341253419452206542273481897118481957214003876355403460637306468102393049520333156780375347640263204718724372241703682994350004087746915014175773261811048763432103770457711717429009820101170765271808170102594295034518285010175787239631874802993506291422590610026672329785652124495641009302268486013096492861289735474196237752232297424371345735351751383134933435393558202035513576319130078543703511574826493529204164070929922517246146205061715004353112591228248327926436857004252926140443365211486180679900284854192917453702602374775766003739045848893740728541719127105603914329937137530658761870366684349567063992595905316964204826517369951190095906335111471899880496946123500125355196835018388180665918300304458090012940611985109001942852743683521429868643531053472434048545892731749093358320058208139034547435730450738542129183319142028074961416567979857242635865727527370548943869532565207900770456977707666907516272409018864478743259000502203770854452230922618370238591932329360329226419986974638254067975395598655505620999710477368846701165356943100551233685147891150859045760339452525522249595188524659879983639572727127689934527563521860074969402716471327141852734813590343763890645767018283185747948419373160325044908731730193823229364874967229191361920778873387569165695526175760231398702600381440851040338695035407152012158470491095629784546208785905191631582599703626019942541369498240555663866740785841032980334557746078598899383044046973494427628026866622474560890168099636530632999426892711232548045516634917415358420 28
```

- 客户端进行解密

In [11]:
```
m = private_key.decrypt(c)
# 使用aes解密，需要先转换类型
m = m.to_bytes(length=16, byteorder='big', signed=True)
m = aes.decrypt(m)
# 解密后转换为整型
m = int().from_bytes(m, byteorder='big', signed=True)
print("得到数值：", m)
```

```
得到数值： 300
```

可以看到，最终得到的结果为期望的结果

# 三、心得体会

1. 由于是计科专业的同学，只在软件安全课程中接触过密码相关的知识，所以无论是课堂知识还是实验内容，理解起来都有些困难，在知识的理解上下了功夫，最终理解了隐私信息获取的原理，paillier算法的基本原理，了解了AES和DES解密算法。
2. 由于时间较短，所以对对称密钥加密算法AES和DES的学习只停留在表面（理论+掉包），会在以后努力从底层实现该算法。