简介

Paillier同态加密是由Pascal Paillier于1999年提出并命名的密码学理论。它是一种基于公私钥密码学的概率非对称算法。

这套理论是一个加法同态加密算法，意味着，只要给定公钥和需要加密的信息m_1和m_2，就可以计算加密后的m_1和m_2之和，再可以用私钥解密结果，这整个过程精度没有损失。

> Paillier同态加密可以直接在密文上计算，然后解密就行了。

算法理论

首先回顾一下二项式定理。$n \in \mathbb{N}^*$

$$(a+b)^n = \sum_{r=0}^{n} C_n^r a^{n-r} b^r = C_n^0 a^n + C_n^1 a^{n-1} b + \cdots + C_n^r a^{n-r} b^r + \cdots + C_n^n b^n \quad (1)$$

当$a=1, b=n, n=x$时可以化成下面的形式：

$$(1+n)^x = 1 + nx + \frac{x(x-1)n^2}{2!} + \ldots \quad (2)$$

可以化为：

$$(1+n)^x \equiv 1 + nx \pmod{n^2} \quad (3)$$

> mod是求余符号。这个表达式的意思是，$(1+n)^x$ 对n^2求余后的结果，与$1+nx$对n^2求余的结果相等

令$y = (1+n)^x \bmod n^2$，可化简为$x \equiv \frac{y-1}{n} \pmod{n^2}$，再令$L(u) = \frac{u-1}{n}$

则

$$L\left((1+n)^x \bmod n^2\right) \equiv L(y) \equiv \frac{y-1}{n} \pmod{n} \equiv x \pmod{n} \quad (4)$$

即$L\left((1+n)^x \bmod n^2\right) \equiv x \pmod{n}$

Paillier同态加密

1. 随机选择两个比较大的素数p和q，并且保证$gcd(pq, (p-1)(q-1)) = 1$。gcd是求最大公因数，满足$gcd(pq, (p-1)(q-1)) = 1$，说明p和q的位数是相同的。
2. 令$n = pq$，$\lambda = lcm(p-1, q-1)$。lcm是求最小公倍数。
3. 随机选择一个整数g，$g \in \mathbb{Z}_{n^2}^*$
4. 令$\mu = \left(L\left(g^\lambda \bmod n^2\right)\right)^{-1} \bmod n$，其中确保存在 [modular multiplicative inverse]。

> 在这里modular multiplicative inverse不用特别注意，肯定满足的这个条件的，往下看。

如果p和q位数相等，令$g = n+1, \lambda = \varphi(n)$, and $\mu = \varphi(n)^{-1} \bmod n, \varphi(n) = (p-1)(q-1)$可以满足上面几条规则。

公钥是(n, g)，公钥用于加密

私钥是(λ, μ)，私钥用于解密

加密

1. 假设m为明文，是需要加密的信息，并且$0 \leq m < n$
2. 随机选择一个r，并且$0 < r < n, r \in \mathbb{Z}_n^*$ 和$gcd(r, n) = 1$
3. 密文$c = g^m \cdot r^n \bmod n^2$

解密

$m = L\left(c^\lambda \bmod n^2\right) \cdot \mu \bmod n$

证明如下

将$c = g^m \cdot r^n \bmod n^2$代入$m = L\left(c^\lambda \bmod n^2\right) \cdot \mu \bmod n$可得

$$m = L\left(g^{\lambda m} \cdot r^{\lambda n} \bmod n^2\right) \cdot \mu \bmod n = \lambda m \cdot \mu \bmod n = \lambda m \cdot \frac{1}{\lambda} \bmod n \tag{5}$$

其中，$r^{\lambda n} \bmod n^2 = 1$，其实$n$并不是一般的数，而是一个**Carmichael number**。

Carmichael number的定义是，对于一个合数n，如果所有与n互质的正整数b，都有$b^{n-1} \equiv 1 \pmod{n^2}$成立，则$n$称为Carmichael number。其中$0 < b < n$

$b^{n-1} \equiv 1 \pmod{n^2}$的意思是，$b^{n-1}$对$n^2$求余的值与1相等。

$r^{\lambda n} \bmod n^2 = 1$其实是Carmichael's theorem的一个推论，这里不再详细说明。

$L\left(g^{\lambda m} \bmod n^2\right) \equiv \lambda m \pmod n$这里可以由上面二项式定理部分得出。

> 对两个数的乘积求余，与对这两个数先求余再相乘的结果相同。

Paillier同态加密性质

加法

$$D\left(E\left(m_1, r_1\right) \cdot E\left(m_2, r_2\right) \bmod n^2\right) = m_1 + m_2 \bmod n \tag{6}$$

这个公式的意思是，明文m_1和m_2，随机选择的加密因子r_1和r_2。m_1和m_2加密后相乘再解密的结果，与$m_1 + m_2$对n求余的结果相同。

$$D\left(E\left(m_1, r_1\right) \cdot g^{m_2} \bmod n^2\right) = m_1 + m_2 \bmod n \tag{7}$$

乘法

$$D\left(E(m_1, r_1)^{m_2} \bmod n^2\right) = m_1 m_2 \bmod n \tag{8}$$

$$D\left(E(m_2, r_2)^{m_1} \bmod n^2\right) = m_1 m_2 \bmod n \tag{9}$$

一般地

$$D\left(E(m_1,r_1)^k \bmod n^2\right) = km_1 \bmod n \tag{10}$$

证明

这里只证明公式(6)，其他的公式的证明与此都是类似的。

$$E\left(m_1,r_1\right) = c_1 = g^{m1} \cdot r_1^n \bmod n^2 \tag{11}$$

$$E\left(m_2,r_2\right) = c_2 = g^{m2} \cdot r_2^n \bmod n^2 \tag{12}$$

$$c_1 \cdot c_2 = g^{m1+m2} \cdot \left(r_1 \cdot r_2\right)^n \bmod n^2 \tag{13}$$

$$\begin{aligned} D\left(c_1 \cdot c_2\right) &\equiv L\left(g^{\lambda(m_1+m_2)} \cdot \left(r_1 \cdot r_2\right)^{\lambda n} \bmod n^2\right) \cdot \mu \bmod n \\ &\equiv \lambda(m_1+m_2) \cdot \mu \pmod n \\ &\equiv m_1+m_2 \pmod n \end{aligned} \tag{14}$$

其中$(r_1 \cdot r_2)^{\lambda n} \equiv 1 \pmod{n^2}$，依然是利用Carmichael's theorem。也不要忘了$\mu = \lambda^{-1}$。

证毕。

实战

这里主要是利用python-paillier开源库使用一下同态加密。

安装paillier

```
pip3 install phe
```

使用

```
from phe import paillier
# 生成公钥、私钥
public_key, private_key = paillier.generate_paillier_keypair()

secret_number_list = [3.141592653, 50000, -4.6e-12]
# 加密
encrypted = [public_key.encrypt(x) for x in secret_number_list]
# print(encrypted)
# 解密
decrypted = [private_key.decrypt(x) for x in encrypted]
print(decrypted)
```

我们可以看下p、q的值和加密后的结果：

p

```
1758244117241871270451181711936927378658916851135093246753068965704485529054648
8109557144394573259681399744292712671058039426817211010490570409306604297886614
1152868937284582062138112406733477512755929323707214538332342044351693933611 64
9857207220600936239062316751726722695929008411793268171572452365889938772852 29
3883289085580724048588769717782371483001544498874988353471375106043794656801 47
22731451350875383596957118960305378234447190645802240185761002489731141 99
```

q

```
2391746262287096678009862671278112852508174119019981182511251638851571661546 76
6300689861042433281757196313413904474956303420023796871665278982790122911937 68
0885001272769868933425948638604899592198945143300023818931720815347225520193 14
9860962102155159091720360544732727155631258719378409059269523502171905002772 55
4573809846022514594762571010353800131808117883074651094376387785811954566395 63
82845811224174803332682004581557186007407331725911731745357670033499911 79
```

3.141592653加密后的结果

```
1381974166757329134931696491309738751738404678061198859921885723025459921913 38
5123982375212079025160252812015920472378042879023897715633365528891206710990 14
1710746698179480921166073146020391382718721090822871717957943612666076607113 78
0804064951383323489884063653090604559784914051934231826747470267214653802057 34
1917918102675193701050612310080687153057809205587710528225754540256786884001 58
5334625543399342772955263493309376816428108442165984466742102878116898105396 14
9799946931928622513787326765247038871867030737310578651737200204643323752801 72
0507814082280932646475262712803408070281885491086860368226159163355776401944 74
1373245284526926925837597363064625543512950655279911334381556797632954512548 82
4732822124609744622446541160810619599902426263555577605738585590743561434246 81
7713608919269603043343743559600023426478151353980949366038213207902651214673 72
8868498975985261618622329477422514588288645471262687676716995078088600544143 78
6864982700763427910036727970675187085387535166360377091101367997044830092862 21
8885631091563750911767351728347251285913979628257817082670933839941234280226 63
1530293412197074413698543562559402911368845700534561312633781138352131272302 41
6416889689492690472158138661410276153471201082493268234352235475248468744523 07
1186688095030235813100511965200473468314200800229604081002804037101300577801 52
3522867257987642352584673420684929171480479621291775066014302601631868338787 24
7134926970118581951351302157051716044870361732483290226383791825839323516140 54
2544150720514396132482992690154454065873695849806001408373281751007412496132 67
8675813376119531021771373949892176976601417690990357361913295233505555916219 51
0086249038794178129026740718239007960579036472348084181085392204825098185047 42
3818846203071388139632323477809881865349732317822253883638845097716644972754 21
260118333197902727070650357059635097564652770269469811 85
```

50000加密后的结果

```
709312826616425899170757880227690317948720915287968716705639395798782102349610
133746936326439094395632072029437972310438911469934968466763023376417324334536
721347961489013748945008676035118840673058124178023189565779626858042747355376
854427838562543146892259640614124327369617985988040623373982625142161819542135
288635829845182575536386881356850920701757042582055347947345909937969825719624
605702335510452435511346260098900735941769761945752828787497606087877681112192
200211963148202110233587838298755813088102806594798804568146572932472775608991
764613578914261919167120589925011455113275307088065289499879294309900104587562
805435699961569205898216174185732740041305728626327631144616282358235414818312
395225776089193797020222694623755441464578760912600715843298813204732838182523
824230600454001487468133824853603170362362090540681323663765477380094930708010
274620784376543139966804194039484730555241872752343993874884547442696132517003
811373627152303044104643553745338709201093505187457231952256143467081208351938
139536893665006271311902014123870129613091392085227158111542092529832783814001
831913685068658085703228963298877469040638999458549685444989652702796555968202
262573642612987187930515949802067478110323313764910864814362242544627645349383
299787571964788763434624180879881778528800229827117218648454462022057555757640
434930954380044164918514491429811428386542193646138448558368811774513662112946
975106262791179073873037123457280857047809741456747745847970591353009088814236
176458627323736711168012437668511187827937971335361988706882916191915925735638
897314098856850315043028103483736179726562899080621166351089804893898019786051
863319034203717656549177386858003509502073551263267375687582878543908212287090
205014341241784997626029017061726227453216055631441235651779739019827648811760
7951206230874810943238181618111150999369610283757267447
```

-4.6e-12加密后的结果

```
172749040835556474753526170265542611349203963266911977837943548080064827658754
570250003698959099335601088483058734687730963681443533747597516125923597057524
371336553141318394734172228748383064816328941410321595916399565412261744129349
778406214868090277894646462486148446123811901815897290607096938508733239316407
166709815318474303674816443652117728035872132703253488665898328958968354330783
827871008955021534988114131942440469156874704133488642273446637221275237574677
040035245721167950596788356470678664040768667713411377415515703456840935599045
602982944338991054461373652348945759137893437064169125907137794231898250195099
510623892527444969217660857352841634985360679786698272826208386785256857219536
815560533980765293483173195427296342450638970691719917939592407125771452089177
819145703225048277613734073726994083280880709364355678439946889761371702661894
979664316455809833532041646281879122877369152649231081084283429782642693557840
047672268134104686000442895363433815965411596753779140262660663574784523874535
096315677420130963206556978453614338283514241226087665901943602528973224491075
536809722276627754637263015468774722891561748181986552415266299426868157318410
381172807548011862765264879325184438603986854714491888726560935941459385149933
503238378331159802266725470185233662091944659498163873994101605585980504385739
814485910488961873981343291814772059819066928705543265149967748402009797419528
063480515434288462770641257729782365845758222872736378069271492623818310810444
439249884675250447171881975872545022329836737440916268821876236341948359381410
783541885808432784176458299311106291189137544916902441378798828040841817699054
816296703013147497994371421721434499039958238767486848455360693949848354907221
164762414058406431917176996333903392370034679794306460602465761003869328604956
16028150350916448310490218903889826694615824406895220483
```

> 为什么加密若干次数后需要重新生成公钥和私钥?

Carmichael 数，也就是这里的n，是非常少的，据统计，在1~100000000范围内的整数中，只有255个Carmichael 数；与n互质且$0 < r < n, r \in \mathbb{Z}_n^*$的$r$也是有限的，从上面输出的p、q可以看出，n已经非常大了，但是r的选择并不是无穷的，如果选择到了重复的r，密文c就具有一定得相似性，明文m就有泄露的风险，所以加密若干次数后一般需要重新生成公钥和私钥。

从p、q的数量级可以看出，频繁重新生成公钥和私钥也是没有必要的，n非常大，满足条件的r很多。

参考链接

- [Carmichael's theorem]
- [Modular multiplicative inverse]
- [Paillier cryptosystem]
- Paillier. 1999. Public-Key Cryptosystems Based on Composite Degree Residuosity Classes
- [python-paillier]