Paillier Cryptosystem

Автор:
Андреј Бардакоски

Јуни 2023 година

Содржина

Вовед

Криптографија со јавен клуч

Криптографија со јавен клуч или асиметрична криптографија е област во криптографијата која ги изучува криптографските алгоритми кои користат пар од клучеви: јавен клуч и таен клуч.

Клучевите се генерираат со алгоритми кои се темелат на математички проблеми наречени trapdoors или one-way functions тоа се функции кои лесно се пресметуваат во една насока но многу тешко во обратната насока.

Тајниот односно приватниот клуч се чува како тајна односно не се споделува, додека јавниот клуч не се крие и е јавно објавен. При енкрипција на порака се користи јавниот клуч па така секој што го има јавниот клуч може да енкриптира порака. Пораката може да биде декриптирана само користејќи го тајниот клуч односно само од оној што го има тајниот клуч

Paillier cryptosystem

Paillier cryptosystem е асиметричен алгоритам за криптографија со јавен клуч, измислен и именуван по Паскал Паилер (Pascal Paillier) во 1999. Паилер криоптосистемот се темели врз Composite Residuosity Class Problem, кој е пресметковно тежок проблем според математичката хипотеза: Одлучлива композитна претпоставка за остаточност (decisional composite residuosity assumption (DCRA)). Според DCRA за дадени n и z тешко е да се определи дали постои y така што $z \equiv y^n \ (mod\ n^2)$.

Паилер криптосистемот како trapdoor функција користи факторизација.

Трите главни функции во Paillier cryptosystem се алгоритам за: генерирање на клучеви, енкрипција и декрипција.

Генерирање на клучеви

Алгоритмот за генерирање на јавен и приватен клуч се состои од следните чекори:

1. Избери два големи прости броиви $\boldsymbol{p}$ и $\boldsymbol{q}$. Провери дали $\text{НЗД}(\boldsymbol{pq}, (\boldsymbol{p}-\boldsymbol{1})(\boldsymbol{q}-\boldsymbol{1}))$ е 1. Ако не почни од почеток.
2. Пресметај $\boldsymbol{n} = \boldsymbol{pq}$
3. Дефинирај функција $\boldsymbol{L}(\boldsymbol{x}) = \frac{\boldsymbol{x}-\boldsymbol{1}}{\boldsymbol{n}}$
4. Пресметај $\boldsymbol{\lambda}$ како $\text{НЗС}(\boldsymbol{p}-\boldsymbol{1}, \boldsymbol{q}-\boldsymbol{1})$
5. Избери природен број $\boldsymbol{g}$ во множеството природни броеви помеѓу 1 и n^2
6. Пресметај $\boldsymbol{\mu} = (\boldsymbol{L}(\boldsymbol{g}^{\boldsymbol{\lambda}} \ \boldsymbol{mod}\ \boldsymbol{n^2}))^{-\boldsymbol{1}}\ \boldsymbol{mod}\ \boldsymbol{n}$ доколку $\boldsymbol{\mu}$ не постои почни пак од чекор 1.
7. Јавниот клуч е $(\boldsymbol{n}, \boldsymbol{g})$
8. Приватниот клуч е $(\boldsymbol{\lambda}, \boldsymbol{\mu})$

Енкрипција

Алгоритмот за енкрипција се состои од следните чекори:

1. $\boldsymbol{m}$ – порака што се енкриптира
2. Избери случаен број $\boldsymbol{r}$ во опсег $\boldsymbol{0 < r < n}$
3. Пресметај шифриран текст $\boldsymbol{c} = \ \boldsymbol{g^m r^n} \ \ \boldsymbol{mod}\ \boldsymbol{n^2}$

Декрипција

Алгоритмот за декрипција се состои од следната операција:

$$\boldsymbol{m = L\left(c^{\lambda} \mod n^2\right) \cdot \mu \mod n}$$

Пример

Генерирање на клуч

1. Избери $\boldsymbol{p}$ и $\boldsymbol{q}$
   1.1. Нека $\boldsymbol{p = 13}$
   1.2. И нека $\boldsymbol{q = 17}$
   1.3. $\text{НЗД}\left(\boldsymbol{13 \cdot 17}, \boldsymbol{(13-1)(17-1)}\right) = \text{НЗД}(\boldsymbol{221}, \boldsymbol{192}) = \boldsymbol{1}$ (Условот е задоволен)
2. $\boldsymbol{n = pq = 13 \cdot 17 = 221}$
3. $\boldsymbol{L(x) = \frac{x-1}{n} = \frac{x-1}{221}}$
4. $\boldsymbol{\lambda = \text{НЗС}(p-1, q-1) = \text{НЗС}(12, 16) = 48}$
5. Нека $\boldsymbol{g = 4886}$
6. $\boldsymbol{\mu} = (L\left(g^{\lambda} \mod n^2\right))^{-1} \mod n = (L(4886^{48} \mod 221^2))^{-1} \mod 221 = (L(30720))^{-1} \mod 221 = (\frac{30720-1}{221})^{-1} \mod 221 = (139)^{-1} \mod 221 = \boldsymbol{159}$
   6.1. Условот е исполнет μ постои
7. Јавниот клуч е $(\boldsymbol{n}, \boldsymbol{g}) = (\boldsymbol{221}, \boldsymbol{4886})$
8. Приватниот клуч е $(\boldsymbol{\lambda}, \boldsymbol{\mu}) = (\boldsymbol{48}, \boldsymbol{159})$

Енкрипција

1. Нека пораката $\boldsymbol{m = 123}$
2. Нека $\boldsymbol{r = 666}$
3. Шифриран текст $\boldsymbol{c = g^m r^n \mod n^2} = 4886^{123} 666^{221} \mod 221^2 = 42021 \cdot 24696 \mod 48841 = \boldsymbol{25889}$

Декрипција

$\boldsymbol{m} = L\left(c^{\lambda} \mod n^2\right) \cdot \mu \mod n = L(25889^{48} \mod 221^2) \cdot 159 \mod 221 = L(17681) \cdot 159 \mod 221 = \frac{17681-1}{221} \cdot 159 \mod 221 = 80 \cdot 159 \mod 221 = \boldsymbol{123}$

Хомоморфна енкрипција

Хомоморфна енкрипција е форма на енкрипција која овозможува да се извршуваат математички или логички операции врз енкриптираните податоци без да има потреба тие прво да бидат декриптирани. Пример ако имаме два броја m_1 и m_2 и доколку ги енкриптираме користејќи хомоморфна енкрипција која овозможува операција собирање и ги добиваме шифрираните пораки $c_1 = E(m_1)$ и $c_2 = E(m_2)$ тогаш било кој може да ја изврши операцијата собирање $Add(c_1, c_2)$ така што резултатот од операцијата ќе може да се дешифрира во $m_1 + m_2$

$$D(Add\left(E(m_1), E(m_2)\right) = m_1 + m_2$$

Операцијата Add() вообечаено не е буквално собирање туку некоја функција која што го задоволува споменатото својство.

Паилер криптосистемот е парцијалена хомоморфна енкрипциска шема која што овозможува два вида на операции: **Собирање на две шифрирани пораки**; **Множење на шифрирана порака со број** (бројот не е шифриран, plaintext)

Собирање

Equation 1 Хомоморфно собирање

$$D((E(m_1) \cdot E(m_2) \mod n^2) = m_1 + m_2 \mod n$$

Множење

Equation 2 Хомоморфно множење

$$D(E(m_1)^{m_2} \mod n^2) = m_1 \cdot m_2 \mod n$$

Специфични случаи

Има неколку специфични случаи со кои што треба да се внимава. Првиот таков случај е множење со 0. Поради тоа што било кој број на степен 0 е 1 доколку помножиме некој шифриран текст со 0 користејќи го методот *Equation 2* резултатот ќе биде 1 и секој што ќе ја види оваа енкриптирана вредност ќе знае дека се дешифрира во 0. Ние може да ја избегнеме оваа непосакувана ситуација така што наместо да добиеме 0 користејќи го методот *Equation 2* ние може директно да енкриптираме 0 користејќи го стандардниот метод за енкрипција, па така од шифрираниот текст може да се добие оригиналната порака 0 само користејќи го тајниот клуч.

Вториот специфичен случај е при множење со 1. Поради тоа што кој било број на степен 1 е самиот број, па ако го помножиме некој шифриран текст со 1 користејќи го методот *Equation 2* резултатот ќе биде самиот шифриран текст, ова не е толку страшно бидејќи не се открива кој е шифрираниот текс, но сепак е проблем бидејќи секој кој што ја следи комуникацијата ќе може да дознае дека шифрираниот текст е помножен со 1. Решение за овој проблем во ваква ситуација е наместо да се кори методот за множење *Equation 2* да се искористи собирање со 0 односно методот *Equation 1* па така ќе се добие сигурна шифрирана вредност

Примери

Хомоморфно собирање

Ќе ја искористиме пораката од првиот пример $\boldsymbol{m_1 = 123}$ за која што добивме шифриран текст $\boldsymbol{c_1 = 25889}$ користејќи го јавниот клуч $(\boldsymbol{221, 4886})$ и тајниот клуч $(\boldsymbol{48, 159})$ и дополнително уште една порака $\boldsymbol{m_2}$ која ќе ја додадеме на првата порака

1. Нека пораката $\boldsymbol{m_2 = 37}$
2. Нека $\boldsymbol{r = 999}$
3. Шифриран текст $\boldsymbol{c_2 = g^{m_2} r^n \mod n^2} = 4886^{37} 999^{221} \mod 221^2 = 31393 \cdot 11811 \mod 48841 = \boldsymbol{30692}$
4. Шифрирана сума $\boldsymbol{c_{sum} = c_1 \cdot c_2} \mod n^2 = 25889 \cdot 30692 \mod 221^2 = \boldsymbol{39800}$
5. Декриптирана сума $\boldsymbol{m_{sum}} = L\left(c_{sum}^{\lambda} \mod n^2\right) \cdot \mu \mod n = L(39800^{48} \mod 221^2) \cdot 159 \mod 221 = L(30941) \cdot 159 \mod 221 = \frac{30941-1}{221} \cdot 159 \mod 221 = 140 \cdot 159 \mod 221 = \boldsymbol{160}$
6. Проверка: $\boldsymbol{m_1 + m_2 = 123 + 37 = 160 = m_{sum}}$

Хомоморфно множење

Повторно ќе ги искористиме податоците од првиот пример дополнително уште една порака $\boldsymbol{m_3}$ со која ќе ја помножиме првата порака

1. Нека пораката $\boldsymbol{m_3 = 25}$
2. Шифриран производ $\boldsymbol{c_{prd}} = {c_1}^{m_3} \mod n^2 = 25889^{25} \mod 221^2 = \boldsymbol{15723}$

3. Декриптиран прозивод $\boldsymbol{m_{prd}} = L\left(c_{prd}^{\lambda} \mod n^2\right) \cdot \mu \mod n = L(15723^{48} \mod 221^2) \cdot 159 \mod 221 = L(2432) \cdot 159 \mod 221 = \frac{2432-1}{221} \cdot 159 \mod 221 = 11 \cdot 159 \mod 221 = \mathbf{202}$
4. Проверка: $\boldsymbol{m_1} \cdot \boldsymbol{m_3} \bmod n = 123 \cdot 25 \mod 221 = 3075 \bmod 221 = \mathbf{202} = \boldsymbol{m_{prd}}$

Практичен пример, код

Ја користиме имплементацијата на Паилер криптосистемот во python: [python-paillier]

```
git clone https://github.com/data61/python-paillier.git
```

Генерирање на клуч

```
Python Console
>>> from phe import paillier
>>> public_key, private_key = paillier.generate_paillier_keypair()
```

Порака 1

Енкрипција

```
>>> plaintext1 = 1122334455
>>> ciphertext1 = public_key.encrypt(plaintext1)
>>> ciphertext1.ciphertext()
105727509050457657659312615860031250205214223809028187802234806075014141727490380379778216401401321022249568
```

Декрипција

```
>>> decrypted1 = private_key.decrypt(ciphertext1)
>>> decrypted1
1122334455
```

Проверка декриптираната вредност се совпаѓа со оригиналната

```
>>> decrypted1 == plaintext1
True
```

Порака 2

Енкрипција

```
>>> plaintext2 = 3333333333
>>> ciphertext2 = public_key.encrypt(plaintext2)
>>> ciphertext2.ciphertext()
181388569671051144260920482927445805053936492968974894273231118870169736641594161878077571367140489488
```

Декрипција

```
>>> decrypted2 = private_key.decrypt(ciphertext2)
>>> decrypted2
3333333333
>>> plaintext2 == decrypted2
True
```

Хомоморфно собирање

Собирање на шифрирани пораки

```
>>> sum_ciphertext = ciphertext1 + ciphertext2
>>> sum_ciphertext.ciphertext()
52255980120479124311475137369922363335848898312282853222671251492489418709645273910850050219
```

Забелешка: операцијата собирање е преоптоварена (overloaded) па така собирањето на шифрирани пораки не е буквално собирање туку се користи методот *Equation 1* споменат предходно

Декрипција

```
>>> sum_decrypted = private_key.decrypt(sum_ciphertext)
>>> sum_decrypted
4455667788
```

Проверка

```
>>> sum_plaintext = plaintext1 + plaintext2
>>> sum_plaintext
4455667788
>>> sum_plaintext == sum_decrypted
True
```

Целиот код

```
from phe import paillier


def example():
    # generating key pair
    public_key, private_key = paillier.generate_paillier_keypair()

    # msg 1
    plaintext1 = 1122334455  # message 1 as plaintext
    ciphertext1 = public_key.encrypt(plaintext1)  # ciphertext 1, encrypted message 1
    decrypted1 = private_key.decrypt(ciphertext1)  # we decrypt the ciphertext 1

    # Checking the decrypted number is what we started with
    assert (plaintext1 == decrypted1)

    # msg 2
    plaintext2 = 3333333333  # message 2 as plaintext
    ciphertext2 = public_key.encrypt(plaintext2)  # ciphertext 2, encrypted message 2
    decrypted2 = private_key.decrypt(ciphertext2)  # we decrypt the ciphertext 2

    # Checking the decrypted number is what we started with
    assert (plaintext2 == decrypted2)

    # Homomorphic addition
    sum_ciphertext = ciphertext1 + ciphertext2
    sum_decrypted = private_key.decrypt(sum_ciphertext)
    sum_plaintext = plaintext1 + plaintext2

    # Checking the decrypted sum is equal to the sum of the plaintexts
    assert (sum_plaintext == sum_decrypted)


if __name__ == "__main__":
    example()
```

Споредба со RSA и ElGamal

На селдната слика е прикажана теоретска споредба на перформанси на Paillier Cryptosystem со RSA и ElGamal

Schemes	Main Scheme	RSA	ElGamal
One-wayness	Class $[n]$	RSA $[n, F_4]$	DH $[p]$
Semantic Sec.	CR $[n]$	none	D-DH $[p]$
Plaintext size	$\|n\|$	$\|n\|$	$\|p\|$
Ciphertext size	$2\ \|n\|$	$\|n\|$	$2\ \|p\|$
Encryption			
$\|n\|, \|p\| = 512$	5120	**17**	1536
$\|n\|, \|p\| = 768$	7680	**17**	2304
$\|n\|, \|p\| = 1024$	10240	**17**	3072
$\|n\|, \|p\| = 1536$	15360	**17**	4608
$\|n\|, \|p\| = 2048$	20480	**17**	6144
Decryption			
$\|n\|, \|p\| = 512$	768	**192**	768
$\|n\|, \|p\| = 768$	1152	**288**	1152
$\|n\|, \|p\| = 1024$	1536	**384**	1536
$\|n\|, \|p\| = 1536$	2304	576	2304
$\|n\|, \|p\| = 2048$	3072	768	3072

Може да заклучиме дека Paillier Cryptosystem има послаби перформанси од RSA и ElGamal но сепак разликата не е драстична. Имаме недостатокот на перформанси но од друга страна Paillier Cryptosystem овозможува хомоморфни операции, па така би бил корисен во ситуации кога има потреба од шифрувач кој овозможува хомоморфни операции, односно кога би можеле да ги искористиме хомоморфните својства.

Примена

Паилер криптосистемот најчесто се користи при:

Electronic voting (е-гласање) и Electronic Cash

Доколку имаме потреба од систем со едноставно бинарно гласање да / не и сакаме да им овозможиме на гласачите нивниот одговор да биде анонимен може да го дизајнираме системот така што гласачите ќе гласаат така што нивниот одговор да биде 1 кој што означува ДА или 0 што ќе означува НЕ и гласачите ќе го енкриптираат нивниот одговор користејќи го Паилер криптосистемот и како таков ќе го постават во системот потоа одржувачот на гласањето ќе ги собере гласовите користејќи го својството за хомоморфно собирање и по декрипција резултатот ќе биде вкупниот број на гласови ДА. Со тоа системот успешно ја завржил задачата

за пребројување на гласови и притоа е задржана анонимноста на одговорите од гласачите, (ниту еден одговор не е декриптиран, декриптиран е само вкупниот збир од одговорите)

Threshold cryptosystem

Threshold cryptosystem е криптосистем кој што при енкрипција на порака користи јавен клуч, а тајниот клуч е поделен помеѓу неколку учеснички странки. За да се декриптира пораката или пак да се потпише некоја порака потребена е соработка од одреден број (threshold number) на странки.

Хомоморфните својства на Паилер криптосистемот понекогаш се користат за да се направи Threshold ECDSA потпис.

Private set intersection

Пресек на приватни множества (PSI - Private set intersection) е кога две странки од кој секоја имаа множество на податоци од некој домен и сакаат да го дознаат пресекот помеѓу двете множества без притоа да ги откријат останатите елементи кои се наоѓаат во соодветното множество. PSI – може да се имплементира со користње на хомоморфните својства на Паилер криптосистемот.

Заклучок

Паилер криптосистемот е асиметричен криптографски алгоритам кој воедно е и парцијално хомоморфен шифрувач кој што овозможува операција собирање и операција множење со број. Паилер криптосистемот главно се упетребува само во ситуации кога имаме потреба од хомоморфните својства што ги нуди.

Референци

Паилеров оригинален научен труд

github.com/data61/python-paillier – Имплементација на Pailler crypthosystem во python

Wikipedia - Paillier cryptosystem

Wikipedia - DCRA