数据隐私实验

第一部分 Paillier

PB16001680 李瀚民

[代码补全]:

加密:

```python
def enc(pub, plain):
    while True:
        r = mpz_urandomb(rand, pub.bits)
        if r < pub.n and r > 0 and gcd(r, pub.n) == 1:
            break
    cipher = (powmod(pub.g, plain, pub.n_sq) * powmod(r, pub.n, pub.n_sq)) % pub.n_sq
    return cipher
```

解密:

```python
def dec(priv, pub, cipher):
    x = powmod(cipher, priv.l, pub.n_sq)
    plain = (((x - 1) // pub.n) * priv.m) % pub.n
    return plain
```

加法:

```python
def enc_add(pub, m1, m2):
    """Add one encrypted integer to another"""
    add_result = m1 * m2 % pub.n_sq
    return add_result
```

加常数:

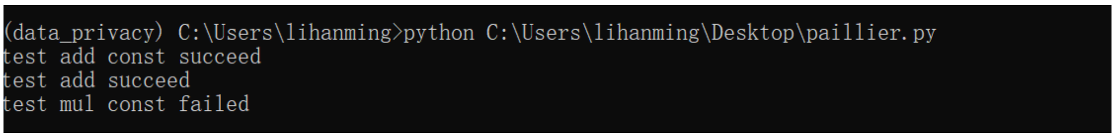
```python
def enc_add_const(pub, m, c):
    """Add constant n to an encrypted integer"""
    # Similiar to enc add
    add_const_result = m * powmod(pub.g, c, pub.n_sq) % pub.n_sq
    return add_const_result
```

乘常数:

```python
def enc_mul_const(pub, m, c):
    """Multiplies an encrypted integer by a constant"""
    mul_result = powmod(m, c, pub.n_sq)
    return mul_result
```

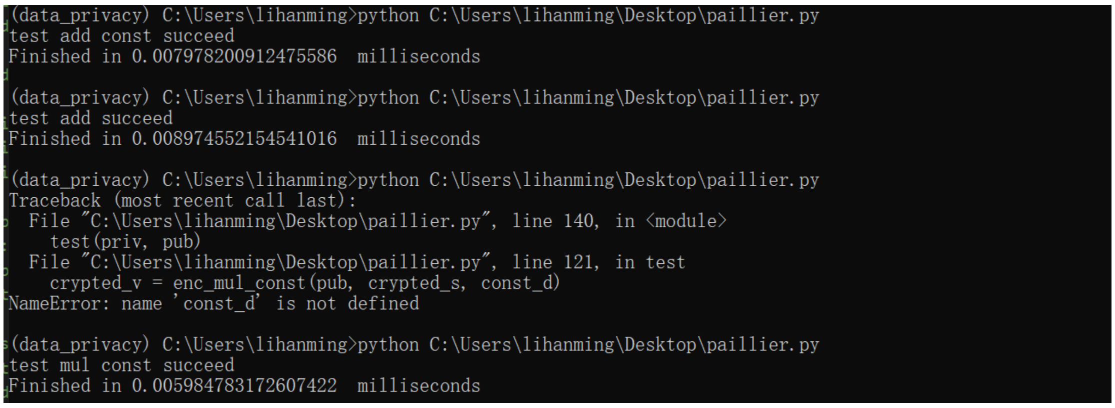
[实验结果]:

具体测试代码后面给出，基本都是通过先加密两个数，然后进行乘法或者加法，在解密出结果，比较结果是否等于原来两数和或积



```
(data_privacy) C:\Users\lihanming>python C:\Users\lihanming\Desktop\paillier.py
test add const succeed
test add succeed
test mul const failed
```

用时：通过合理注释掉一些代码，来得到各个测试的运行时间，图中第三次多注释了一行，请忽略



```
(data_privacy) C:\Users\lihanming>python C:\Users\lihanming\Desktop\paillier.py
test add const succeed
Finished in 0.007978200912475586  milliseconds

(data_privacy) C:\Users\lihanming>python C:\Users\lihanming\Desktop\paillier.py
test add succeed
Finished in 0.008974552154541016  milliseconds

(data_privacy) C:\Users\lihanming>python C:\Users\lihanming\Desktop\paillier.py
Traceback (most recent call last):
  File "C:\Users\lihanming\Desktop\paillier.py", line 140, in <module>
    test(priv, pub)
  File "C:\Users\lihanming\Desktop\paillier.py", line 121, in test
    crypted_v = enc_mul_const(pub, crypted_s, const_d)
NameError: name 'const_d' is not defined

(data_privacy) C:\Users\lihanming>python C:\Users\lihanming\Desktop\paillier.py
test mul const succeed
Finished in 0.005984783172607422  milliseconds
```

[测试代码]：

```
def test(priv, pub):
    # Get boundaries for integers
    lower_bound = 2 ** 10
    upper_bound = 2 ** 1000

    # Get time
    start_time = time.time()

    # Test

    # Set numbers
    x = random.randint(lower_bound, upper_bound)
    y = random.randint(lower_bound, upper_bound)
    s = random.randint(lower_bound, upper_bound)
    const_c = random.randint(lower_bound, upper_bound)
    const_d = 3
    crypted_x = enc(pub, x)
    crypted_y = enc(pub, y)
    crypted_s = enc(pub, s)

    # test add const
    crypted_u = enc_add_const(pub, crypted_x, const_c)
    u = dec(priv, pub, crypted_u)
    # end_time_add_const = time.time()
    if u == x + const_c:
        print('test add const succeed')
    else:
        print('test add const failed')
    # print('Finished in ' + str(end_time_add_const - start_time) +  '
 milliseconds')
```

```
    # test add
    crypted_z = enc_add(pub, crypted_x, crypted_y)
    z = dec(priv, pub, crypted_z)
    # end_time_add = time.time()
    if z == x + y:
        print('test add succeed')
    else:
        print('test add failed')
    # print('Finished in ' + str(end_time_add - start_time) +  '  milli
seconds')

    # test mul const
    crypted_v = enc_mul_const(pub, crypted_s, const_d)
    v = dec(priv, pub, crypted_v)
    # end_time_mul = time.time()
    if v == s * const_d:
        print('test mul const succeed')
    else:
        print('test mul const failed')
    # print('Finished in ' + str(end_time_mul - start_time) +  '  milli
seconds')

    return
```