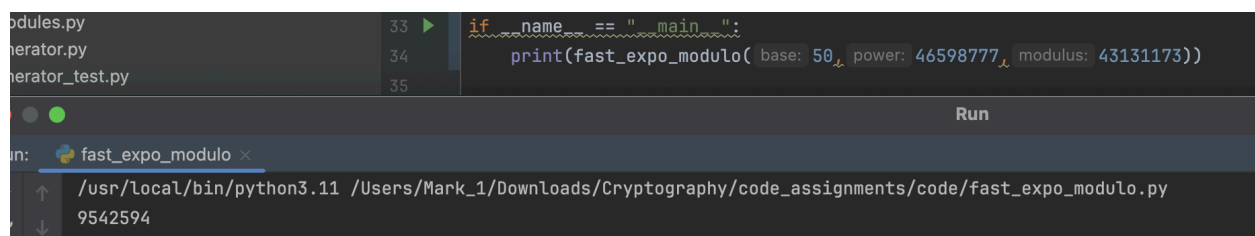
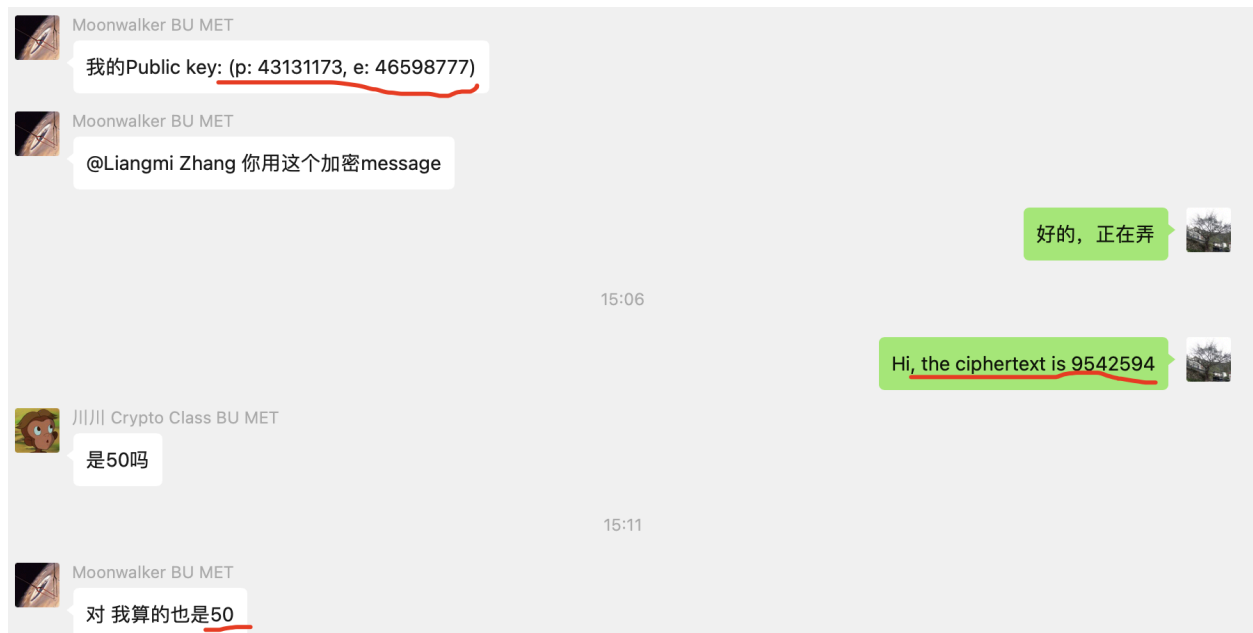


## Exchange and Computation History

(I'm the guy on the right in the screenshots)

### RSA

I'm Alice

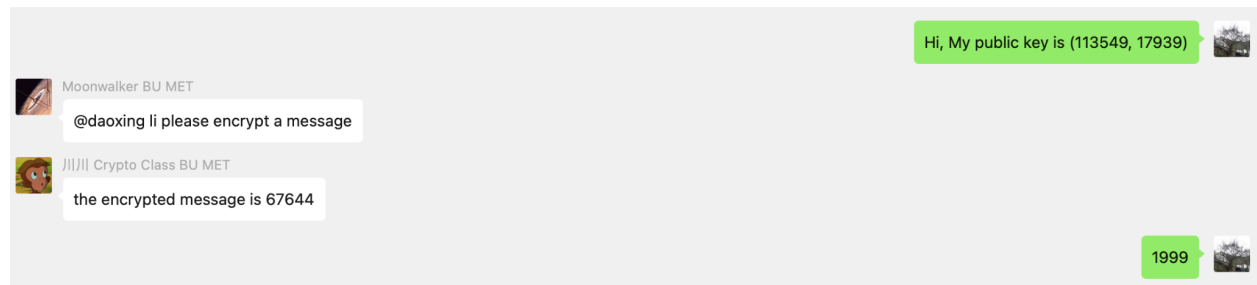


I'm Bob

Public key generation

```
rs (cryptography) ~/Downloads/... class MyTestCase(unittest.TestCase):
14 def test(self): self.assertRaises(ValueError, testMethodTest)
15
16 n = p * q # 113549
17 phi_of_n = (p - 1) * (q - 1) # phi_of_n: 112860
18 e = 2 # 17939
19 while phi_of_n % e == 0: # that is, gcd(e, phi_of_n) != 1
20     e = e * 2
21     phi_of_n = phi_of_n // 2
22 public_key = (e, n) # public_key: (113549, 17939)
23 # receiver generates private key
24 private_key = rsa_private_key_generator(public_key, p, q) # private_key: (113549, 30299)
25 # sender generates the message
26 message = random.randint(0, n - 1) # 1999
27 while euclidean(message, n) != 1:
28     message = random.randint(0, n - 1)
29 # sender encrypts the message
30
```

## Message Decryption:

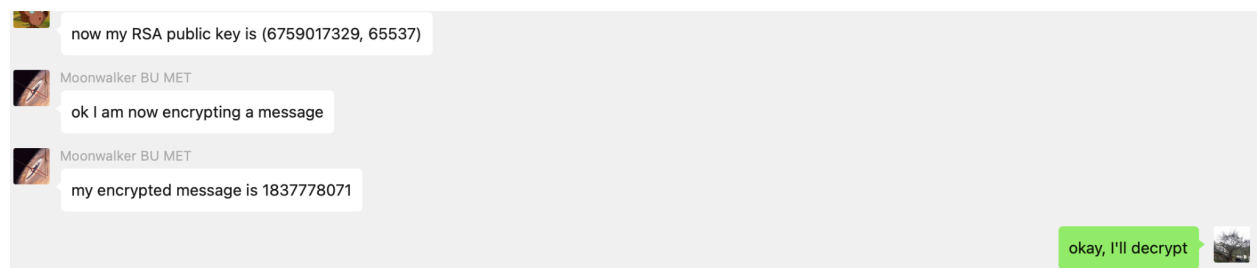


```
pollard_p_1_factorization_test_factorial_version_test.py 60
pollard_rho_factorization.py 61
pollard_rho_factorization_test.py 62
primitive_root_search.py 63
project_user_manual.pdf 64
random_number_generator.py 65
random_number_generator_test.py 66
random_prime_generator.py 67

__name__ == "__main__":
print(rsa_decrypt(rsa_private_key_generator( public_key: (113549, 17939), prime_p: 419, prime_q: 271), ciphertext: 67644))

Run
Run: rsa_without_padding x
/usr/local/bin/python3.11 /Users/Mark_1/Downloads/Cryptography/code_assignments/code/rsa_without_padding.py
1999
```

I'm Eve



```
63  
64  
65 print(break_rsa_with_pollard_rho( public_key: (6759817329, 65537), ciphertext: 1837778071))  
66
```

Run

Run: break\_rsa ×

/usr/local/bin/python3.11 /Users/Mark\_1/Downloads/Cryptography/code\_assignments/code/break\_rsa.py

178

## EL GAMAL

I'm Alice

prime = 80687

$(b, b^r) = (26833, 55116)$

b is the primitive root of p and r is my private key.

Debug: Python tests in break\_el\_gamal\_test.py ×

Debugger Console

Mai...read Evaluate expression (⇧⌘) or add a watch (⇧⌘⇧)

test2, break_el...	b = {int} 26833
_callTestMethod...	b_to_power_of_r = {int} 55116
run, case.py:62	index = {int} 0
_call_, case.p...	prime = {int} 80687
run, suite.py:12	r = {int} 6921
_call_, suite.p...	self = {MyTestClass} <break_el_gamal_test.MyTestClass testMethod=test2>
run, suite.py:12	

I'm Bob

## Private Key Calculation:

```
def el_gamal_receiver_prepare(b, b_to_power_of_r, prime):    b: 2    b_to_power_of_r: 362690    prime: 839731
    """
    for receiver to do precomputations
    :param b: integer, which is the primitive root of the group
    :param b_to_power_of_r: which is  $b^r \bmod p$ , shared by the sender.
    :param prime: a prime number
    :return:
        brl, an integer, which is  $(b^r)^l \bmod \text{prime}$  ;
        b_to_power_of_l, an integer, which is  $(b^l) \bmod \text{prime}$  ;
        brl_inverse, and integer, which is the inverse of brl in group  $U(\text{prime})$ 
    """
    # check the primality of the argument prime
    if not is_prime(prime):
        raise Exception(f"{prime} is not prime. Argument should be prime.")

    l = random.randint(a=0, prime - 2) # generate a private key    l: 88441
    b_to_power_of_l = fast_expo_modulo(b, l, prime)    b_to_power_of_l: 52277
    # computes  $(b^r)^l$ 
    brl = fast_expo_modulo(b_to_power_of_r, l, prime)    brl: 199954
    # computes inverse of  $(b^r)^l$  in G
    brl_inverse = extended_euclidean_2(brl, prime)[0]    brl_inverse: 132939
    return brl, b_to_power_of_l, brl_inverse
```



Daoxing Li

to me ▾

1:56 PM (1 hour ago)



Hi, Liangmi,

Please send me an encrypted message using the following numbers:

prime = 839731

(g, h) = (2, 362690)

g is the primitive root of the prime, and  $h = g^x \bmod \text{prime}$  where x is my private key.

Best

Daoxing

Liangmi Zhang <[zhanlian@bu.edu](mailto:zhanlian@bu.edu)> 于2024年12月9日周一 13:35写道:

...



Liangmi Zhang <[zhanlian@bu.edu](mailto:zhanlian@bu.edu)>

to Yiyao, Daoxing ▾

2:30 PM (1 hour ago)



hi, Daoxing

I is my private key.

$b^l = 88441$ . Please send me ciphertext with EL GAMAL

Liangmi

## Decryption:

I'm Eve