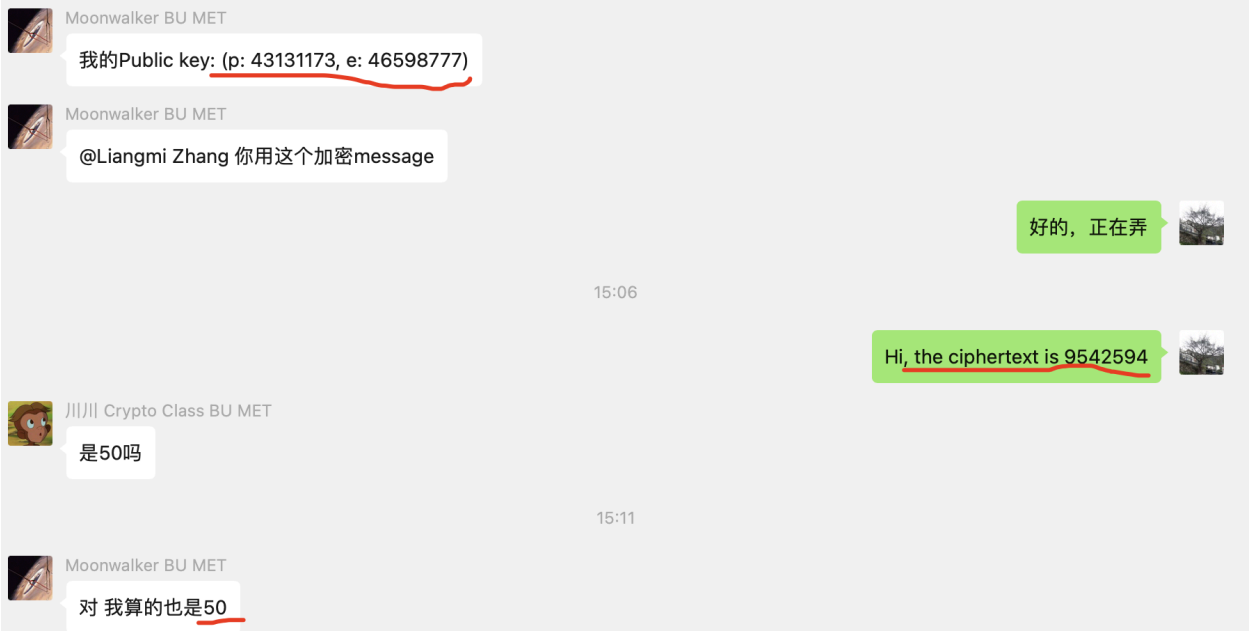
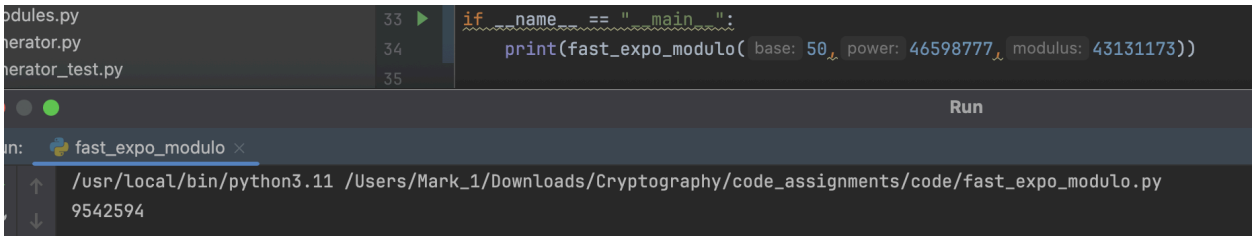
Exchange and Computation History

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

RSA

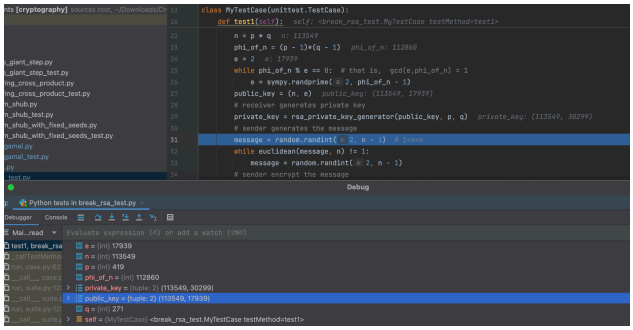
I’m Alice



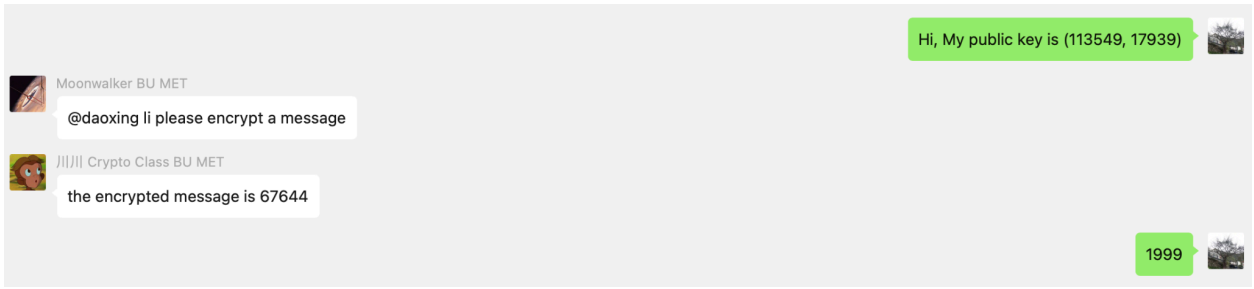


I’m Bob

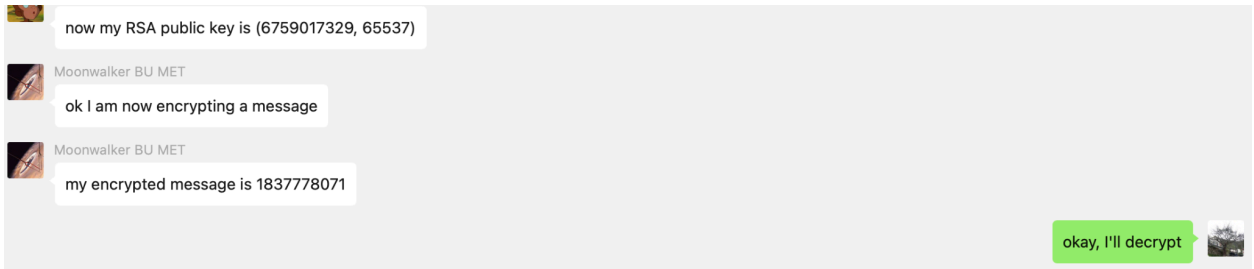
Public key generation

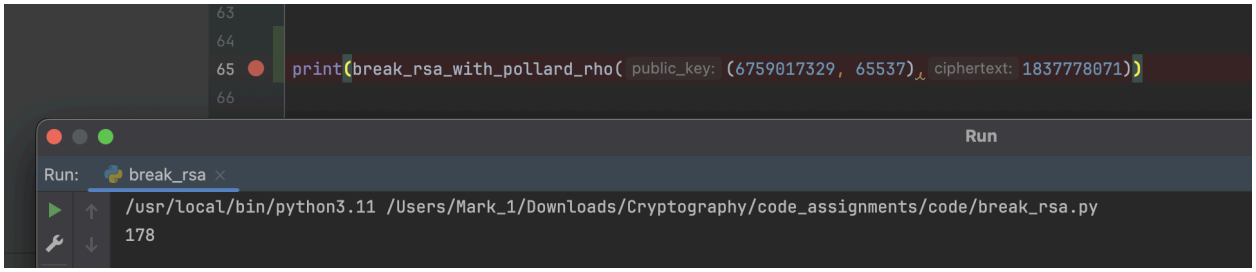


Message Decryption:



I’m Eve





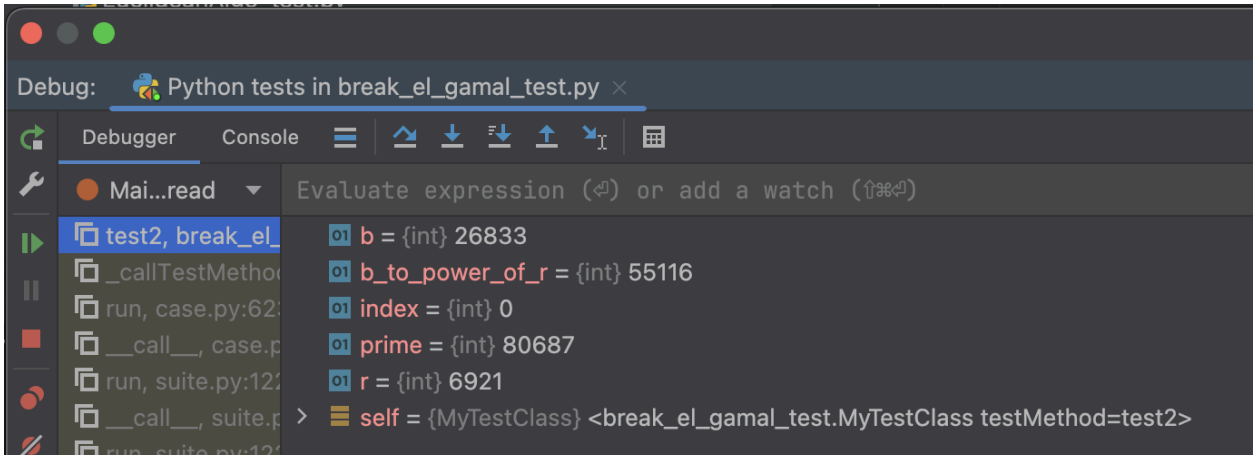
EL GAMAL

I'm Alice

prime = 80687

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

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

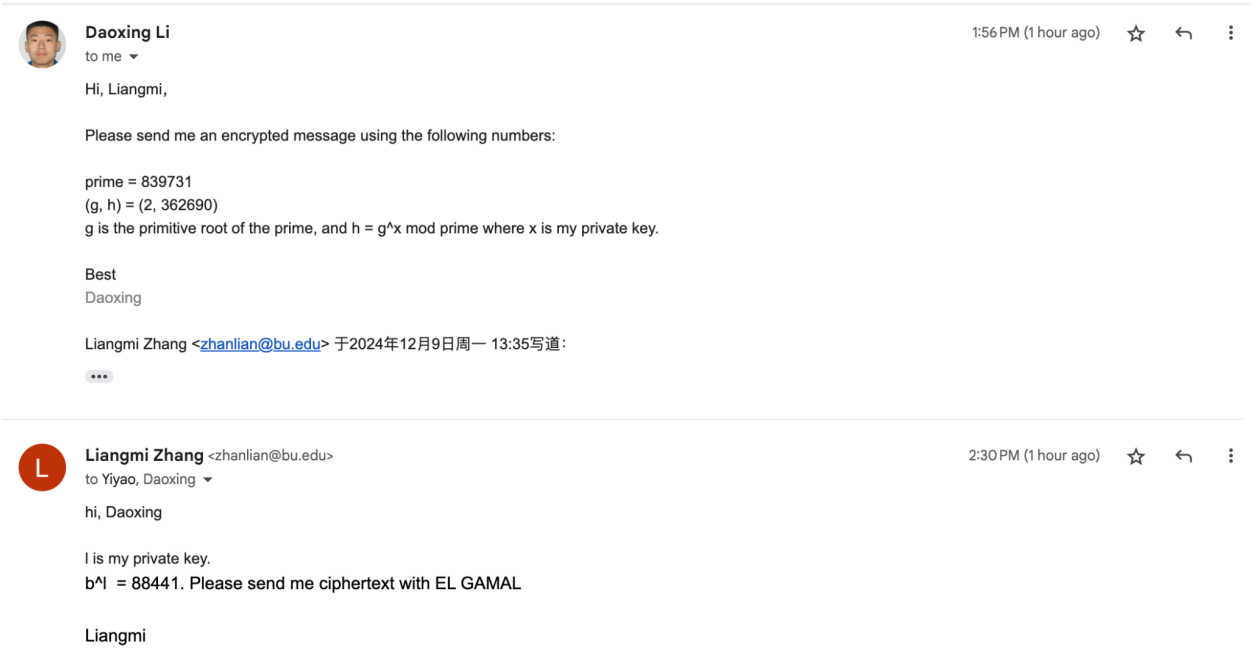


I'm Bob

Private Key Calculation:

```python
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 mod p, shared by the sender.
    :param prime: a prime number
    :return:
    brl, an integer, which is (b^r)^l mod prime ;
    b_to_power_of_l, an integer, which is (b^l) mod prime ;
    brl_inverse, and integer, which is the inverse of brl in group U(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 1:56 PM (1 hour ago)

to me

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 mod prime where x is my private key.

Best
Daoxing

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

Liangmi Zhang <zhanlian@bu.edu> 2:30 PM (1 hour ago)

to Yiyao, Daoxing

hi, Daoxing

I is my private key.
b^I = 88441. Please send me ciphertext with EL GAMAL

Liangmi

Decryption:

I’m Eve