1. Use of ElGamal encryption to

a. Encrypt and send a message (Alice)

5. Receive informations from Bob(In the class object)

6. randomly pick one element from group F

7. computes 2 cipher texts, c1 = g^r mod p, c2 = y^r mod p, which is the encrypt function in the program

8. multiply the second cipher text with the message she wants to send (return value of encrypt function)

9. sends these 2 cipher text to Bob

b. Receive and decrypt a message (Bob)

1. chooses a very large number p(using random\_prime function) and a cyclic group F

2. from F, Bob randomly picks a primitive root g to be the generator, in this program he could using primitiveRoots(p) which takes the large prime number as input.

3. randomly pick a number a as private key and calculate public key y=g^a mod p, in this program this will complete in the class constructor

4. publish the public key y, generator g, and number p, in the program, these are already in the class object

10. using decrypt function to get the message

c. Eavesdrop and decrypt the message (Eve)

from all the public information, calculate the original message.

In this program, using Elgamal\_decipher(self, c1,c2) function.

2. Use of RSA encryption to

a. Encrypt and send a message (Alice)

1. Alice pick 2 random prime numbers and generate the key pair (public key and private key), already done in the constructor in this program

2. Send the public key to Bob

b. Receive and decrypt a message (Bob)

3. Bob decrypted the key using the private key which Eve doesn’t know(RSA\_decrypt function)

c. Eavesdrop and decrypt the message (Eve)

Eve try to get the original message from the cipher text and the public key, using Decipher(self,c,e), but this cracker could just be used to calculate 2 primes in relatively small condition, if 2 primes are too large, this algorithm will run a quite long time which is not possible. Thus, RSA is hard to crack.

3. All functions

CS789Final.py

extended\_gcd(a,b):

Takes 2 variables as input, finally calculate their greatest common devisor and Bézout's identity.

Inverse(a,m):

Takes 2 variables as input, where a is the input variable and m is the modulo, r Return the multiplicative inverse of a mod m.

breakdown(N):

Takes number N as input, this is a helper function for MillerRabin Algorithm, help to find n-1 = (2^r)\*d

MillerRabin(N,K):

Takes a test number N and numbers of iteration K as input, return if N is a prime.

random\_prime(n)

Takes a common interger n as input, return a random prime between(2,2^n)

gcd(a,b)

Takes 2 intergers as input, calculate and return the greatest common divisor of a and b:

primitiveRoots(p):

Takes a prime as input, first find all primitive root of p, then randomly pick one from the list and return.

ElGamalsteps.py:

Contains a class object Elgamal

Several variables stored: Elgamal.g--generator, Elgamal.y--public key, Elgamal.x--private key, Elgamal.p—modulo, Elgamal.Limit\_searching\_time—time limit for the cracker

\_\_init\_\_(self,g,x,p): constructor

encrypt(self,msg):

Takes String message as input, encrypt the message and return en\_msg and ciphertext c for decrypt

decrypt(self, en\_msg, c):

Takes encrypted message and ciphertext as input, decrypt and return the original message.

Elgamal\_decipher(self, c1,c2):

Takes 2 ciphertext as input, simulate Eve’s condition to crack Elgamal encryption to gain the original message.

RSAsteps.py:

Contains a class object rsa

Several variables stored: rsa.p—prime p, rsa.q—prime q, rsa.n—product of p and q, rsa.t—phi(pq), rsa.Limit\_searching\_time—time limit for the cracker

\_\_init\_\_(self,p,q): constructor

gcd(self,a,b):

find the greatest common divisor of a and b.

public\_key(self):

calculate the public key

private\_key(self):

calculate the private key based on public key and the totient

RSA\_encrypt(self,msg):

Take interger message as input, return the encrypted message

RSA\_decrypt(self,en\_msg):

Take encrypted message as input, decrypt the en\_message

Decipher(self,c,e):

Take cipher text and the public key as input, try to simulate Eve’s condition to get the original message.

UnitTest.py:

The union of tests of these functions. Include most of the edge cases and simple cases.