1. Use of ElGamal encryption to

a. Encrypt and send a message (Alice)

b. Receive and decrypt a message (Bob)

c. Eavesdrop and decrypt the message (Eve)

2. Use of RSA encryption to

a. Encrypt and send a message (Alice)

b. Receive and decrypt a message (Bob)

c. Eavesdrop and decrypt the message (Eve)

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.