NTRU Encrypt Implementation

**Polynomial Package**

------------------

\* poly.py : Polynomial Library for polynomials with rational coefficients

\* testPoly.py : A test for functions defined in our library

**Ntru Package**

------------

\* ntru.py : Ntru Encryption Class Implementation for Python

\* example\_bobalice.py: An example using Ntru given in the Ntru Documentation NTRU.md

In the project we have analysed the working and implementation of NTRU encrypt algorithm. The pseudo code of which has already been submitted. The actual implementation has 2 basic parts one is the driver file “ntru.py” and the support files “poly.py” and “testpoly.py”

**Documentation for Polynomial package:**

Each polynomial is represented by a Python list of coefficients for example:

\* c1=[1,2,3,1] represents the polynomial 1+2x+3x^2+x^3

\* c2=[fraction(1,2),fraction(2,3)] represents the polynomial (1/2)+(2/3)x

Operations/Functions

--------------------

1) addPoly(c1,c2) :------------------------------------ Returns Addition of polynomials c1 and c2

2) subPoly(c1,c2) :------------------------------------ Returns Subtraction of c2 from c1

3) multPoly(c1,c2) :------------------------------------ Returns Product of c1 and c2

4) divPoly(c1,c2) :------------------------------------ Returns the Quotient and Division of c1 / c2

5) cenPoly(c1,q) :------------------------------------ Returns the centered lift of the given polynomial

6) resize(c1,c2) :------------------------------------ Adds leading zeros to the smaller of the two vectors which represent polynomials.

7) trim(c1) :------------------------------------ Removes Leading zeros from the input vector representing the polynomial

8) modPoly(c1,k) :------------------------------------ Takes the k-modulo of the polynomial c1 by taking the modulo of each coefficient of c1.

9) isTernary(f,alpha,beta):------------------------------------ Checks if the polynomial is a Ternary polynomial returns a Boolean value

10) extEuclidPoly(a,b) :------------------------------------- Returns [gcd,s,t] where s and t are Bezout Polynomials

\* All functions below work with fraction coefficients (refer to Example 1.2)

\* All functions return trimmed output

**Documentation for NTRU package**

Class Variables

Parameters:

N : A prime number (Strict upper bound on the polynomials)

p : An integer parameter

q : An integer parameter

d : An integer parameter

Polynomials:

f : A Ternary polynomial T(d,d+1)

g : A Ternary polynomial T(d,d)

h : Public Key (Can be generated by the Key Maker(Bob)) or Set by the sender (Alice)

f\_p : Bezout polynomial s (mod p)

f\_q : Bezout polynomial s (mod q)

D : Set to (X^N-1)

Class Methods:

--------------------------------

1) Function: genPublicKey(f,g,d)

Input : Polynomials f,g,d

Output : None

Description : Generates Public Key and sets h variable to equal it

-------------------------------------

2) Function: setPublicKey(public\_key)

Input : Polynomial public\_key

Output : None

Description : Sets class variable h (public key) to the given custom public\_key

--------------------------

3) Function getPublicKey()

Input : None

Output : h

Description : Getter function for class variable h (public key)

--------------------------

4) Function encrypt(m,randPol)

Input : Two polynomials as coefficient vectors m and randPol

Output : encrypted message as polynomial e

Description : Encrypts given message m and a random polynomial randPol. Note that before calling this function you need to either set the public key h or generate it.

--------------------------

5) Function decrypt(en)

Input : Encrypted message en represented as a polynomial coefficients vector

Output : Decrypted message m

Description : This method decrypts the given message using private key information stored during the generation of the public key. Therefore can only be used one the public key has been generated.

**Implementation:**

The basic working and mathematics behind the NTRU Encrypt system had already been explained in the Psuedo code submitted earlier.

The driver file “ntru.py” calls/uses the other secondary files and helper functions in order to achieve encryption based on the parameters acquired in the example file namely “example\_bobalice.py”

Example:

An Example: (Provided in file bobalice.py)

------------------------------------------

1) Bob is expecting to receive some secure information from Alice

2) Bob creates an instance of NTRU of with parameters (N=7,p=29,q=491531) which are publically available

>>> Bob=Ntru(7,29,491531)

3) Next he generates a public\_key by specifying a function f,g and parameter d

>>> f = [1,1,-1,0,-1,1]

>>> g = [-1,0,1,1,0,0,-1]

>>> d=2

>>> Bob.genPublicKey(f,g,2)

>>> pub\_key=Bob.getPublicKey()

4) Alice wants to send a secure message to Bob. She sets up another instance of Ntru using parameters and public key Bob provided

>>> Alice=Ntru(7,29,491531)

>>> Alice.setPublicKey(pub\_key)

5) She encrypts her message using her instance and a random Ternary polynomial for noise

>>> msg=[1,0,1]

>>> ranPol=[-1,-1,1,1]

>>> encrypt\_msg=Alice.encrypt(msg,[-1,-1,1,1])

6) Finally Bob decrypts message sent to him

>>> print Bob.decrypt(encrypt\_msg)