CSE4003 Cyber Security

Project

FALL SEMESTER 22-23

SLOT – C1+TC1

Project Report NTRU Cryptosystem

Submitted by:

S.no	Name	Reg.no
1	Somya Rathi	20BCE2323
2	Aryan Agrawal	20BCE2876
3	Rajat Bargoti	20BCE2880

Professor
ILAYARAJA V

VELLORE INSTITUTE OF TECHNOLOGY
VELLORE – 632 014
TAMIL NADU
INDIA

Abstract: NTRU is one of the few public key cryptosystems which is supposedly quantum- computer resistant. Its security is based on the presumed difficulty of a lattice problem, namely, the shortest vector problem. This project describes the NTRU cryptosystem and its cryptanalysis. The project describes how NTRU is a fast-implementing algorithm and can be used as an alternative in the android encryption text. Finally, a comparison of the performance of the NTRU with another public key cryptosystem RSA is presented which describes the fast implementation of the NTRU algorithm as compared to various other cryptosystems. The project consists of a full and deep research and analysis on NTRU algorithm.

Keywords:
1. NTRU **2.** RSA **3.** ECC **4.** O(N) **5.** NTRUEncrypt

INDEX

1. INTRODUCTION

Quantum computers are well known to be able to break cryptosystems that are based on the integer factorization problem or some discrete logarithm problem in polynomial time. This affects RSA and ECC and also number field cryptosystems. The NTRU cryptosystem is an alternative algorithm based on a completely different mathematical problem from RSA and ECC called the closest lattice vector problem. As a result, NTRU is very fast and resistant to quantum computers. NTRU cryptosystem - nth degree truncated polynomial ring unit, relies on the presumed difficulty of factoring certain polynomials in a truncated polynomial ring into a quotient of two polynomials having very small coefficients. In comparison to widely- known systems such as RSA 2 or ECC 3, the main advantage of NTRU is that its time complexity quadratic order O(N2) in worst case. This system is considered as the fastest cryptosystem, and its strong points are short key size and the speed of encryption and decryption. Since both encryption and decryption use only simple polynomial multiplication, these operations are very fast compared to other asymmetric encryption schemes, such as RSA, Elgamal and elliptic curve cryptography. The security of NTRU is based on intractability of hard problems in certain type of lattices, called convolutional modular lattices, therefore, it is also categorized as a lattice-based cryptosystem.

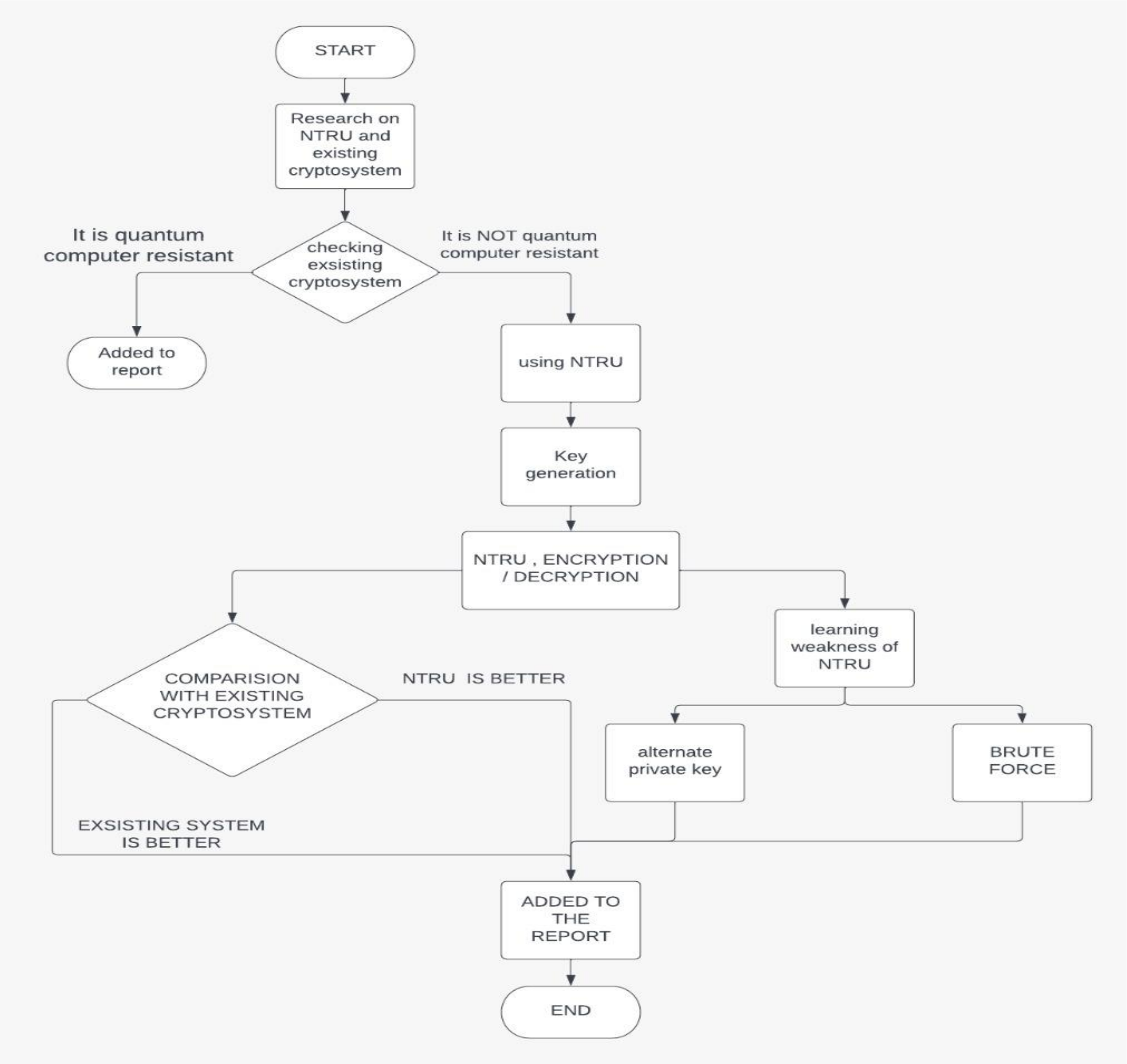
2. PROBLEM DISCUSSION

A tremendous utility in the age of big data is the collection and sharing of data for the common good. Specifically, users share their information to central repositories and this allows other users and third parties to use this shared information for analysis. While there is a great deal of benefit to shared data, a major issue is the protection of the user's privacy while still allowing other users the benefit of performing analysis.

3. SOLUTION

Research by Shieh, Lin, and Wu proposes the creation of a recommendation system with an end-to-end encrypted domain. In fact, to make such an encryption scheme useful requires ring homomorphisms for the encryption, because to effectively perform calculations in our encrypted domain two operations are needed: multiplication and addition. Specifically, the encryption scheme is based on lattice-based polynomial ring homomorphism and is a special case of the NTRU encryption scheme. This NTRU based method is an improvement on RSA type encryption methods, which only use one operation, and as an added benefit, NTRU appears to be resistant to quantum computer based in spite of the obvious advantages of NTRU encryption, there is a lack of freely available software packages for the implementation of lattice-based encryption. Our project is the creation of a python library of functions which implement the algorithms necessary for an NTRU encryption of polynomials.

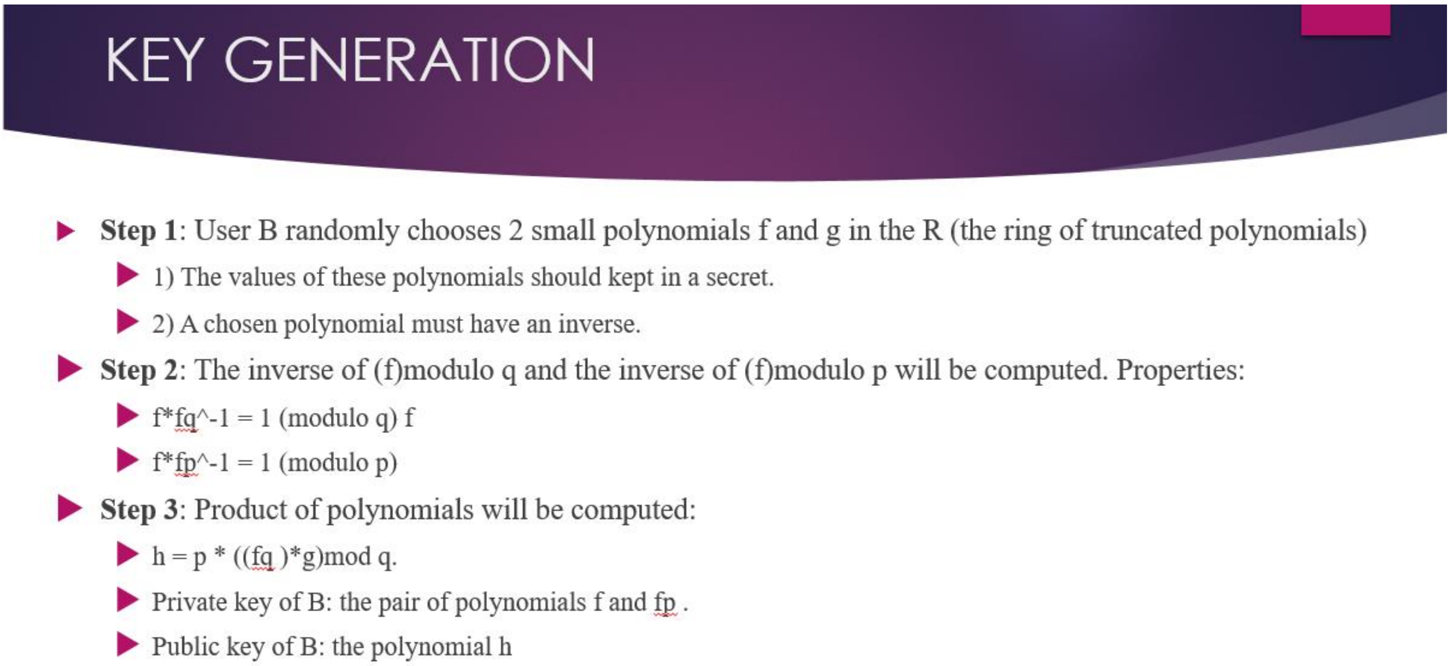
Work Flow Diagram


START
Research on NTRU and existing cryptosystem
checking exsisting cryptosystem
It is quantum computer resistant
It is NOT quantum computer resistant
Added to report
using NTRU
Key generation
NTRU , ENCRYPTION / DECRYPTION
COMPARISION WITH EXISTING CRYPTOSYSTEM
NTRU IS BETTER
EXSISTING SYSTEM IS BETTER
learning weakness of NTRU
alternate private key
BRUTE FORCE
ADDED TO THE REPORT
END

HARDWARE / SOFTWARE REQUIRED

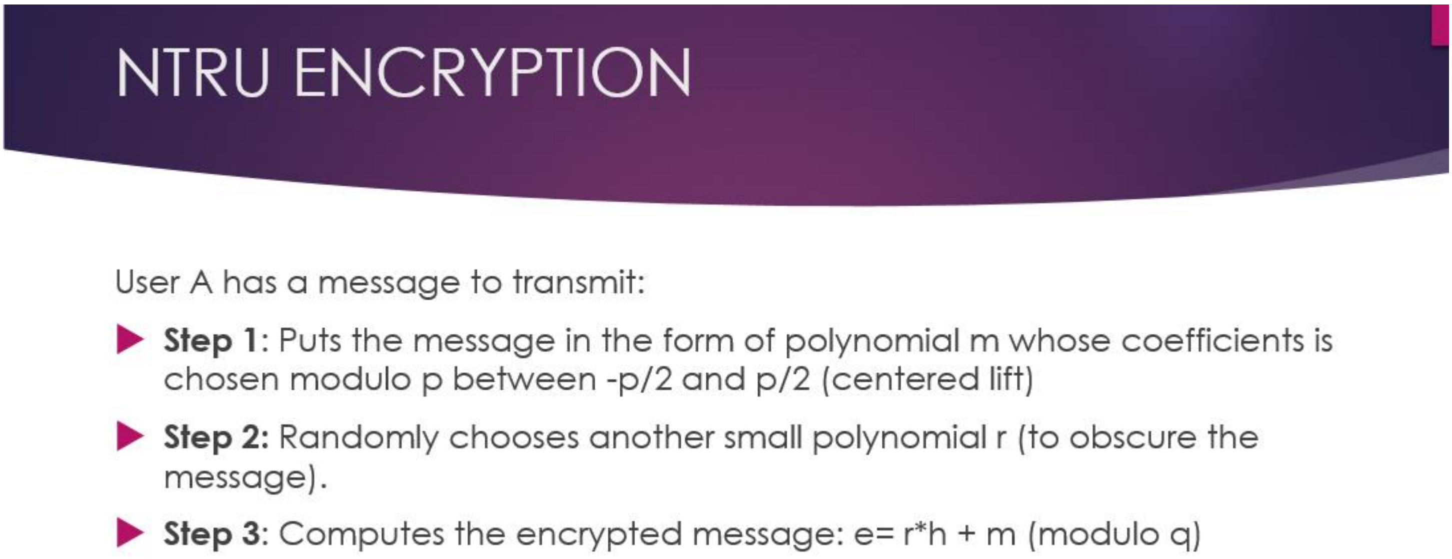
- ▶ 4 GB RAM, Window 10 configuration
- ▶ PYTHON 3.9
- ▶ IDLE
- ▶ MODULES NEEDED: 1)NTRU 2)OPERATOR 3)NUMPY 4)FUNCTIONS

MODULE DESCRIPTION



KEY GENERATION

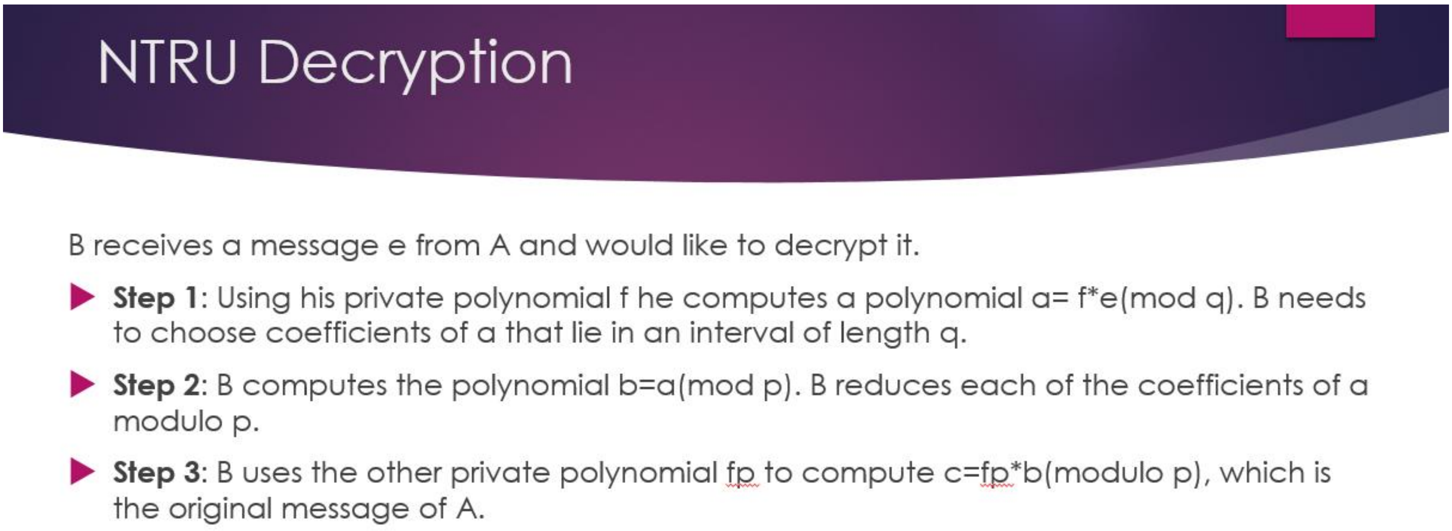
- **Step 1**: User B randomly chooses 2 small polynomials f and g in the R (the ring of truncated polynomials)
  - 1) The values of these polynomials should kept in a secret.
  - 2) A chosen polynomial must have an inverse.
- **Step 2**: The inverse of (f)modulo q and the inverse of (f)modulo p will be computed. Properties:
  - f*fq^-1 = 1 (modulo q) f
  - f*fp^-1 = 1 (modulo p)
- **Step 3**: Product of polynomials will be computed:
  - h = p * ((fq)*g)mod q.
  - Private key of B: the pair of polynomials f and fp .
  - Public key of B: the polynomial h



NTRU ENCRYPTION

User A has a message to transmit:

- **Step 1**: Puts the message in the form of polynomial m whose coefficients is chosen modulo p between -p/2 and p/2 (centered lift)
- **Step 2:** Randomly chooses another small polynomial r (to obscure the message).
- **Step 3**: Computes the encrypted message: e= r*h + m (modulo q)



NTRU Decryption

B receives a message e from A and would like to decrypt it.

- **Step 1**: Using his private polynomial f he computes a polynomial a= f*e(mod q). B needs to choose coefficients of a that lie in an interval of length q.
- **Step 2**: B computes the polynomial b=a(mod p). B reduces each of the coefficients of a modulo p.
- **Step 3**: B uses the other private polynomial fp to compute c=fp*b(modulo p), which is the original message of A.

4. LITERATURE SURVEY

[1] Quantum Computing: The Risk to Existing Encryption Methods - Zach Kirsch

Quantum computing is a new way of computing. In a conventional computer the quintessential information particle, the bit, can only exist in two states, 0 or 1. A quantum computer benefits from the ability of subatomic particles to exist in more than one state simultaneously. In this case, quantum bits (or QuBits for short) can store much more information because they make direct use of quantum mechanics properties, such as superposition and entanglement. Essentially, while bits can only be 0 or a 1, QuBits can assume any superposition of these values. This means computational operations can be performed at a much higher speed and with much less power consumption.

[2] Factorization Problem -Christof Paar, Jan Pelzl

Factorization Problem - RSA Cryptosystem: RSA and in general asymmetric algorithms are not meant to replace symmetric algorithms because they are computationally costly. RSA is mainly used for secure key exchange between end nodes and often used together with symmetric algorithms such as AES, where the symmetric algorithm does the actual data encryption and decryption. Kirsch stated that RSA is theoretically vulnerable if a fast-factorizing algorithm is introduced or huge increase in computation power can exist. The latter can be achieved with the use of quantum mechanics on computers, known as quantum computers.

[3] Introduction to Public-Key Cryptography - Christof Paar, Jan Pelzl

[4] Research on NTRU Algorithm for Mobile Java Security-Xiaoyu Shen, Shenjun Du, Rong Chen

[5] Existing public-key cryptography is based on the difficulty of factoring and the difficulty of calculating elliptic curve discrete logarithms. Because those two problems will be readily and efficiently solved by a sufficiently large-scale quantum computer, there are already many studies directed to post-quantum cryptography, such as lattice-based cryptography, multivariate cryptography or hash-based cryptography, all of which are strong candidates for securing our data in a post-quantum world.

[6] Traditionally, it has been difficult to deploy mobile Java security in mass market mobile devices primarily due to size and speed constraints. NTRU, a relatively new public key cryptosystem based on the shortest vector problem in a lattice, with many advantages compared with other cryptosystems, such as high speed and low memory use, provides the possibility to overcome the constraints. This paper briefly describes NTRU cryptosystem and two approaches to optimize the algorithm, such as changing forms and using low hamming weight products; the former approach simplifies both the key generation and the decryption, and the latter one increases the speed of the convolution multiplication by nearly 2 times. Experiments on the performance of enhanced NTRU-251 compared with RSA-1024 in the mobile Java device are made. Preliminary experimental results show the advantages of NTRU over RSA, such as, at the similar security level, the key size of NTRU is less than a quarter of that of RSA, and the speed of NTRU is much faster than that of RSA; the key generation is more than 200 times faster, the encryption is almost 3 times faster, and the decryption is about 30 times faster. These experimental results show the applicable prospect of NTRU in mobile Java systems.

[7] Short Message Service (SMS) is getting more popular now-a-days. It will play a very important role in the future business areas of mobile commerce (M- Commerce). Presently many business organizations are using SMS for their business purposes. SMS's security has become a major concern for both business organizations and customers. There is a need for an end-to-end SMS Encryption in order to provide a secure medium

for communication. Security is main concern for any business company such as banks who will provide these mobile banking services. Till now there is no such scheme that provides complete SMSs security. The transmission of an SMS in GSM network is not secure at all. Therefore, it is desirable to provide SMS security for business purposes. In this paper, we have analyzed Number Theory Research Unit (NTRU) Crypto algorithm and NTRU Sign (NTRU Signature) algorithm. We have compared theoretically the performance metrics like key size, key generation time, encryption time, decryption time, CPU computational power, speed, efficiency, memory space and security strength between NTRU and RSA. This theoretical result encouraged us to simulate the NTRU cryptosystem and NTRU Sign algorithm on mobile phones using full size of SMS as future work.

[8] We describe NTRU, a new public key cryptosystem. NTRU features reasonably short, easily created keys, high speed, and low memory requirements. NTRU encryption and decryption use a mixing system suggested by polynomial algebra combined with a clustering principle based on elementary probability theory.The security of the NTRU cryptosystem comes from the interaction of the polynomial mixing system with the independence of reduction modulo two relatively prime integers' p and q.

[9] The users of Cloud Computing keep their facts onto the third-party owners and experience the online demand programs, services and garage from a shared pool of configurable computing resources, without the load of facts garage and protection and costs. The furnished security must guarantee no longer best on authentication but also on files over the cloud for the outsourced records, that is now maintained by way of third parties consisting of cloud carriers. Unluckily, the infrastructure of cloud computing is constructed on internet, and will stumble upon all of the risk of net. To lower the risks, an authentication mechanism is the viable answer. However, conventional alphanumeric password authentication mechanism is not always relaxed sufficient. A cozy authentication mechanism, the use of totally public-key encryption-based password is proposed in this paper for enhancing traditional authentication mechanism and let users get entry to cloud services securely. The Mechanism of generating new public key using NTRU algorithm (nth degree truncated polynomial) for the purpose of security and it gets encrypted according to the parameters. By means of doing this, no unauthorized users can recognize the credentials. Hence the software is stored from numerous community threats e.g., hacking.

NTRU is a patented and open supply public key cryptosystem that makes use of lattice-based cryptography completely to encrypt and decrypt statistics. It includes algorithms: NTRUEncrypt, which is used for encryption, and NTRUSign, that is used for digital signatures. The NTRU Encrypt is a public-key cryptosystem which is based on the shortest vector hassle. Its most important characteristics are the low-key reminiscence and computational requirements as presenting a high protection stage.

[10] At equivalent cryptographic strength, NTRU performs costly private key operations much faster than RSA does. The time of performing an RSA private operation increases as the cube of the key size, whereas that of an NTRU operation increases quadratically. According to the Department of Electrical Engineering, University of Leuven, a modern GTX280 GPU, a throughput of up to 200 000 encryptions per second can be reached at a security level of 256 bits. Comparing this to a symmetric cipher (not a very common comparison), this is only around 20 times slower than a recent AES implementation. Unlike RSA and Elliptic Curve Cryptography, NTRU is not known to be vulnerable to quantum computer-based attacks. The National Institute of Standards and Technology wrote in a 2009 survey that are viable alternatives for both public key encryption and signatures thatare not vulnerable to Shor’s Algorithm and of the various lattice based cryptographic schemes that have been developed, the NTRU family of cryptographic algorithms appears to be the most practical".

[11] NTRU implements the NTRUEncrypt public key encryption algorithm in Java and C NTRUEncrypt is lattice-based and not known to be breakable even with quantum computers. Commonly used cryptosystems like RSA or ECC, on the other hand, will be broken if and when quantum computers become available. In addition, NTRU is significantly faster than other public-key cryptosystems. The chart below compares the C implementation (libntru) against other cryptosystems and against the NTRU reference implementation.

5. METHODOLOGY

5.1 PARAMETERS

In NTRU cryptosystem, operations are based on objects in a truncated polynomial ring

$R=Z[X]/(X^N-1)$, polynomial degree at most N-1:
$a0 + a1x + a2x^2 + \cdot \cdot \cdot + aN-1x^(N-1)$
Following is the list of keys and parameters used in NTRU: N – the polynomials in the ring R have degree N-1. (Non-secret) q – the large modulus to which each coefficient is reduced. (Non secret) p – the small modulus to which each coefficient is reduced. (Non-secret) f- a polynomial that is the private key. g - a polynomial that is used to generate the public key h from f (Secret but discarded after initial use) h - the public key, also a polynomial r - the random “blinding” polynomial (Secret but discarded after initial use)
d – coefficient

5.2 KEY GENERATION

1st step:
User B randomly chooses 2 small polynomials f and g in the R (the ring of truncated polynomials).
Note:

1) The values of these polynomials should kept in a secret.
2) A chosen polynomial must have an inverse.

2nd step:
The inverse of f modulo q and the inverse of f modulo p will be computed.
Properties:
$f*fq^-1 = 1$ (modulo
q) $f*fp^-1 = 1$
(modulo p) *3rd step:*
Product of polynomials will be computed: $h = p * ((Fq)*g) \bmod q$.
Private key of B: the pair of polynomials f and fp .
Public key of B: the polynomial h.

5.3 NTRU Encryption:

User A has a message to transmit:
1st step:
Puts the message in the form of polynomial m whose coefficients is chosen modulo p between -p/2 and p/2 (centered lift) *2nd step:* Randomly chooses another small polynomial r (to obscure the message).
3rd step:
Computes the encrypted message: $e= r*h + m$ (modulo q)

5.4 NTRU Decryption:

B receives a message e from A and would like to decrypt it.

1st step:

Using his private polynomial f he computes a polynomial a= f*e(mod q). B needs to choose coefficients of a that lie in an interval of length q.

2nd step:

B computes the polynomial b=a(mod p). B reduces each of the coefficients of a modulo p.

3rd step:

B uses the other private polynomial fp to compute c=fp*b(modulo p), which is the original message of A.

In spite of the obvious advantages of NTRU encryption, there is a lack of freely available software packages for the implementation of lattice-based encryption. Our project is the creation of a python library of functions which implement the algorithms necessary for an NTRU encryption of polynomials as described below We have built a python package which provides an implementation of NTRU Encryption System. To ensure accuracy of the encryption and decryption we required to know polynomials with high accuracy. Since existing polynomial libraries such as the one provided by the Numpy Python Package has coefficients as floats, we could not employ them. Therefore, we implemented a fresh polynomial package which allows us to perform operations on polynomial with rational coefficients.

6. IMPLEMENTATION

6.1

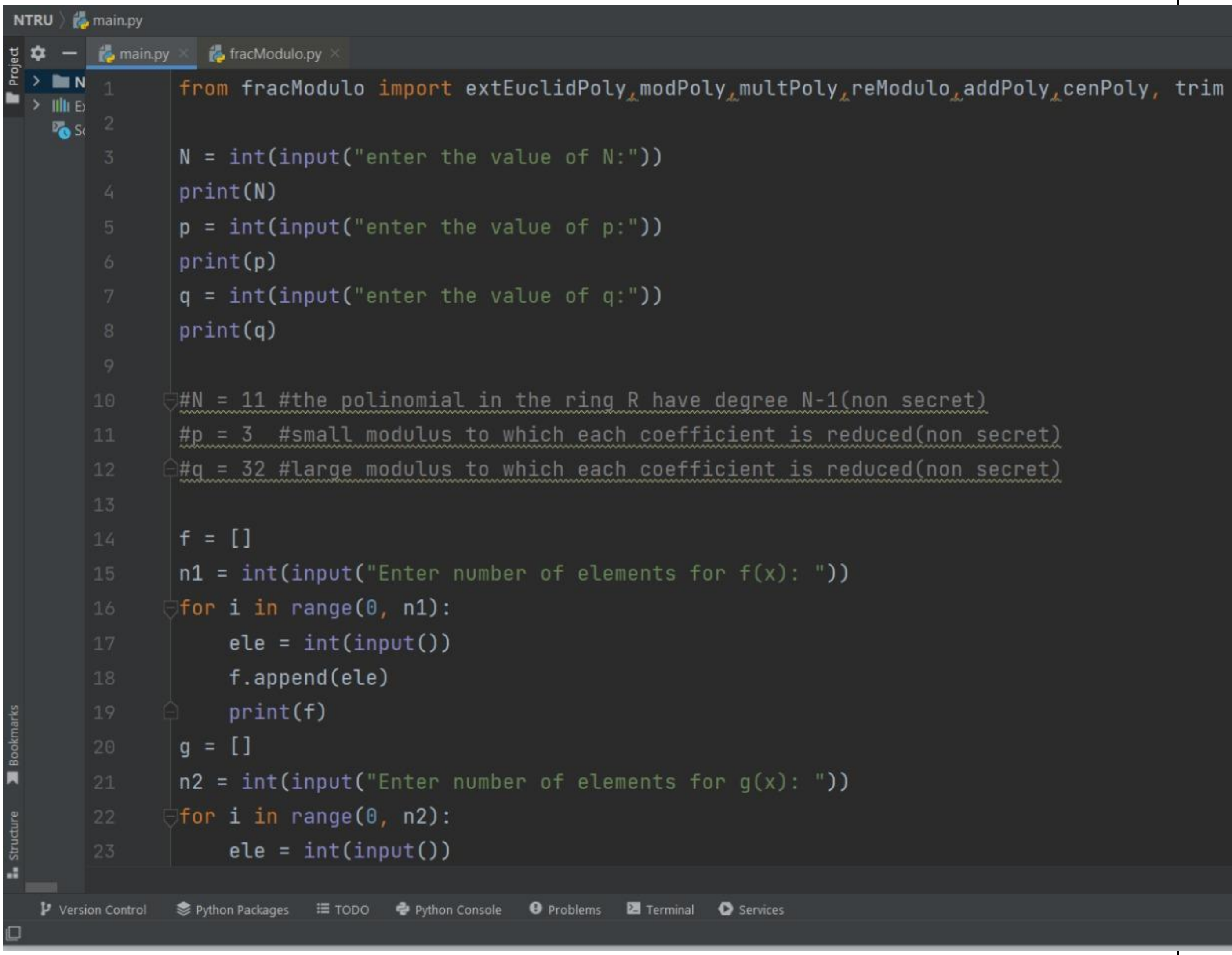
This library allows the user do mathematical operations on rational coefficient polynomials. For rational coefficients we have used fractions data type which is a standard library in Python.

addPoly(c1,c2): Returns addition of polynomials c1 and c2 *subPoly(c1,c2):* Returns subtraction of c2 from c1 *multPoly(c1,c2):* Returns product of c1 and c2 *divPoly(c1,c2):* Returns the quotient and remainder of c1/c2 *cenPoly(c1,q):* Returns the centered lift of the given polynomial *resize(c1,c2):* Adds leading zeros to the smaller of the two vectors which represent polynomials.

trim(c1): Removes Leading zeros from the input vector representing the polynomial *modPoly(c1,k):* Returns a polynomial with the coefficients of c1 modulo an integer k.
isTernary(f,alpha,beta): Checks if the polynomial is a Ternary polynomial returns a Boolean value *extEuclidPoly(a,b)* : Returns [gcd(a,b),s,t] where s and t are Bezout polynomials.
* All functions above work with fraction coefficients

6.2 MAIN.py

Contains definition of Ntru class object *genPublicKey(f,g,d):*
Generates a public key
setPublicKey(public_key): Sets class-variable h (public key) to the given custom public_key
getPublicKey(): getter function for class variable h (public key) *encrypt(m,randPol):* Encrypts given message m using a random polynomial randPol and public key. (Note that before calling this function you need to either set the public key or generate it)
decrypt(en): This method decrypts the given message using private key information stored during the generation of the public key. Therefore, can only be used once the public key has been generated. *decryptSQ(e):* Decrypts messages using a slightly different approach used for analytics in encrypted domain (refer to report)

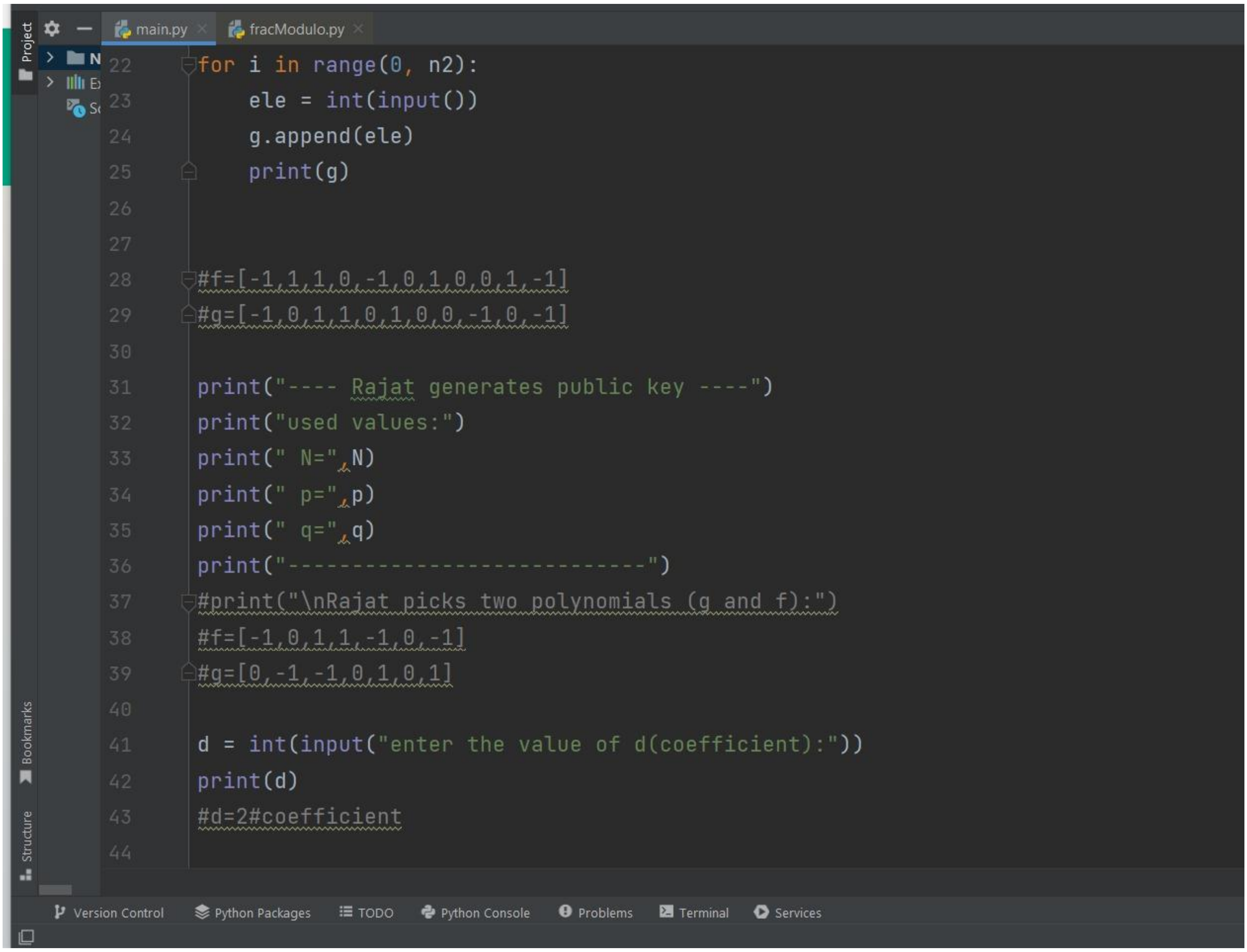


```python
from fracModulo import extEuclidPoly,modPoly,multPoly,reModulo,addPoly,cenPoly, trim

N = int(input("enter the value of N:"))
print(N)
p = int(input("enter the value of p:"))
print(p)
q = int(input("enter the value of q:"))
print(q)

#N = 11 #the polinomial in the ring R have degree N-1(non secret)
#p = 3  #small modulus to which each coefficient is reduced(non secret)
#q = 32 #large modulus to which each coefficient is reduced(non secret)

f = []
n1 = int(input("Enter number of elements for f(x): "))
for i in range(0, n1):
    ele = int(input())
    f.append(ele)
    print(f)
g = []
n2 = int(input("Enter number of elements for g(x): "))
for i in range(0, n2):
    ele = int(input())
```

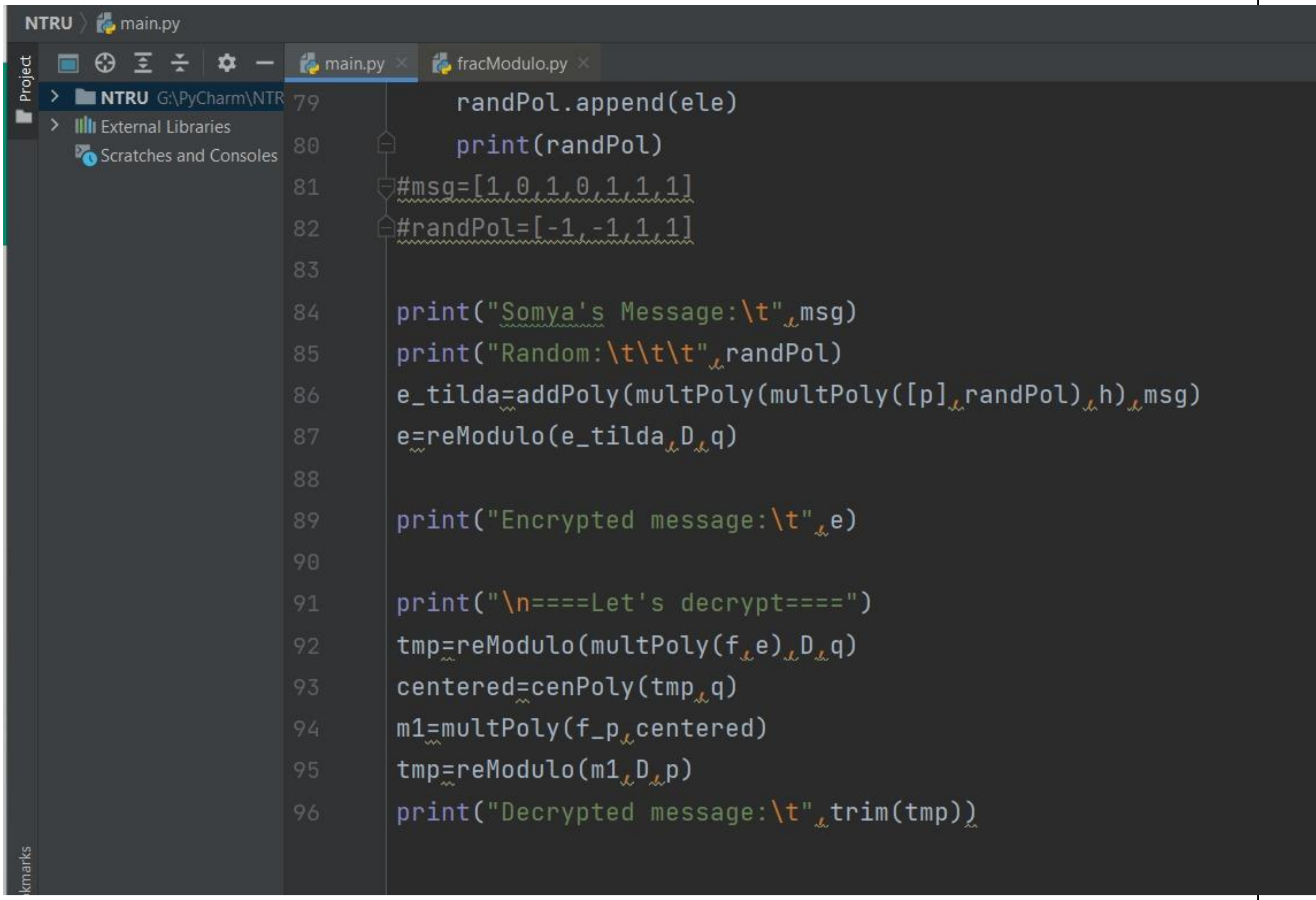


```
for i in range(0, n2):
    ele = int(input())
    g.append(ele)
    print(g)


#f=[-1,1,1,0,-1,0,1,0,0,1,-1]
#g=[-1,0,1,1,0,1,0,0,-1,0,-1]

print("---- Rajat generates public key ----")
print("used values:")
print(" N=",N)
print(" p=",p)
print(" q=",q)
print("----------------------------")
#print("\nRajat picks two polynomials (g and f):")
#f=[-1,0,1,1,-1,0,-1]
#g=[0,-1,-1,0,1,0,1]

d = int(input("enter the value of d(coefficient):"))
print(d)
#d=2#coefficient
```



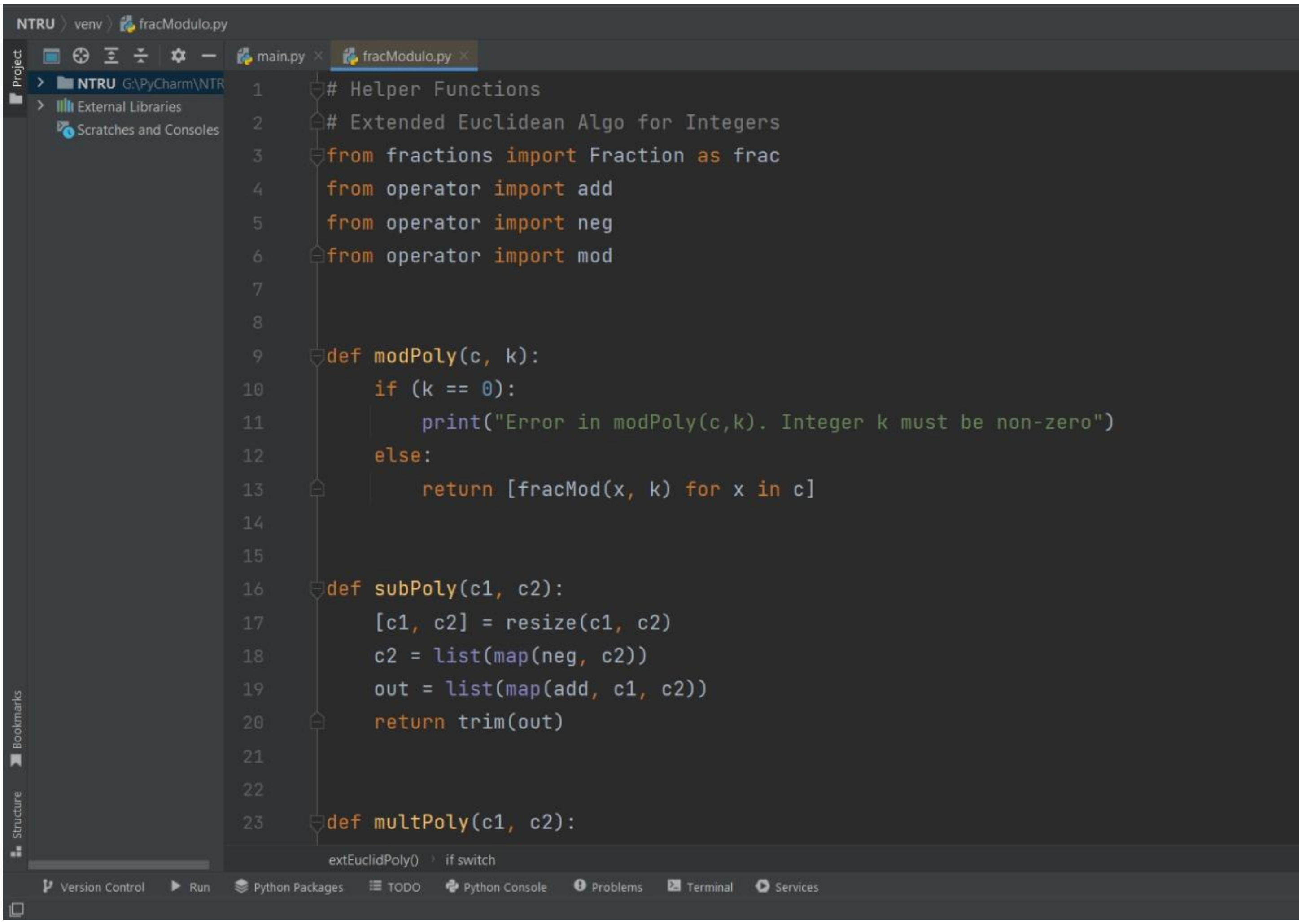
```
    randPol.append(ele)
    print(randPol)
#msg=[1,0,1,0,1,1,1]
#randPol=[-1,-1,1,1]

print("Somya's Message:\t",msg)
print("Random:\t\t\t",randPol)
e_tilda=addPoly(multPoly(multPoly([p],randPol),h),msg)
e=reModulo(e_tilda,D,q)

print("Encrypted message:\t",e)

print("\n====Let's decrypt====")
tmp=reModulo(multPoly(f,e),D,q)
centered=cenPoly(tmp,q)
m1=multPoly(f_p,centered)
tmp=reModulo(m1,D,p)
print("Decrypted message:\t",trim(tmp))
```

6.3 frac modulo



```python
# Helper Functions
# Extended Euclidean Algo for Integers
from fractions import Fraction as frac
from operator import add
from operator import neg
from operator import mod


def modPoly(c, k):
    if (k == 0):
        print("Error in modPoly(c,k). Integer k must be non-zero")
    else:
        return [fracMod(x, k) for x in c]


def subPoly(c1, c2):
    [c1, c2] = resize(c1, c2)
    c2 = list(map(neg, c2))
    out = list(map(add, c1, c2))
    return trim(out)


def multPoly(c1, c2):
```

To test our library as a tool for encrypted domain, we created example Rajatsomya_example.

1) Rajat is expecting to receive some secure information from Alice
2) Rajat creates an instance of NTRU of with parameters (N = 7, p = 29, q = 491531) which he makes publicly available

```
>>> Rajat=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
>>> Rajat.genPublicKey(f, g, 2)
>>> pub_key = Rajat.getPublicKey()
```

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

```
>>> Alice = Ntru(7, 29, 491531)
>>> Alice.setPublicKey(pub_key)
```

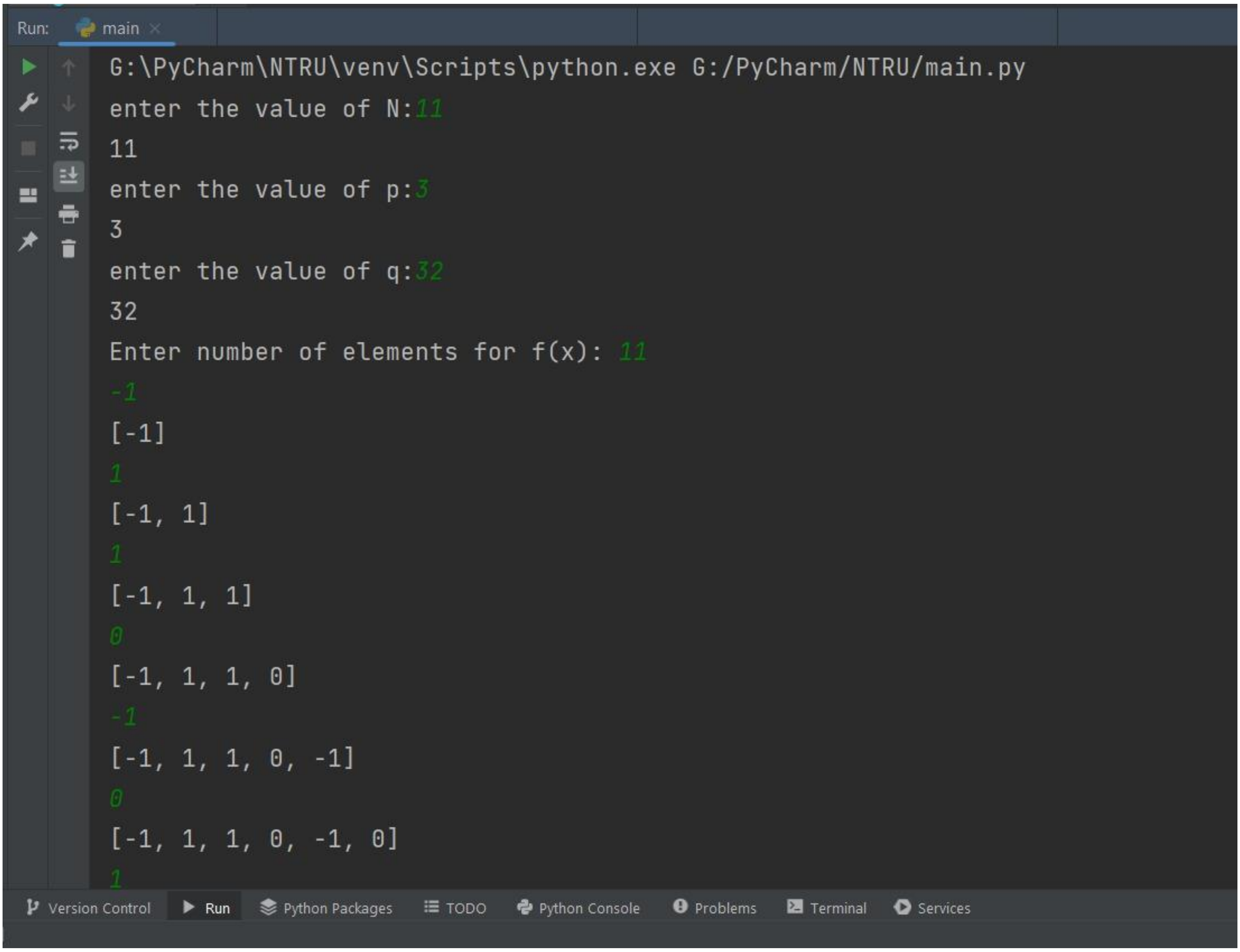
5) She encrypts her message using her instance and a random Ternary polynomial for noise

```
>>> msg = [1, 0, 1]
>>> ranPol = [-1, -1, 1, 1]
>>> encrypt_msg = Alice.encrypt(msg, [-1, -1, 1, 1])
```

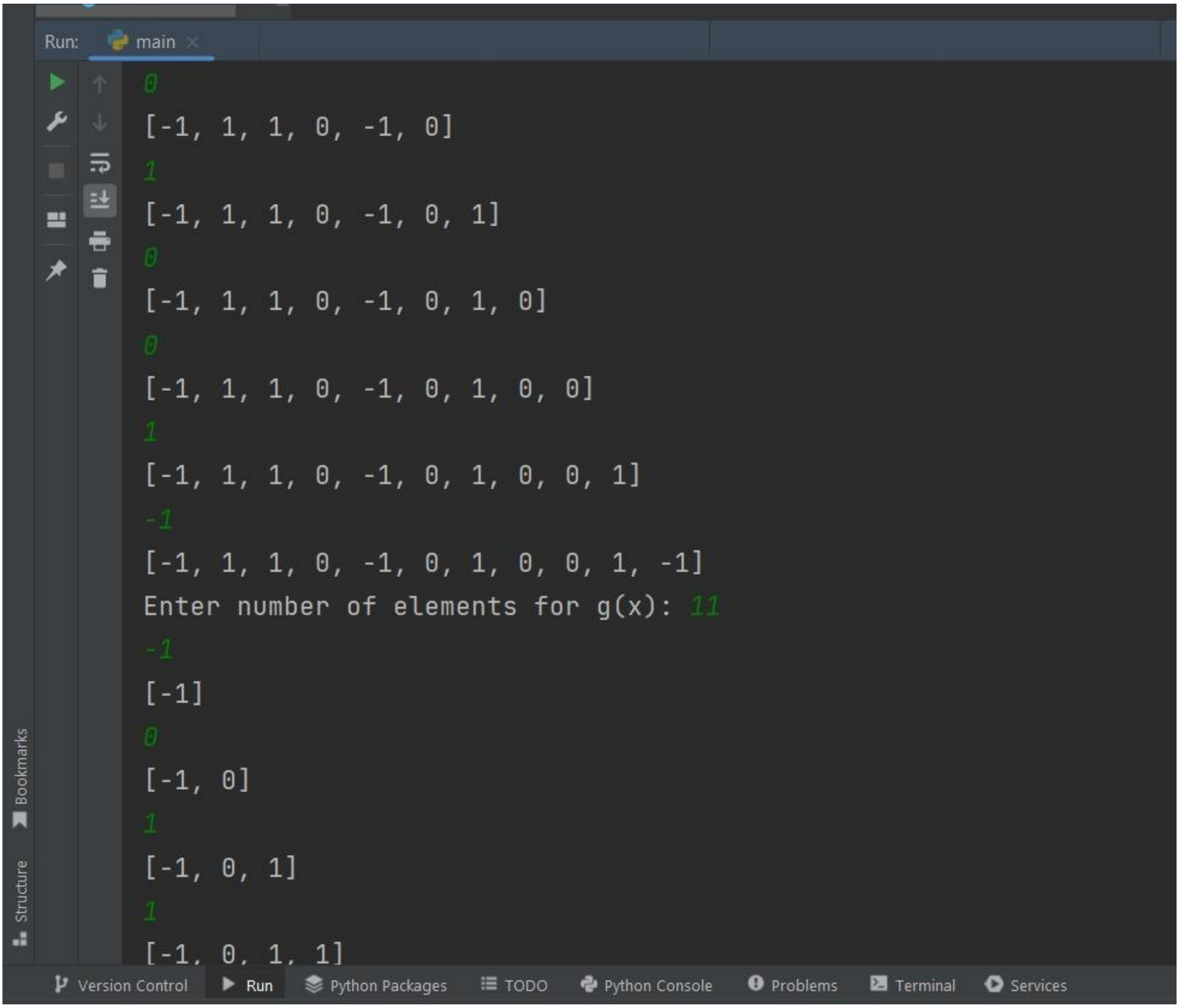
6) Finally, Rajat decrypts message sent to him

```
>>> print Rajat.decrypt(encrypt_msg)
```

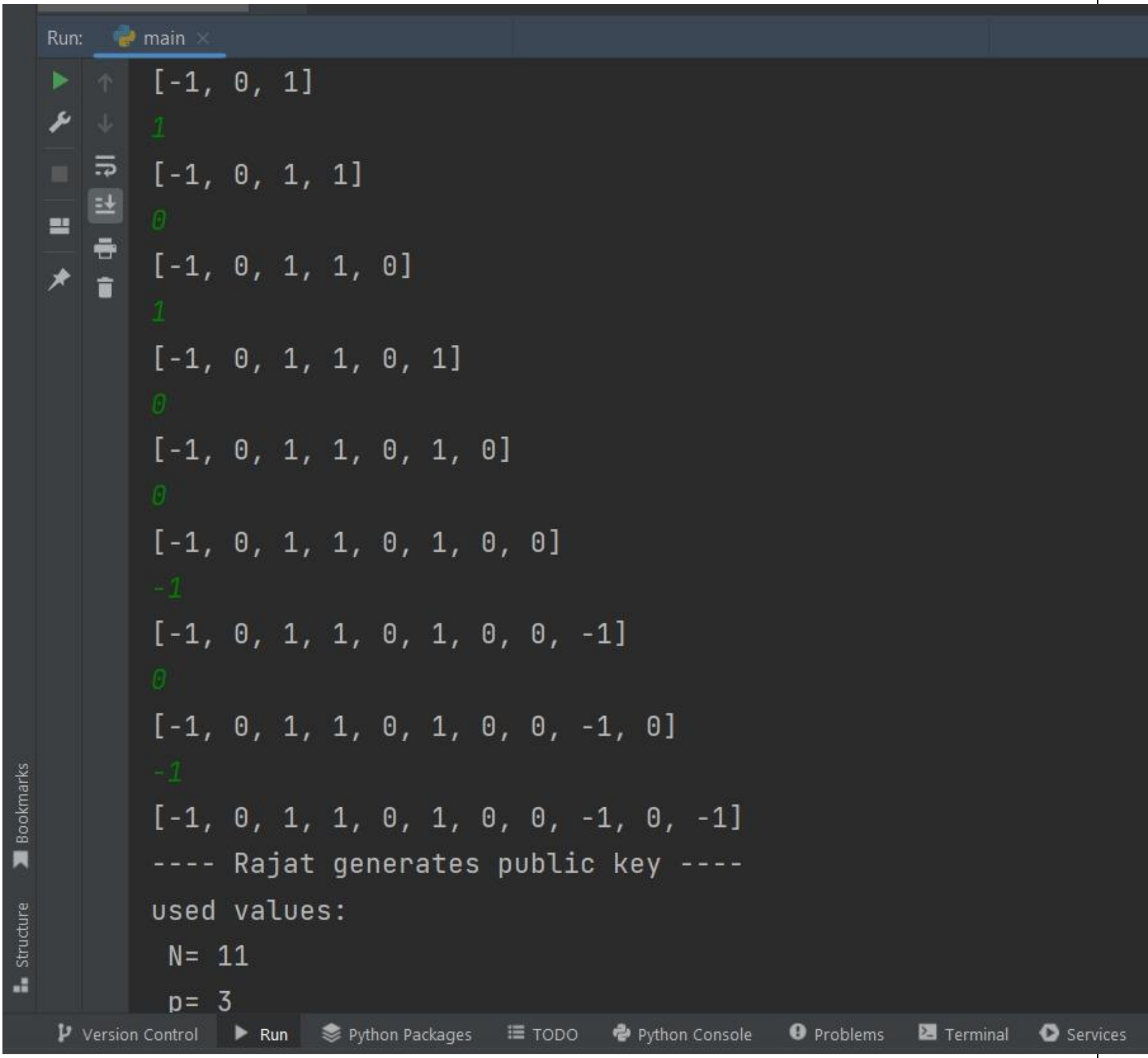
7. RESULT AND DISCUSSION



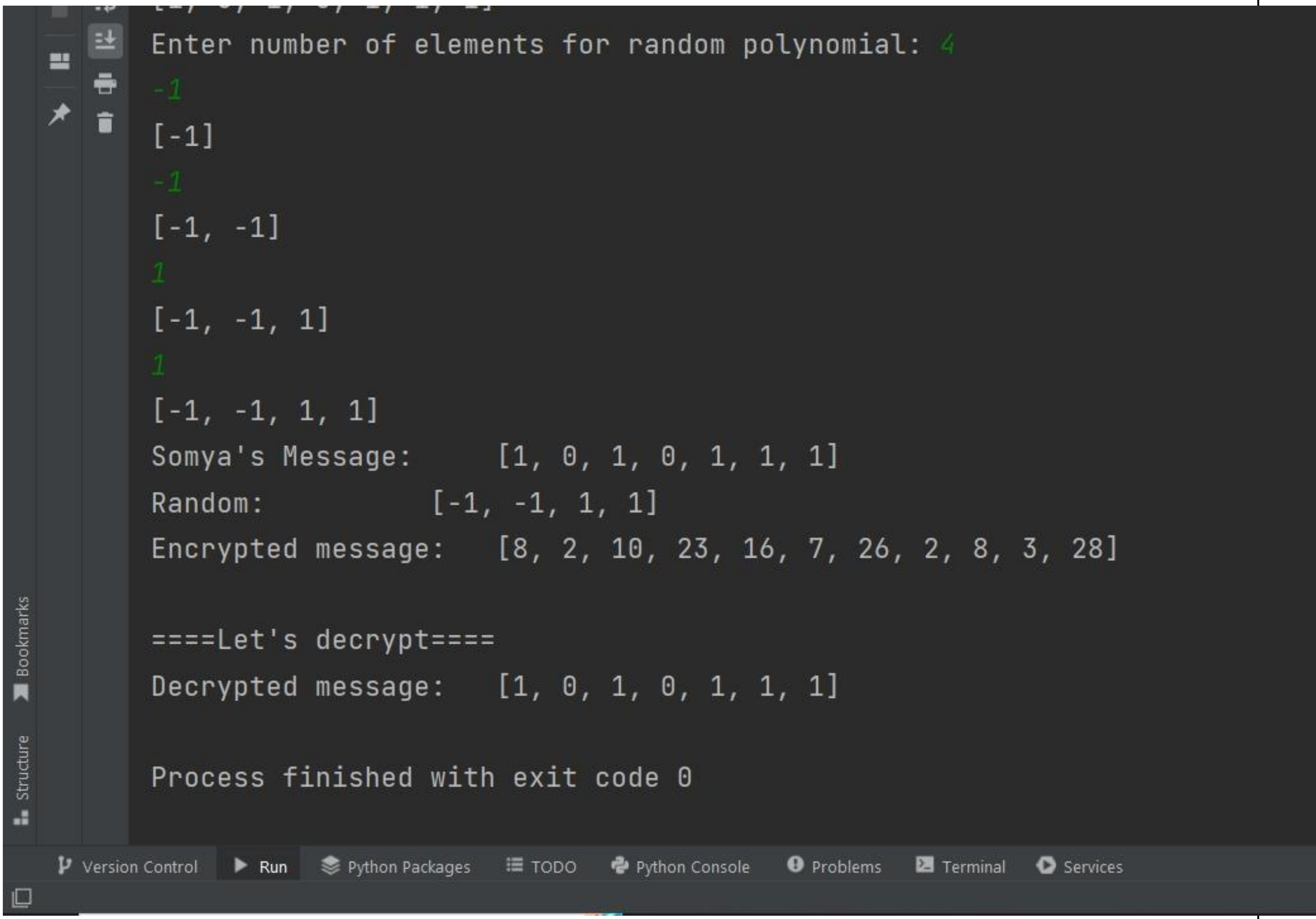
```
G:\PyCharm\NTRU\venv\Scripts\python.exe G:/PyCharm/NTRU/main.py
enter the value of N:11
11
enter the value of p:3
3
enter the value of q:32
32
Enter number of elements for f(x): 11
-1
[-1]
1
[-1, 1]
1
[-1, 1, 1]
0
[-1, 1, 1, 0]
-1
[-1, 1, 1, 0, -1]
0
[-1, 1, 1, 0, -1, 0]
1
```



```
0
[-1, 1, 1, 0, -1, 0]
1
[-1, 1, 1, 0, -1, 0, 1]
0
[-1, 1, 1, 0, -1, 0, 1, 0]
0
[-1, 1, 1, 0, -1, 0, 1, 0, 0]
1
[-1, 1, 1, 0, -1, 0, 1, 0, 0, 1]
-1
[-1, 1, 1, 0, -1, 0, 1, 0, 0, 1, -1]
Enter number of elements for g(x): 11
-1
[-1]
0
[-1, 0]
1
[-1, 0, 1]
1
[-1, 0, 1, 1]
```



```
[-1, 0, 1]
1
[-1, 0, 1, 1]
0
[-1, 0, 1, 1, 0]
1
[-1, 0, 1, 1, 0, 1]
0
[-1, 0, 1, 1, 0, 1, 0]
0
[-1, 0, 1, 1, 0, 1, 0, 0]
-1
[-1, 0, 1, 1, 0, 1, 0, 0, -1]
0
[-1, 0, 1, 1, 0, 1, 0, 0, -1, 0]
-1
[-1, 0, 1, 1, 0, 1, 0, 0, -1, 0, -1]
---- Rajat generates public key ----
used values:
 N= 11
 p= 3
```

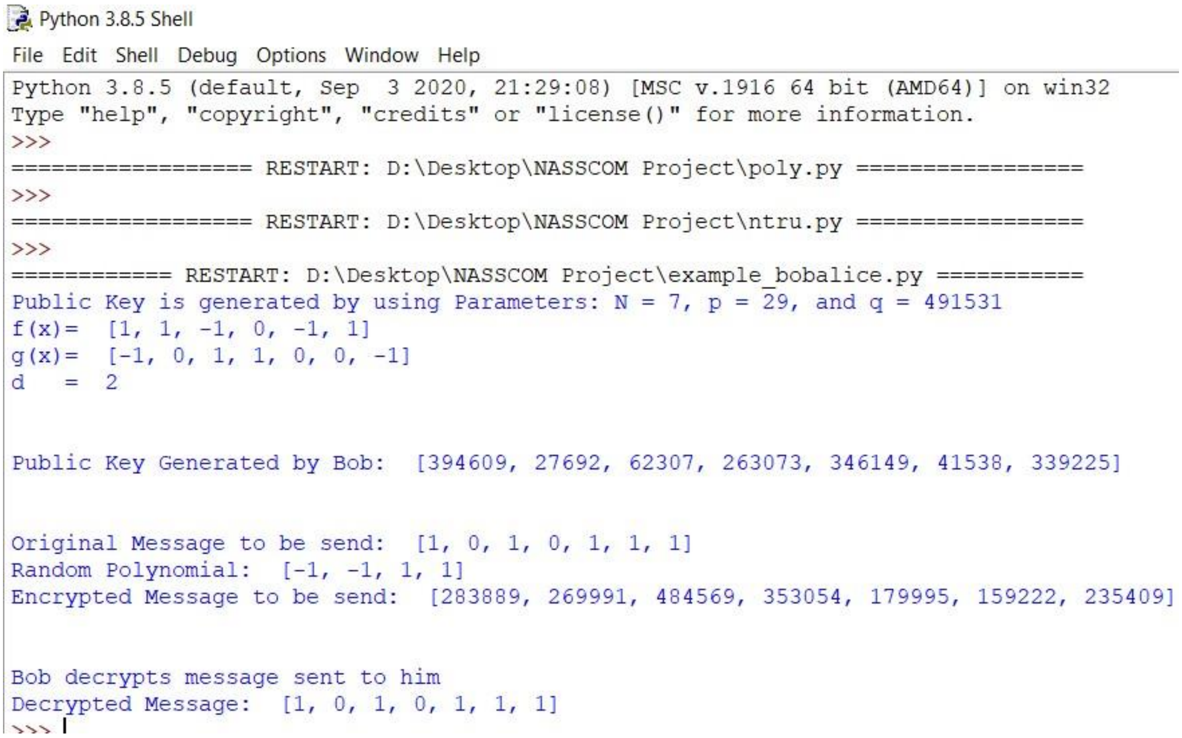


```
Enter number of elements for random polynomial: 4
-1
[-1]
-1
[-1, -1]
1
[-1, -1, 1]
1
[-1, -1, 1, 1]
Somya's Message:     [1, 0, 1, 0, 1, 1, 1]
Random:          [-1, -1, 1, 1]
Encrypted message:   [8, 2, 10, 23, 16, 7, 26, 2, 8, 3, 28]

====Let's decrypt====
Decrypted message:   [1, 0, 1, 0, 1, 1, 1]

Process finished with exit code 0
```

Using the latest suggested parameters, the NTRUEncrypt public key cryptosystem is secure to most attacks. There continues however to be a struggle between performance and security. It is hard to improve the security without slowing down the speed, and vice-versa. One way to speed up the process without damaging the effectiveness of the algorithm, is to make some changes in the secret key f. First, construct f such that, f= 1 +pF in which F is a small polynomial (i.e., coefficients {-1,0,1}). By constructing f this way, f is invertible mod p. In fact, f-1= 1(mod p), which means that Bob does not have to actually calculate the inverse and that Bob does not have to conduct the second step of decryption. Therefore, constructing f this way saves a lot of time but it does not affect the security of the NTRUEncrypt because it is only easier to find fp but f is still hard to recover. In this case f has coefficients different from -1, 0 or 1, because of the multiplication by p. But because Bob multiplies by p to generate the public key h, and later on reduces the ciphertext modulo p, this will not have an effect on the encryption method. Second, f can be written as the product of multiple polynomials, such that the polynomials have many zero coefficients. This way fewer calculations have to be conducted. Second, f can be written as the product of multiple polynomials, such that the polynomials have many zero coefficients. This way fewer calculations have to be conducted. In most commercial applications of the NTRUEncrypt, the parameter N=251 is used. To avoid lattice attacks, brute force attacks and meet-in-the-middle attacks, f and g should have about 72non-zero coefficients.



```
Python 3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
================== RESTART: D:\Desktop\NASSCOM Project\poly.py =================
>>>
================== RESTART: D:\Desktop\NASSCOM Project\ntru.py =================
>>>
============ RESTART: D:\Desktop\NASSCOM Project\example_bobalice.py ===========
Public Key is generated by using Parameters: N = 7, p = 29, and q = 491531
f(x)=  [1, 1, -1, 0, -1, 1]
g(x)=  [-1, 0, 1, 1, 0, 0, -1]
d   =  2


Public Key Generated by Bob:  [394609, 27692, 62307, 263073, 346149, 41538, 339225]


Original Message to be send:  [1, 0, 1, 0, 1, 1, 1]
Random Polynomial:  [-1, -1, 1, 1]
Encrypted Message to be send:  [283889, 269991, 484569, 353054, 179995, 159222, 235409]


Bob decrypts message sent to him
Decrypted Message:  [1, 0, 1, 0, 1, 1, 1]
>>> 
```

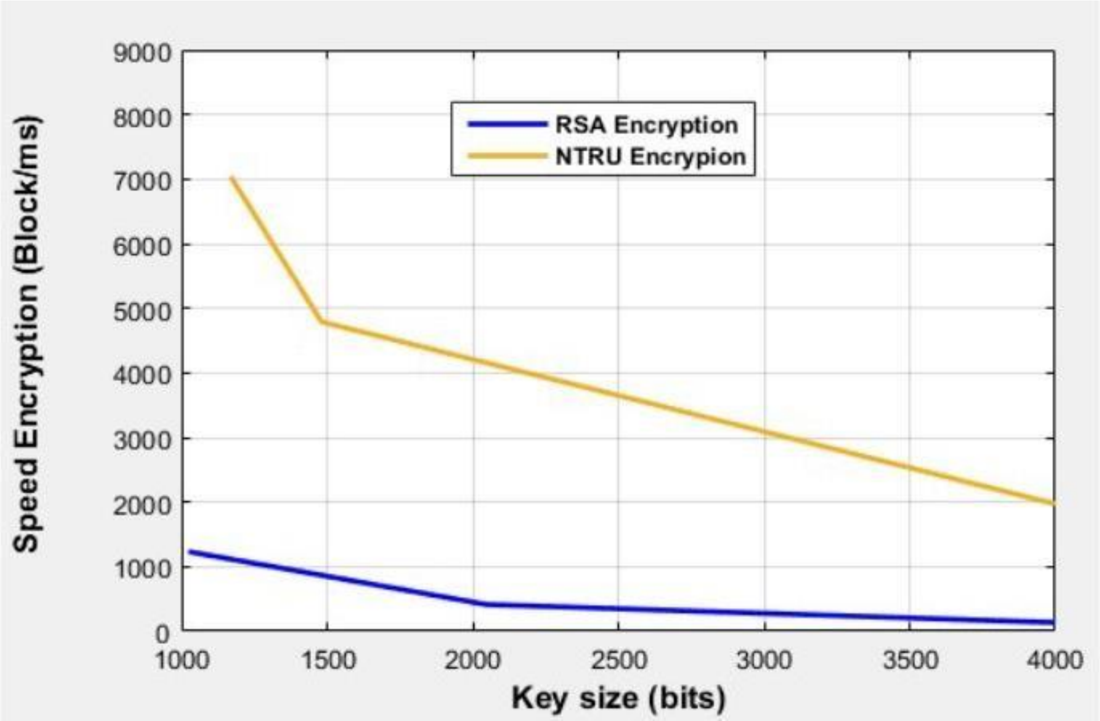
Output of our example

8. COMPARSION OF NTRU WITH OTHER PUBIC KEY CRYPTOSYSTEMS

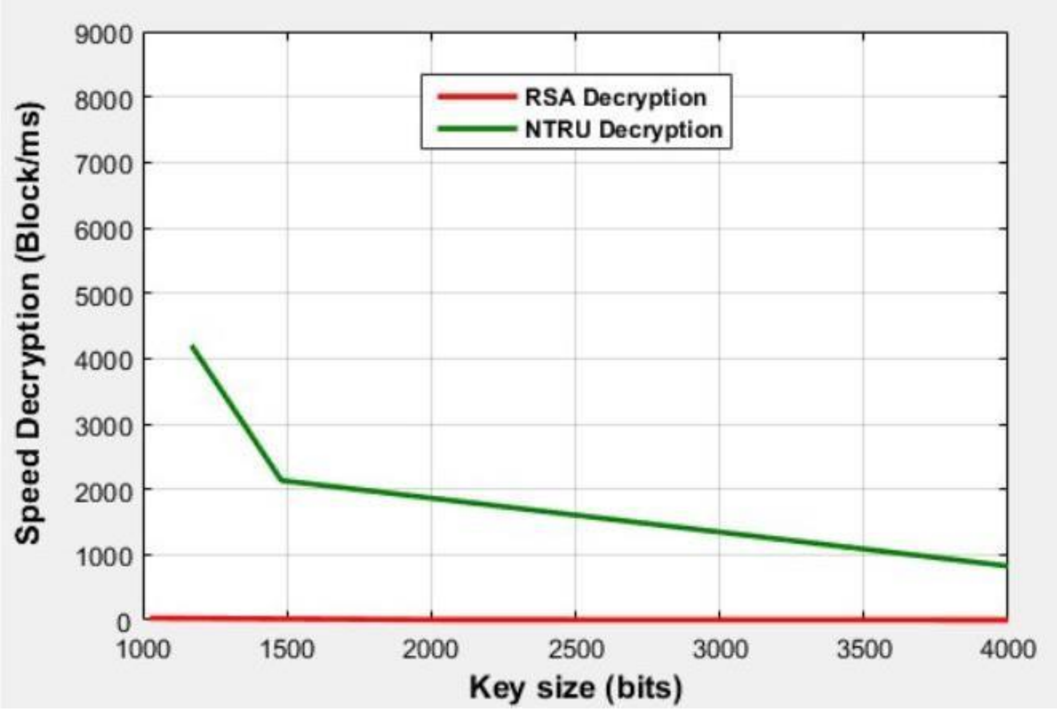
Public-key cryptography uses two separate keys for encryption and decryption. Even though it is not the same key, the two parts of this key pair are mathematically linked. The public key is used to encrypt plaintext or to verify a digital signature which can be announced to the public. The private key used to decrypt ciphertext or to create a digital signature must be kept secret. Public key algorithm are usually based on the computational complexity of hard problems. For example, the RSA cryptosystem is based on the difficulty of the integer factorization problem, Elliptic curve cryptography (ECC) is based on the difficulty of number theoretic problems involving elliptic curves.

8.1 RSA

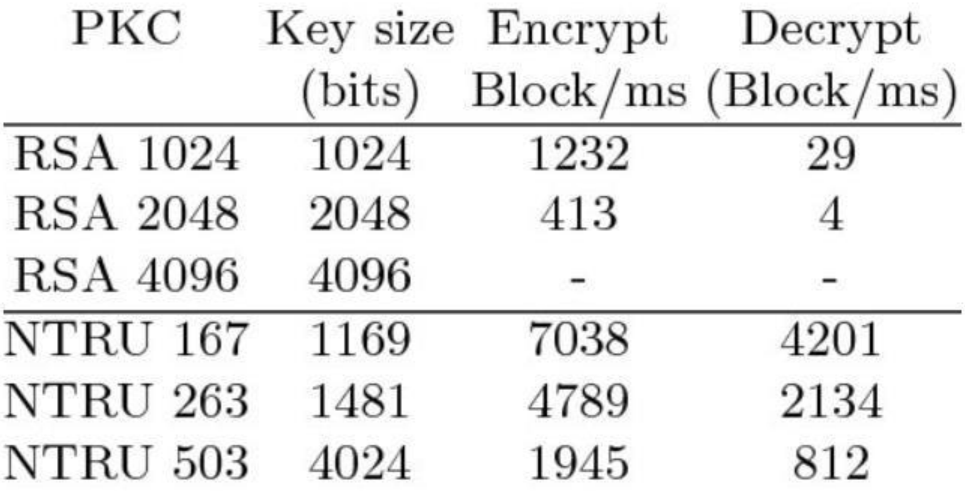
RSA is one of the first public key cryptosystem. RSA stands for Ron Rivest, Adi Shamir and Leonard Adleman and was proposed in 1977. The three main steps of RSA algorithm are key generation, encryption and decryption. Public key cryptosystems are mainly used to exchange either a secret key or a short message. RSA and NTRU work in units of message blocks, and in both systems, a single message block is large enough to maintain a short message or a secret key at high security. Therefore, in comparison between these two systems, we will evaluate the length of time to encrypt and decrypt a single message block.



Encryption speed (Block/ms) vs Key size (bits)



Decryption speed (Block/ms) vs Key size (bits)

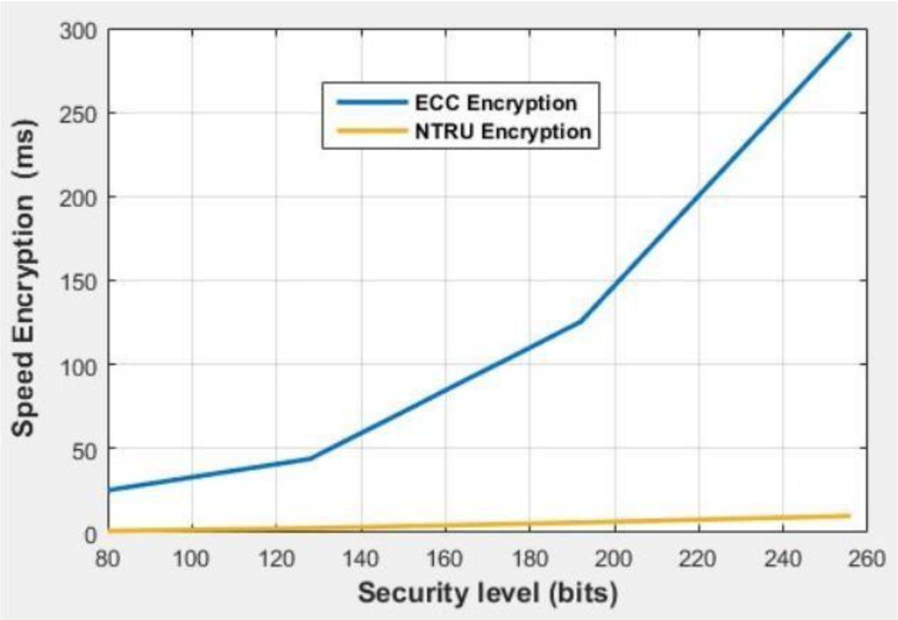


PKC	Key size (bits)	Encrypt Block/ms	Decrypt (Block/ms)
RSA 1024	1024	1232	29
RSA 2048	2048	413	4
RSA 4096	4096	-	-
NTRU 167	1169	7038	4201
NTRU 263	1481	4789	2134
NTRU 503	4024	1945	812

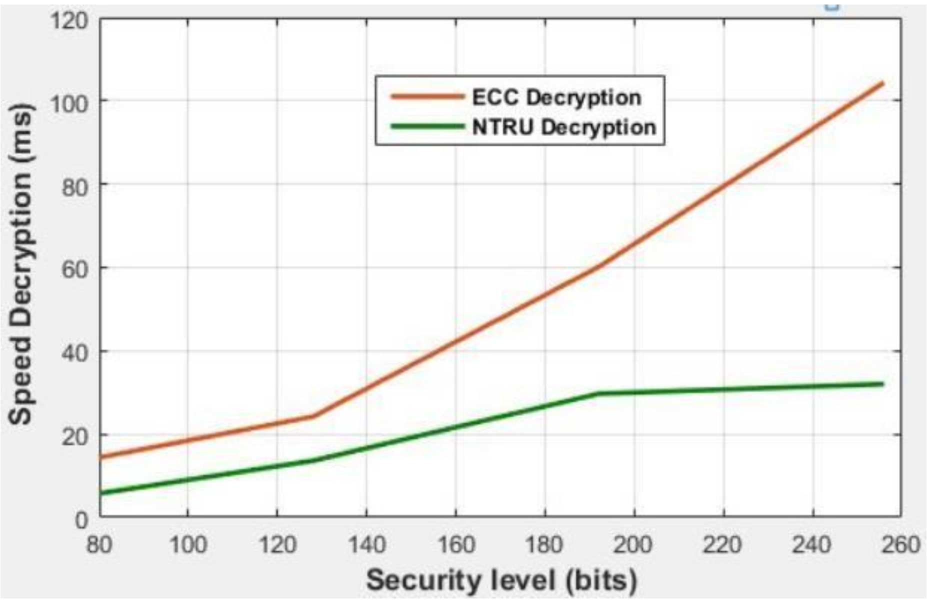
Clearly, we can conclude that the performance of NTRU is better than RSA in terms of key generation, encryption and decryption. We can also see that the key generation of NTRU is faster than in RSA. Furthermore, NTRU encrypts more messages than RSA with the same key sizes. Finally, the NTRU also decrypts more messages than RSA with the same key sizes.

8.2 ECC CRYPTOSYSTEM

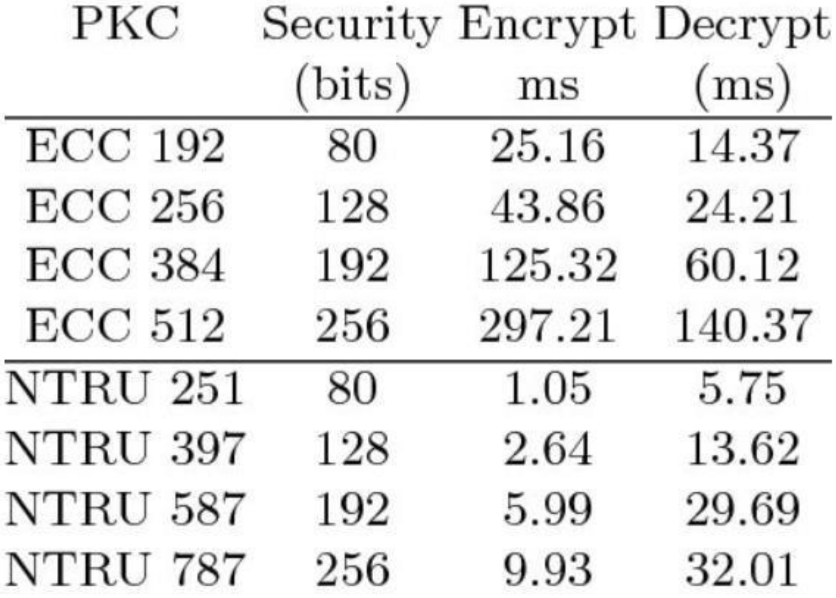
Elliptic-curve cryptography (ECC) is an approach to public-key cryptography based on the algebraic structure of elliptic curves over finite fields. ECC requires smaller keys compared to non-EC cryptography (based on plain Galois fields) to provide equivalent security. Elliptic curves are applicable for key agreement, digital signatures, pseudo-random generators and other tasks. Indirectly, they can be used for encryption by combining the key agreement with a symmetric encryption scheme. They are also used in several integer factorization algorithms based on elliptic curves that have applications in cryptography, such as Leenstra elliptic-curve factorization.



Encryption speed (ms) vs Security level (bits)



Decryption speed (ms) vs Key size (bits)



PKC	Security (bits)	Encrypt ms	Decrypt (ms)
ECC 192	80	25.16	14.37
ECC 256	128	43.86	24.21
ECC 384	192	125.32	60.12
ECC 512	256	297.21	140.37
NTRU 251	80	1.05	5.75
NTRU 397	128	2.64	13.62
NTRU 587	192	5.99	29.69
NTRU 787	256	9.93	32.01

From above graph, we discovered that the NTRU is much faster than the ECC with all levels of security. From the figures we also concluded that the performance of NTRU is superior in comparison to the performance of minimum values of ECC timings. Similarly, we also concluded that the performance of NTRU is superior in comparison to the performance of maximum values of ECC timings.

9. CONCLUSION and Drawback

After successfully implementing the python code for NTRU encryption and decryption, as a programmer it can be understood that decoding the encrypted message will be very difficult for a third user who is trying to break in the system. After analyzing the performance of NTRU with respect to other public key cryptosystems such as RSA and ECC it was found that it is much faster and more secure. The first part of this project described the NTRU cryptosystem parameters, private key, public key, encryption and decryption.

Next, some of the general attacks against the NTRU cryptosystem were discussed such as alternate private keys, brute force, meet-in-the-middle, multiple transmission and lattice attacks. Following in the project we compared NTRU with RSA and ECC. The performance of RSA was compared with NTRU. In this comparison, it was concluded that NTRU was better than RSA and ECC if the security level starts from 192 bits to 256 bits. However, in terms of key generation, encryption, decryption and so on, NTRU out performed them both. Thus, through all the analysis we concluded that NTRU is a very potential alternative for android based text encryption and has wide applications in cryptography.

10. FUTURE WORK

There can be research in the ways of hacking into the cryptosystem and decrypt the encrypted message by a third user. Since it is known that NTRU cryptosystem is resistant to quantum computers, so other ways should be found out to break it so that the faults can be found out and its security can be improved.

11. REFERENCES

[1] https://resources.infosecinstitute.com/quantum-computation

[2] C. Paar and J. Pelzl, “Introduction to Public-Key Cryptography,” in Understanding Cryptography. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 149–171.

[3] Z. Kirsch, “Quantum Computing: The Risk to Existing Encryption Methods,” Ph.D. dissertation, Tufts University, Massachusetts, 2015,
http://www.cs.tufts.edu/comp/116/archive/fall2015/zkirsch.pdf

[4] https://www.microsoft.com/en-us/research/project/post-quantum-cryptography/

[5] Shen, X., Du, Z., & Chen, R. (2009). Research on NTRU Algorithm for Mobile Java Security.
2009 International Conference on Scalable Computing and Communications; Eighth International Conference on Embedded Computing.doi:10.1109/embeddedcom-scalcom.2009.72

[6] Nanda, A. K., & Awasthi, L. K. (2013). A Proposal for SMS Security Using NTRU Cryptosystem. Quality, Reliability, Security and Robustness in Heterogeneous Networks, 706–718.doi:10.1007/978-3- 642-37949- 9_62

[6] Arshpreet Kaur, Miss Lofty Sahi, NTRU based Security in Cloud Computing International Journal of Engineering And Computer Science ISSN:2319-7242Volume 7 Issue 6 June 2018, Page No. 24071-24075Index Copernicus Value (2015): 58.10, 76.25 (2016) DOI:
10.18535/ijecs/v7i6.09

[7] Hoffstein, J., Lieman, D., Pipher, J. and Silverman, J.H., 1999. NTRU: A public key cryptosystem. NTRU Cryptosystems, Inc.(www. ntru. com).

[8] https://www.wikiwand.com/en/NTRUEncrypt

[9]https://sdsudspace.calstate.edu/bitstream/handle/10211.3/135700/Nguyen

[10]http://people.scs.carleton.ca/~maheshwa/courses/4109/Seminar11/NTRU

[11]https://onlinelibrary.wiley.com/doi/full/10.1002/sec.1693