Enhancing the NRTU cryptosystem

Awnon Bhowmik
Department of Mathematics and Computer Science, CUNY York College
90-24 Guy R. Brewer Blvd
Jamaica, NY 11451
awnon.bhowmik@yorkmail.cuny.edu

Unnikrishnan Menon
Department of Electrical and Electronics Engineering, Vellore Institute of Technology
Vellore Campus, Tiruvalam Rd
Katpadi, Vellore, Tamil Nadu 632014
unnikrishnanr.menon2017@vitstudent.ac.in

May 5, 2020

Abstract

NTRU is an open-source public key cryptosystem that uses lattice-based cryptography to encrypt and decrypt data. Unlike other popular public-key cryptosystems, it is resistant to attacks sing Shor's Algorithm and its performance has been shown to be significantly greater. This paper talks about how Koblitz encoding from Elliptic Curve Cryptography (ECC) can be used to convert each character in a dataset to a point on an elliptic curve. A Pythagoras Theorem analogy is used to turn the point to a single number, which is converted to a sequence of coefficients in Z. A polynomial is then generated for each of these characters. Then the polynomial is reduced, and we show that choosing appropriate parameters for the cryptosystem can make it highly secure and that the decryption algorithm turns out taking linear time. Since each character is represented by its own polynomial, it increases obscurity thereby increasing the complexity for decryption and thus the security level. We also implement a form of data compression and test whether data compression and expansion during the encryption-decryption process results in original data with no or minimal loss.

keywords: post quantum cryptography, lattice based encryption, quantum cryptography, Koblitz encoding, post quantum cryptosystem, ntru cryptography. ntru cryptosystem

1 Why Lattice Cryptography

The real reason[1]

- In 1994, Shor's Algorithm break RSA and ECC with quantum computers
- 2015, NSA announcement: prepare for the quantum apocalypse
- 2017, NIST call for competition/standardization
- 2030, predicted general purpose quantum computers

Further usefulness[1]

- Good understanding of underlying hard problem

- Fast, parallel-able, hardware friendly
- Numerous applications: FHE, ABE, MMap, Obfuscation...

Data vaulting attack[1]

- A.k.a. harvest-then-decrypt attack
- Data needs to be secret for, lets say, 30 years
- Quantum computer arrives in, lets say, 15 years
- Perhaps the most practical attack in cryptography

Encrypt Schemes[1]

- NTRUEncrypt - standardized by IEEE and ASC X9

Signature Schemes[1]

- BLISS (NTRU)
- pqNTRUSign (NTRU)

This paper specifically focuses on the NTRUEncrypt and NTRUDecrypt algorithm.

2 Koblitz Encoding & Decoding Algorithm

The encoding algorithm[2] is as follows...

- Given a message M, convert each character m_k into a number a_k using Unicode, where $b = 2^{16}$ and $0 < a_k < 2^{16}$
- Convert the message M into an integer using

$$m = \sum_{k=1}^{n} a_k b^{k-1}$$

  . In practice we choose an n to be less than or equal to 160 such that m satisfies $m \leq 2^{16 \cdot 160} < p$.
- Fix a number d such that $d \leq \frac{p}{m}$. In practice we choose the prime p large enough so that we can allow $d = 100$.
- For integers $j = 0, 1, 2, \cdots d-1$ we do the following loop
  - Compute the x coordinate of a point on the elliptic curve as $x_j = (dm + j) \mod p$ where $m = \left\lfloor \frac{x_j}{d} \right\rfloor$
  - Compute $s_j = (x_j^3 + Ax + B) \mod p$
  - If $(s_j)^{\frac{p+1}{2}} \equiv s_j \mod p$, then define y coordinate of a point on the elliptic curve as $y_j = (s_j)^{\frac{p+1}{4}} \mod p$. Return the point (x_j, y_j).

Thus we are able to encode our message M, as an element of the Abelian group $G = E(\mathbb{F}_p)$.
The decoding algorithm is as follows...

- Consider each point (x, y) and set m to be the greatest integer less than $\frac{x-1}{k}$. Then the point (x, y) decodes as the symbol m.

3 NTRUEncrypt

NTRU operations are based on objects in a truncated polynomial ring $R = \mathbb{Z}[X]/(X^N - 1)$ with convolution multiplication and all polynomials in the ring have integer coefficients and degree at most $N - 1$:

$$\mathbf{c}(X) = c_0 + c_1X + c_2X^2 + \cdots + c_{N-1}X^{N-1} \tag{1}$$

4 Equivalence class modulo 3

Any number divided by 3 can result in remainders $\{0, 1, 2\}$. So any number divided by 3 is one of $3r, 3r + 1, 3r + 2$. This is useful in constructing the trinary basis $\{-1, 0, 1\}^{\dim}$. The elements in the basis forms the coefficients of the polynomial (1).

Theorem. *Let $R \subseteq S \times S$ be an equivalence class on a set S. Then the set of $\mathcal{R}$-classes constitutes the whole of S*

Proof.

$$\begin{aligned}
\forall x \in S : x \in [x]_{\mathcal{R}} \quad & \text{Definition of equivalence class} \\
\neg\left(\exists x \in S : x \notin [x]_{\mathcal{R}}\right) \quad & \text{Assertion of universality} \\
\neg\left(\exists x \in S : x \notin \bigcup [x]_{\mathcal{R}}\right) \quad & \text{Definition of set union} \\
\forall x \in S : x \in \bigcup S/\mathcal{R} \quad & \text{Assertion of universality} \\
S \subseteq \bigcup S/\mathcal{R} \quad & \text{Definition of subset}
\end{aligned}$$

Also:

$$\begin{aligned}
\forall X \in S/\mathcal{R} : X \subseteq S \quad & \text{Definition of equivalence class} \\
\bigcup S/\mathcal{R} \subseteq S \quad & \text{Union is smallest superset: general result}
\end{aligned}$$

By definition of set equality

$$\bigcup S/\mathcal{R} = S$$

and so the set of all $\mathcal{R}$-classes constitutes the whole of S[3]. □

This theorem tells us that in our case, we are using mod 3 to generate coefficients from the set $\{-1, 0, 1\}$ which is basically the numbers of the form $3r - 1, 3r, 3r + 1$. It means that the union of all these equivalence classes spans the entire set of integers before modulo reduction.

5 Lattice Cryptography

What is a lattice?[1]

A **lattice** consists of all the integral combinations of $d \leq n$ linearly independent vectors over $\mathbb{R}$.

$$\mathcal{L} = \mathbb{Z}\mathbf{b}_1 + \cdots + \mathbb{Z}\mathbf{b}_d = \{\lambda_1\mathbf{b}_1 + \cdots + \lambda_n\mathbf{b}_n : \quad \lambda_i \in \mathbb{Z}\}$$

where d is the dimension and $\mathbf{B} = (\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_n)$ is a basis. Here is an example

$$\mathbf{B} = \begin{pmatrix} 5 & \frac{1}{2} & \sqrt{3} \\ \frac{3}{5} & \sqrt{2} & 1 \end{pmatrix}$$

Here $d = 2, n = 3$ and hence $d \leq n$

- All crypto talks begin with an image of a dim-2 lattice
- There is an infinitude of bases which is formed due to matrix multiplication of **B** and some other matrix
- It involves solving the shortest vector problem (SVP). The exact version of the problem is only known to be NP-hard for randomized reductions.[4] We have used something similar in our algorithm, but here we calculate the sum of squares rather than the L^2 norm.

6 Flow Diagram for Proposed Algorithm

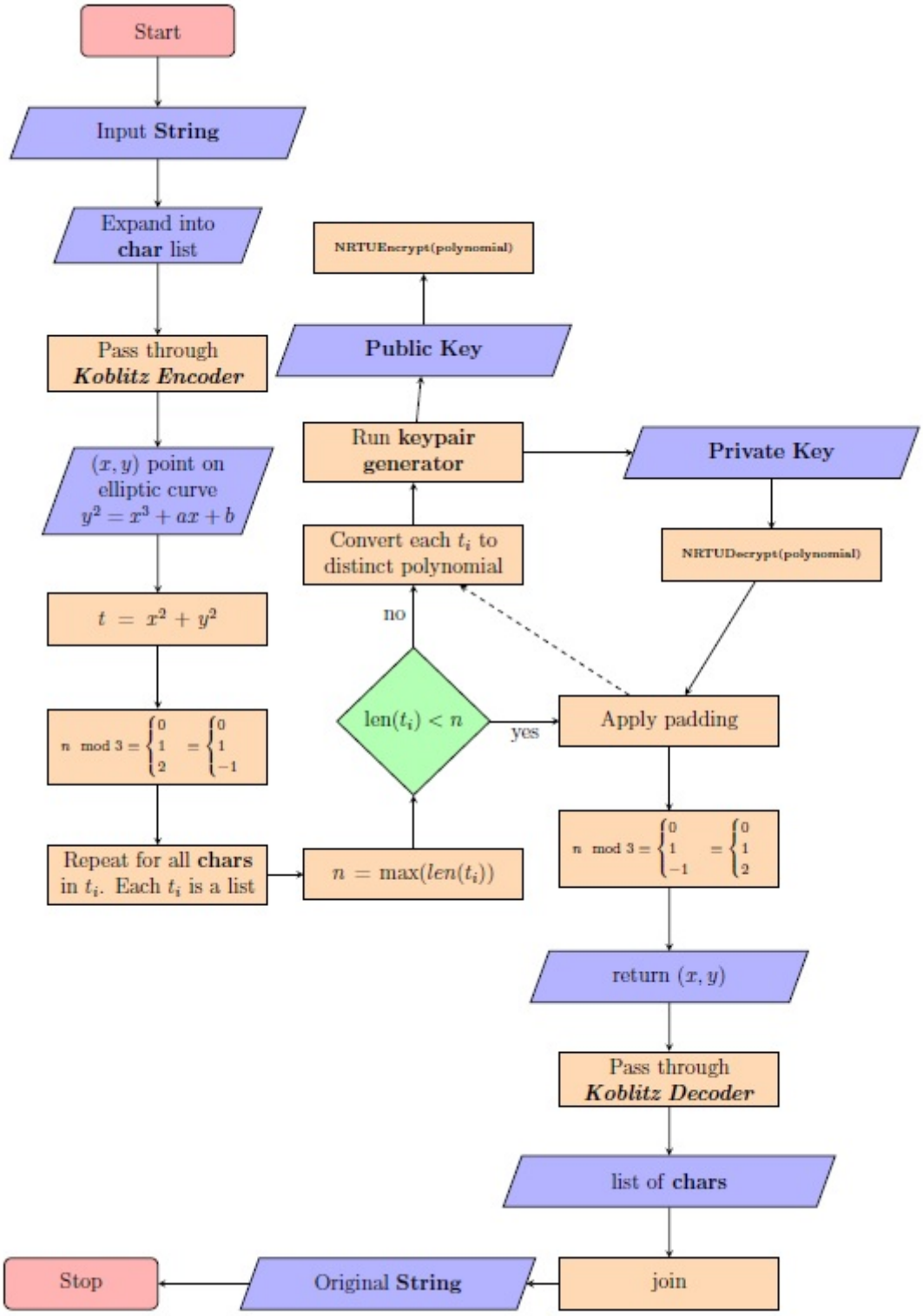

Fig 1. Flowchart

7 Proposed Algorithm

1. Input an arbitrary string containing alphanumeric and/or special characters
2. Split into characters and pass through *Koblitz encoder*
3. Each character in the string is turned into a point on an elliptic curve $y^2 = ax^3 + bx + c$, whose parameters a, b, c are user defined
4. Convert point(s) (x, y) to a single number using the parametrization $t = x^2+y^2$ and apply modulo 3 to turn it into ternary base number. Here $\{0, 1, 2\} = \{0, 1, -1\}$. Store these ternary numbers into a list. There should be multiple such lists due to multiple characters in the original string.
5. Find the maximum size among these lists. For the smaller lists, we apply padding to maintain consistency
6. Using these ternary values from each list, every character in the string is represented as a polynomial
7. We generate the public and private keys
8. Use public key to run NTRUEncrypt() and generate cipher text.
9. Use private key to run NTRUDecrypt() and obtain original lists of ternary numbers
10. Reapply padding
11. Apply a method called ternary_to_decimal() to convert the padded lists into (x, y) points
12. Pass the (x, y) points through *Koblitz Decoder* to obtain the list of characters. Combining these characters, finally allows us to retrieve the original plain text.

8 An attempt on loss less encoding

Messages encrypted by the Koblitz method results in a long polynomial. It is safe to assume that the length of this polynomial increases linearly with the length of message. Hence, a modification to the aforementioned algorithm in Fig 1. was attempted by converting the polynomials into a gray scale map to obtain some sort of data compression. Following sample results were obtained.

```
entropy@hyperspace:~/Desktop/NTRU_cryptography$ python3 test.py
Convert this to image:
[[0.21428571428571427, 0.10714285714285714, 0.29464285714285715], [0.22321428571428573, 0.24107142857142858, 0.25], [0.39285714285714285
, 0.11607142857142858, 0.3125], [0.33035714285714285, 0.0, 1.0]]
Generating Image...
crap.py:43: DeprecationWarning: `imsave` is deprecated!
`imsave` is deprecated in SciPy 1.0.0, and will be removed in 1.2.0.
Use ``imageio.imwrite`` instead.
  scipy.misc.imsave('encrypted_image.png', to_img)
Reading from image
Data Type: uint8
Min: 0.000, Max: 255.000
[[0.21568627 0.10588235 0.29411765]
 [0.22352941 0.23921569 0.25098039]
 [0.39215686 0.11764706 0.31372549]
 [0.32941176 0.         1.        ]]
entropy@hyperspace:~/Desktop/NTRU_cryptography$
```

Fig 2. Attempt 1

```
entropy@hyperspace:~/Desktop/NTRU_cryptography$ python3 test.py
Convert this to image:
[[0.21428571428571427, 0.10714285714285714, 0.29464285714285715], [0.22321428571428573, 0.24107142857142858, 0.25], [0.39285714285714285
, 0.11607142857142858, 0.3125], [0.33035714285714285, 0.0, 1.0]]
Generating Image...
Reading from image
Data Type: float64
Min: 0.000, Max: 1.000
[[0.21428571 0.10714286 0.29464286]
 [0.22321429 0.24107143 0.25      ]
 [0.39285714 0.11607143 0.3125    ]
 [0.33035714 0.         1.        ]]
entropy@hyperspace:~/Desktop/NTRU_cryptography$
```

Fig 3. Attempt 2

The first attempt involved PNG image format and produced results that were correct up to 2 decimal places only. This means that this format produced a large percentage error. Another thing to notice is that this involved only 8 bit precision unsigned integers. However, using TIFF image format with 64 bit precision unsigned integers resulted in a much better match between the encrypted and the decrypted matrix. Following is a sample gray scale map that was generated.

Fig 4. Gray scale map of data compression

Remark: It is to be noted that every character in our input string is treated differently, i.e. if we write the message "banana", then each of the characters have a different random polynomial. The polynomials for all 3 a's and both n's are distinct. This is what makes this cryptosystem more secure.

9 Result

The following snapshot resembles a sample test run of the entire program.

```
entropy@hyperspace:~/Desktop/NTRU_cryptography$ python3 main.py

Enter Message: Show me da $$$

Curve Parameters
Enter A: -96
Enter B: -48

Public Key = -18x^14+-39x^13+14x^12+-1x^11+42x^10+-7x^9+30x^8+57x^7+-10x^6+52x^5+27x^4+39x^3+-13x^2+-34x+14

Encrypted =
-31x^14+48x^13+5x^12+-42x^11+26x^10+17x^9+-23x^8+16x^7+-47x^6+41x^5+-29x^4+-51x^3+-26x^2+-9x+1
4x^14+0x^13+43x^12+29x^11+-27x^10+-50x^9+-7x^8+-19x^7+40x^6+-51x^5+-32x^4+25x^3+-31x^2+25x+23
45x^14+14x^13+60x^12+-3x^11+-57x^10+48x^9+44x^8+34x^7+30x^6+45x^5+2x^4+-12x^3+18x^2+36x+31
4x^14+44x^13+-30x^12+32x^11+41x^10+-60x^9+26x^8+-35x^7+11x^6+-45x^5+-21x^4+-54x^3+61x^2+52x+-50
22x^14+49x^13+-18x^12+-26x^11+51x^10+42x^9+28x^8+38x^7+22x^6+38x^5+37x^4+-10x^3+18x^2+28x+39
25x^14+53x^13+38x^12+-48x^11+31x^10+-49x^9+55x^8+-31x^7+-57x^6+63x^5+-58x^4+33x^3+-18x^2+-46x+-14
-11x^14+15x^13+9x^12+-2x^11+-62x^10+25x^9+19x^8+54x^7+34x^6+-6x^5+54x^4+-46x^3+-28x^2+45x+-50
-20x^14+50x^13+-1x^12+-35x^11+-9x^10+51x^9+34x^8+-2x^7+-58x^6+39x^5+-46x^4+-49x^3+-38x^2+-21x+29
-59x^14+-6x^13+35x^12+-57x^11+57x^10+-51x^9+-48x^8+19x^7+-38x^6+-8x^5+-36x^4+-63x^3+-27x^2+20x+32
5x^14+-4x^13+-12x^12+2x^11+-25x^10+40x^9+55x^8+-24x^7+-15x^6+-5x^5+-15x^4+3x^3+-12x^2+-6x+37
27x^14+33x^13+7x^12+-25x^11+12x^10+-59x^9+35x^8+43x^7+5x^6+-60x^5+-26x^4+-23x^3+-31x^2+19x+17
5x^14+-1x^13+-22x^12+-39x^11+-61x^10+25x^9+47x^8+34x^7+-23x^6+-63x^5+46x^4+-1x^3+2x^2+58x+-27
25x^14+-51x^13+-16x^12+0x^11+-53x^10+-24x^9+14x^8+42x^7+35x^6+23x^5+-41x^4+47x^3+43x^2+-49x+63
-2x^14+41x^13+-3x^12+31x^11+58x^10+-12x^9+47x^8+28x^7+42x^6+-10x^5+-34x^4+14x^3+-17x^2+-39x+14

Decrypted = Show me da $$$

Successful decryption: True
```

Fig 5. Sample test run showing the entire process

10 Experiments by varying parameters

10.1 String length vs time to encrypt/decrypt

A function was written on Jupyter Notebook to test out the relationship between string length s (x-axis) and time of encryption/decryption $t_{\text{enc}}/t_{\text{dec}}$ (y-axis).

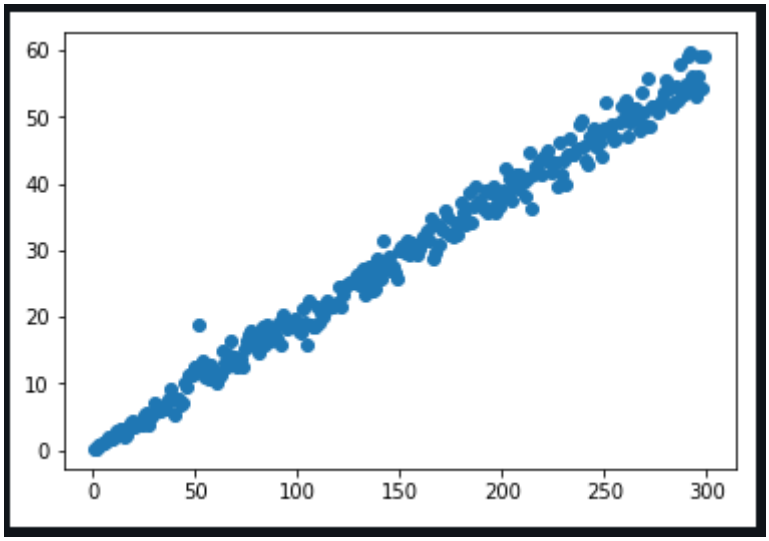

Fig 6. string length vs time to encrypt

A best fit shape through these points is a straight line through the *origin*. It can be safely assumed that for an arbitrary string length s, time to encrypt t_{enc} and time to decrypt t_{dec}, we can write

- $t_{\text{enc}} \propto s$
- $t_{\text{dec}} \propto s$

To test the authenticity of our encryption paradigm we fixed elliptic curve parameters $(a, b) = (-96, -48)$. We generated a custom string of length 74 that contained numeric values, alphabets (upper and lower case), special characters, spaces etc and generated 100 new strings by randomly shuffling the elements of this custom string. It was observed that for each of the 100 test cases, we were able to successfully retrieve the original message after completing an encrypt-decrypt cycle

10.2 Valid elliptic parameters vs String length

1. We take a custom string of length 74. This consists of all alphnumeric characters, special characters and space.
2. We run a $O(n^2)$ complexity loop to check for (a, b) in range $[-100, 100]$.
3. Due to lack of recommended processing power, we break out of the loop as soon as the first combination is found. The combination turned out to be $(a, b) = (-96, -48)$
4. We shuffled the original string 100 times to make 100 different strings of length 74.
5. For each of the shuffled strings, we only consider the first 20 characters due to shortage of processing power. So basically we now have 100 strings of length 20 that comprises of any random character on the English keyboard.
6. We run the encrypt-decrypt function on these strings. The job of the encrypt-decrypt function is to return true only if the decrypted message matches perfectly with the input plaintext.

We observed that for each of the 100 test cases, the encrypt-decrypt function returns true.

11 Conjecture

1. The elliptic curve parameter $(a, b) = (-96, -48)$ works irrespective of the string length of the message and the type of characters present in the string.
2. $(a, b) = (-96, -48)$ is one of the many combinations of elliptic curve parameter that works. Proof of this is dependent on greater computational power.

12 Future Work

We are planning to wrap up the entire program into a pip install package which would be available for general public use. NTRU algorithm has been around for a while but there is still no official implementation of this algorithm in libraries such as pycryptodome. We would love to see our version of NTRU becoming a part of this library. It will serve as a great security layer against post quantum cryptography. We would also like to incorporate this crypto system as a security layer for a web socket based client server chat app, built with Tkinter. For the moment, the work we have done so far has been compiled into a repository[5]. We have also noticed that using the numpy module makes quick work of calculations which we have coded from scratch. This would also fix some formatting issues to make the encrypted polynomial more legible.

13 Conclusion

In this paper, we show a method where we make use of concepts stemming from the core of elliptic curve cryptography and Pythagoras formula, in order to increase the complexity for decryption, thereby increasing the security level of the existing lattice based NTRU cryptosystem. We analyse how introducing these concepts tuning the appropriate set of parameters can lead to a more stable and secure encryption – decryption cycle with a lower time complexity.

This is significant because as of 2020, the most popular public-key algorithms can be broken by a sufficiently strong quantum computer. The problem with current popular algorithms is that their security relies on one of three hard mathematical problems: the integer factorization problem, the discrete logarithm problem or the elliptic curve discrete logarithm problem. All of these problems can be easily solved on a powerful quantum computer running Shor's algorithm. NTRU is resistant to attacks based on quantum computing, to which the standard RSA and ECC public-key cryptosystems are vulnerable to. NTRU is the first lattice-based public key cryptosystem not based on factorization or discrete logarithmic problems. This algorithm is based on the shortest vector problem (SVP) in a lattice.

While NTRU algorithm has been around for a while, it would be easier for hackers to find a way around it and funnel funds, liquid assets and/or confidential information from online accounts. Our method increases the complexity of breaching into such accounts by introducing additional layers of security that is purely based on fundamental mathematics. Our work so far is available here.[5]

References

[1] O. Security, "A short review of the ntru cryptosystem," Jul 2017. [Online]. Available: https://www.slideshare.net/OnBoardSecurity/a-short-review-of-the-ntru-cryptosystem

[2] N. D. Rene´e Brady and A. Tracy, "Encrypting with elliptic curve cryptography," July 2010. [Online]. Available: https://www.math.purdue.edu/~egoins/notes/Encrypting_Text_Messages_via_Elliptic_Curve_Cryptography.pdf

[3] "Union of equivalence classes is whole set." [Online]. Available: https://proofwiki.org/wiki/Union_of_Equivalence_Classes_is_Whole_Set

[4] M. Ajtai, "Generating hard instances of lattice problems," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, 1996, pp. 99–108.

[5] A. K. Bhowmik and U. R. Menon, "Ntru cryptography," 2020. [Online]. Available: https://github.com/7enTropy7/NTRU_cryptography