

---

# ANALYSIS OF LATTICE-BASED CRYPTOGRAPHY ALGORITHMS

---

**Amey Meher**  
Department of Computer Science  
North Carolina State University  
Raleigh, NC 27695  
avmeher@ncsu.edu

**Deep Mehta**  
Department of Computer Science  
North Carolina State University  
Raleigh, NC 27695  
dmehta2@ncsu.edu

**Gage Fringer**  
Department of Computer Science  
North Carolina State University  
Raleigh, NC 27695  
gwfringe@ncsu.edu

## Group IJ

### 1 Introduction

Due to the recent developments in Quantum computing, traditional methods for data encryption such as RSA are at the risk of getting obsolete. This is due to the fact that Quantum computers are capable of performing certain computations in a much faster sense than conventional computers, and by using this potential of computing power, attackers would be able to decrypt these traditional encryption algorithms.

In the advent of these technologies, NIST is introducing new standards for Post-Quantum era encryption. One of the approaches proving to be a promising option is lattice-based encryption schemes. Lattice-based cryptography has been studied actively over the last decade due to its distinctive advantages in strong security, fast implementations, and versatility applications. There are different implementations for the Lattice-based cryptography algorithms, namely Frodo, Lizard / Ring-Lizard, Kyber and NewHope. These few are the active candidates to post-quantum cryptography due to their applicabilities and efficiencies.

### 2 Background knowledge

With the concern that quantum computing possessed on current cryptographic schemes, NIST created the public project for Post-Quantum Cryptography (PQC) in 2017 to engage groups in working to implement algorithms with an aim of making them quantum-resistant in the hopes of eventually standardizing selected algorithms after multiple rounds of selection. This helped to motivate the importance of creating such an algorithm and also brought to light a variety of different attempts at creating these algorithms.

The algorithms are built on underlying problems which are assumed to not have polynomial solutions. As the RSA algorithm is based on finding the factors of large prime numbers, these algorithms are based on a variety of problems based on Lattices. Some examples of these problems relevant to the algorithms being observed are:

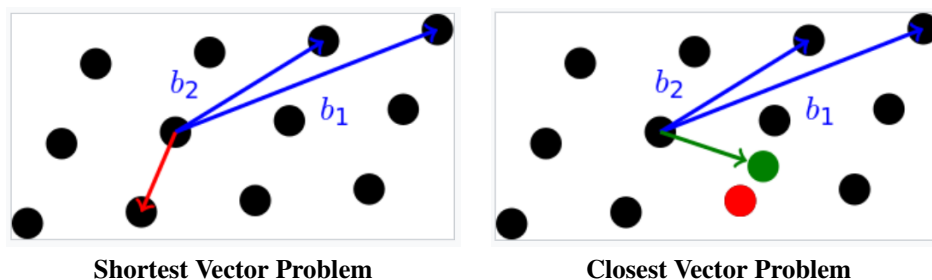


Figure 1: Visual representations of base problems

- Short vector problem: Suppose we are given a long basis for some lattice  $L$ . This problem asks us to find a grid point in  $L$  as close as possible to the origin point.

- Short basis problem: Suppose we are given a long basis for some lattice  $L$ . The short basis problem asks us to find a short basis for  $L$ .
- Closest vector problem: Suppose we are given a long basis for some lattice  $L$ . Also, we are given a randomly chosen challenge point  $P$ . The closest vector problem asks us to find the closest lattice point in  $L$  to challenge  $P$ .

### 3 Experiment setup

Our team has selected to survey the current state of PQC via some of the algorithms that have been developed to date. Along with this, we intend to gain in-depth knowledge of some of the relevant lattice-vector problems mentioned above and understand better why these do not have polynomial time solutions.

After researching some PQC algorithms, our team decided to focus on the NewHope and Kyber algorithms for their documentation in their public repositories, along with their similarities in gathering results. Both of these submissions to NIST's project have test files with various input sizes along with time measurements based on cycle counts, so we will plan to run these files to generate our own values for these algorithms and further analyze them accordingly. We plan to analyze both algorithms on the below factors:

- Trend for encryption time / operations for different message sizes
- Trend for decryption time / operations for different cipher sizes
- Trend for cipher sizes after encryption for different message sizes
- Analysis of breaking the cipher for a conventional computer

### 4 LWE Algorithm

In post-quantum cryptography, Learning with error is a key-agreement protocol that is designed to resist quantum computer attacks. NewHope is based on a mathematical problem ring learning with errors (RLWE) which are believed to be hard to break for both classical and quantum computers. Kyber is based on modular learning with errors (MLWE). This protocol offers a practical and secure solution for post-quantum key exchange for TLS and other applications.

The basic idea is that one party generates a random polynomial and adds some noise to it, then sends it to the other party. The other party also generates a random polynomial and adds some noise to it, then multiplies it with the received polynomial and sends back the result. Both parties can then use some clever techniques to extract the same secret key from their polynomials

Some of the design choices of NewHope are:

1. It uses binomial sampling to generate the noise vectors, which is simpler and faster than using Gaussian sampling.
2. It uses a novel method for error reconciliation that corrects errors in groups of 2 or 4 coefficients at a time, which reduces the decryption failure rate and increases the security.
3. It derives the base vector from the output of a hash function, which prevents the use of "back-doored" values that could compromise the security

It is an unauthenticated key-exchange protocol; by not designing or instantiating a lattice-based authenticated key-exchange protocol we reduce the complexity of the key-exchange protocol and simplify the choice of parameters. It is an advantage to decouple key exchange and authentication as it allows a protocol designer to choose the optimal algorithm for both tasks.

For the same lattice dimension, this algorithm doubles the security parameter, halves the communication overhead, and speeds up computation by more than a factor of 8 in a portable C implementation and by more than a factor of 27 in an optimized implementation targeting current Intel CPUs. These speedups are achieved with comprehensive protection against timing attacks.

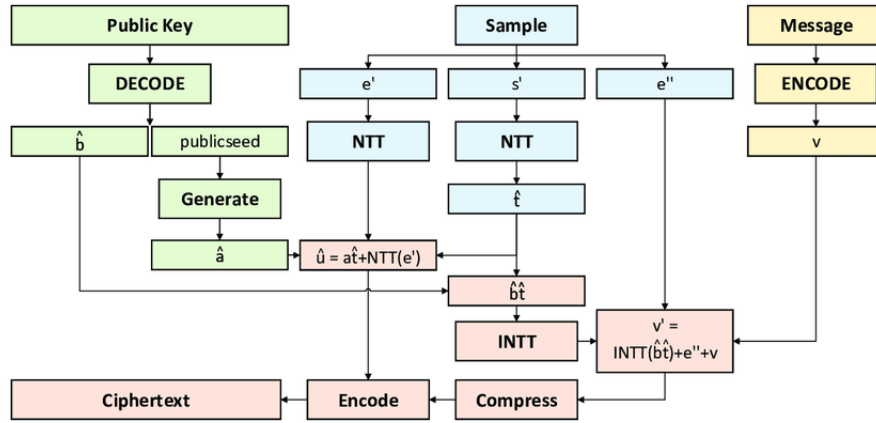


Figure 2: New Hope Encryption Process

The protocol consists of four main steps: parameter generation, key generation, reconciliation, and shared secret derivation. The space and time analysis of NewHope depends on the choice of parameters and the implementation platform. Here is a brief overview of the space and time complexity of NewHope:

1. **Parameter generation:** This step involves generating a public base vector  $a$  from a seed using a hash function. The space complexity is  $O(n)$  bits, where  $n$  is the dimension of the ring. The time complexity is  $O(n)$  hash function evaluations.
2. **Key generation:** This step involves sampling two secret polynomials  $s$  and  $e$  from a binomial distribution, computing  $b = as + e \pmod q$ , where  $q$  is a prime modulus, and sending  $b$  as the public key. The space complexity is  $O(n \log q)$  bits for storing  $s$ ,  $e$ , and  $b$ . The time complexity is  $O(n \log n)$  operations for polynomial multiplication using NTT, plus  $O(n)$  operations for sampling and addition.
3. **Reconciliation:** This step involves receiving a public key  $b'$  from the other party, computing  $u = bs' + e' \pmod q$ , where  $s'$  and  $e'$  are secret polynomials sampled from a binomial distribution, and applying a reconciliation function to  $u$  to obtain a bit string  $c$ . The space complexity is  $O(n \log q)$  bits for storing  $s'$ ,  $e'$ ,  $u$ , and  $c$ . The time complexity is  $O(n \log n)$  operations for polynomial multiplication using NTT, plus  $O(n)$  operations for sampling, addition, and reconciliation.
4. **Shared secret derivation:** This step involves applying a hash function to  $c$  to obtain a shared secret  $z$ . The space complexity is  $O(k)$  bits, where  $k$  is the output length of the hash function. The time complexity is  $O(k)$  hash function evaluations.

The overall space complexity of NewHope is  $O(n \log q + k)$  bits, where  $n$  is the dimension of the ring,  $q$  is the prime modulus, and  $k$  is the output length of the hash function. The overall time complexity of NewHope is  $O(n \log n + k)$  operations, where  $n$  is the dimension of the ring and  $k$  is the output length of the hash function.

## 5 Implementation

Given that little information was found on these algorithms in terms of pseudocode or how they specifically work, the analysis was done with forks of each algorithm's public repositories (links in the references). Within these copies, each algorithm had 'ref' folders which contained test files that were used after compilation. All of these test files were run, and their results were stored in a 'results' directory within the ref folder.

All of the test file runs were done on the following hardware:

- Ubuntu 20.04.5 LTS Virtual Machine
- Proc: Intel Core i7-8565U @ 1.80 GHz (4 of 8 cores)
- Memory: 7.8 GB

## 6 Analysis

### 6.1 Differences in Timing

The section with some of the most notable differences was in looking at the measured number of clock cycles on average when running either of these two algorithms. Both algorithms' 'speed' tests provided these statistics for a number of different categories, but the three focused on are 'Keypair' (where the keys are generated), 'Encapsulation' (where the message is encrypted using the key), and 'Decapsulation' (unencrypting the message with the key).

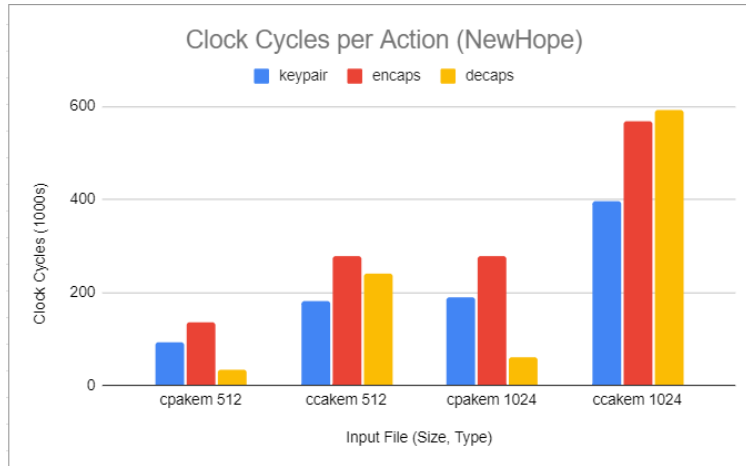


Figure 3: Clock Cycle speeds for the NewHope algorithm

Looking at the above figure, we can see that a general trend of an increase in clock cycles occurs as the input size increases from 512 to 1024. A more interesting observation to note was that the 512 byte input had a similar runtime to the 1024 byte input when the 512 byte input used Chosen-ciphertext Attack (CCA) security and the 1024 byte input did not. Generally speaking, the use of CCA security seems to double the required time for generating keys and encrypting the data when compared to a same-size input not using CCA. This makes sense because NewHope could be considerably faster when not needing to implement extra measures to handle CCA security. The other thing to note is that inputs that do not use CCA security seem to have a substantially faster decapsulation time, which could be attributed to the fact that with CCA, extra work is done to attempt to gain the secret key, which could substantially impact the process of decryption when CCA is enabled.

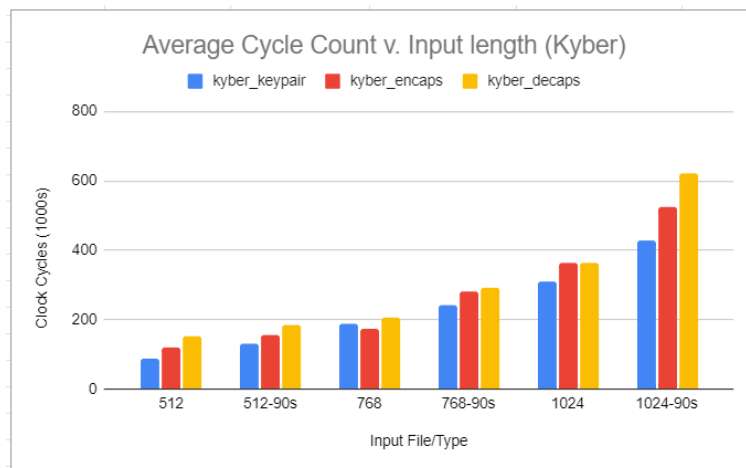


Figure 4: Clock Cycle speeds for Kyber algorithm

The above figure follows a similar trendline to Figure 2 in that larger input sizes tend to have longer runtimes. An interesting observation comes in how the '90s' versions of input sizes have a larger difference from their base file as the

input size increases, which is likely due to the fact that the 90s files use AES-256 in counter mode and SHA2 instead of SHAKE. These files were introduced in a later iteration of the NIST submission, and was meant to showcase how Kyber would behave on hardware where the symmetric primitives exist. The outliers in spread of values for 1024-90s may exist because the hardware used may not be what was considered appropriate by the code authors, causing some issues in validity of that data. Otherwise, it was interesting to see that encapsulation and decapsulation took similar clock cycles.

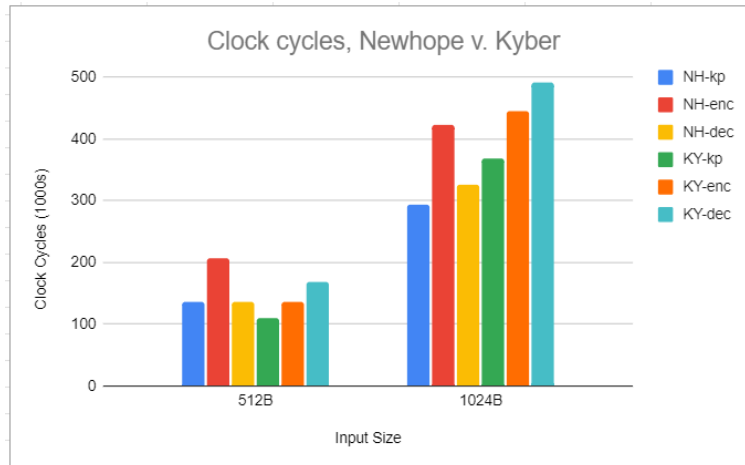


Figure 5: Direct Comparison of both algorithms

It is important to note that before analysis was done on this direct comparison between the two algorithms that since each algorithm had two iterations of its testing, these were averaged together before a direct comparison, as seen in the figure above. That being said, it could be seen that Newhope is generally a more costly algorithm when using 512 byte input, but Kyber tends to take a bit longer overall with larger inputs. This is interesting because while the two algorithms are based on the same problem (LWE), Kyber utilizes some understanding of two subsets of the problem (Module-LWE and Ring-LWE) where NewHope focuses on Ring-LWE, and the impact of this could be the cause of increased encryption time. Another interesting note was that when taking the averages of these two algorithms and directly comparing them, each algorithm tends to follow a different trend in performance. For example, encapsulation was the most costly action for NewHope, while the same could be said for decapsulation with Kyber.

### 6.2 Differences in Cipher Size

Input Size (Bytes)	NewHope	Kyber
512	1104	768
1024	2184	1568

Table 1: Size of Cipher for a given input size

Looking at the above table, we can see that regardless of input size, NewHope has a larger cipher than Kyber. This is interesting to see because it would explain why some actions would take longer on a 512 byte input for NewHope (i.e. a longer cipher would mean more time having to encrypt/decrypt), but would not be the same reasoning for why NewHope would be faster than Kyber on the larger input size. This could be because Kyber seems to specify a target cipher size, so the input size had the same cipher size regardless of file type (90s or base), and since NewHope did not seem to have this, the variance increased cipher sizes.

## 7 Conclusion

When looking holistically at all of the above observations, we can see that the current state of PQC algorithms is a fairly wide field, but NIST has decided to take the charge on making sure that it is a matter of importance. Even algorithms such as NewHope and Kyber that have the same basis yield substantially different results, but the consistency of Kyber's cipher size and generally more consistent times between all three actions made it the more desirable of these two. Future work would need to be done against other algorithms in this field (i.e. the others listed in the Introduction) to further understand the current space better.

## 8 References

- [1] [How Quantum Computers Break The Internet... Starting Now](#) - Veritasium, Mar. 20, 2023
- [2] [Frodo: Take off the Ring! Practical, Quantum-Secure Key Exchange from LWE](#) - Joppe Bos, Craig Costello, Léo Ducas, Ilya Mironov, Michael Naehrig, Valeria Nikolaenko, Ananth Raghunathan, and Douglas Stebila., 23rd ACM Conference on Computer and Communications Security, 2016.
- [3] [CRYSTALS – Kyber: a CCA-secure module-lattice-based KEM](#) - Joppe Bos, Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, and Damien Stehlé, International Association for Cryptologic Research, 2017
- [4] [Lizard: Cut off the Tail! Practical Post-Quantum Public-Key Encryption from LWE and LWR](#) - Jung Hee Cheon, Duhyeong Kim, Joohee Lee, and Yongsoo Song, International Association for Cryptologic Research, 2016
- [5] [Post-quantum key exchange – a new hope](#) - Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe, International Association for Cryptologic Research, 2015
- [6] [What is Lattice-Based Cryptography and Why You Should Care](#) - Joël Alwen. Medium. Jun. 15, 2018
- [7] [CRYSTALS-kyber GitHub](#). GitHub. Last Edited: Dec. 13, 2020
- [8] [NewHope GitHub](#). GitHub. Last Edited: Feb. 17, 2023
- [9] [Selected Algorithms - 2022, NIST PQC](#), National Institute of Standards and Technology, 2022
- [10] [NewHope - Wikipedia](#). Wikipedea. Last Edit: Dec. 6, 2022
- [11] [Efficient Parallel Implementations of LWE-Based Post-Quantum Cryptosystems on Graphics Processing Units](#) - Scientific Figure on ResearchGate. ResearchGate. 2020
- [12] [NewHope - Post-quantum key encapsulation](#), Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. Last Update: Apr. 14, 2020



# Analysis of Lattice-Based Cryptographic Algorithms

**Group IJ:**

Amey Meher

Deep Mehta

Gage Fringer

# RSA Encryption algorithm

It is based on the difficulty of factoring the product of two large prime numbers.

Consider the product of two large prime numbers as  $N$ . Could you guess would be the time complexity for factoring  $N$  into the original two prime numbers?



$$\exp((\log N)^{1/3} (\log \log N)^{2/3})$$

Time complexity for General number field sieve which is considered as the best known factoring algorithm on a conventional computer



# Shor's algorithm

Quantum computing has the ability to crack encryption algorithms that have been considered secure for decades.

Shor's algorithm, one of the most famous algorithms is able to factor product of large prime numbers in **polynomial time of  $O(N^3)$** , reducing the time to break the encryption from trillions of years to just few hours.

# Construction of a new problem

q (Max coefficient) = 17

Random problem

7	10	5	3
1	6	2	10
7	3	11	6
4	15	2	9
11	8	5	12
16	9	8	5

**X**

Secret key

x <sub>1</sub>
x <sub>2</sub>
x <sub>3</sub>
x <sub>4</sub>

=

3
11
6
8

Given a random problem matrix and a matrix obtained after multiplying with a secret matrix, find the values of the secret matrix.

Easy to solve for the matrix using Gaussian elimination technique (Algebra 101)

# Introducing error matrix

q (Max coefficient) = 17

Random problem

7	10	5	3
1	6	2	10
7	3	11	6
4	15	2	9
11	8	5	12
16	9	8	5

$\times$

Secret  
key

$x_1$
$x_2$
$x_3$
$x_4$

$+$

Random  
error

$e_1$
$e_2$
$e_3$
$e_4$

$=$

3
11
6
8

Though after introducing an error by adding the error matrix to the result, this makes it difficult to guess the secret key.

This fundamental problem which is known as Learning with Errors (LWE) is used in the new cryptography algorithms which are accepted by NIST.

# Learning with Errors (LWE)

This is at least as hard as factorization of big integers, but also Quantum safe

# NP-Hardness proof

**Worst case Gap Shortest Vector Problem**

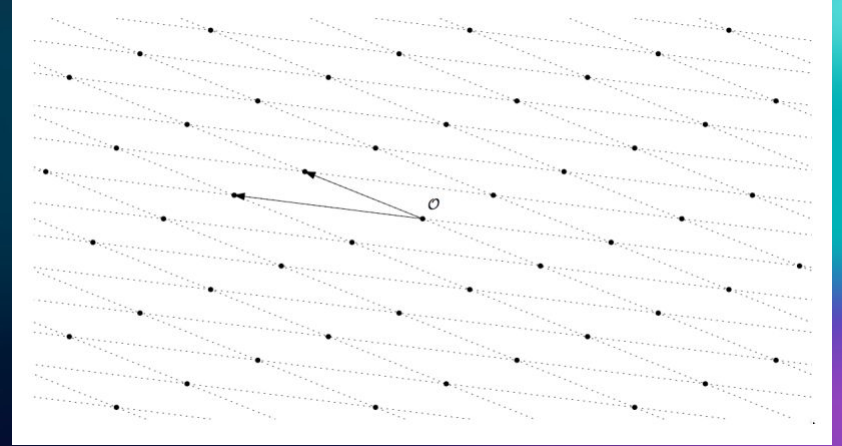
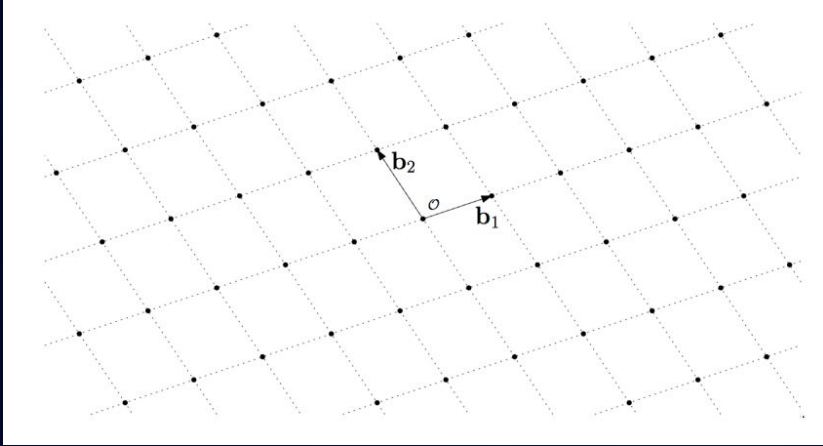


Reduction

**Average case Learning with error Problem**

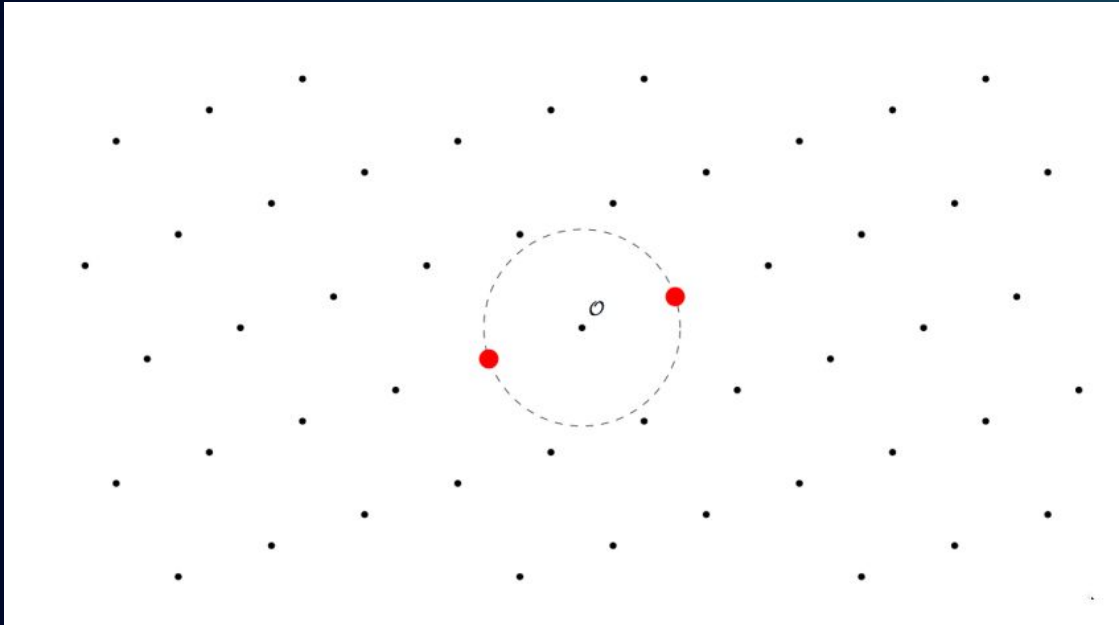


# Lattices



Integer linear combinations of a certain basis. Also, there are many bases for the same lattice, some short and orthogonal whereas other long and acute.

# Shortest vector problem



Given some basis for the lattice, find the shortest non-zero lattice point.



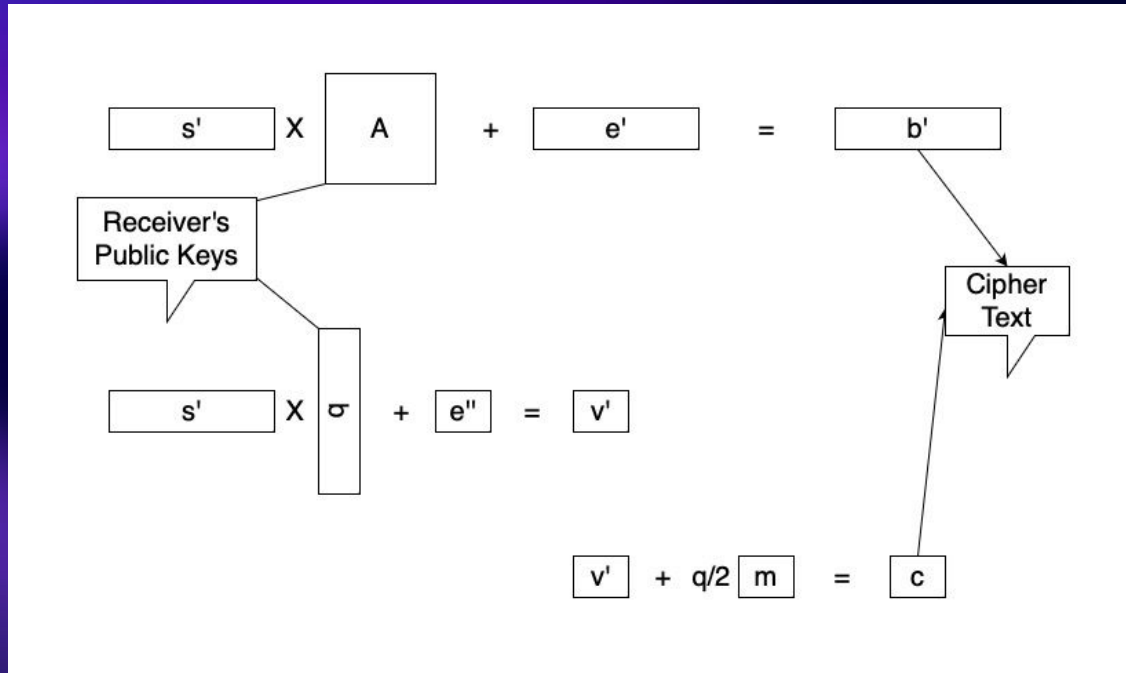
# Regev's iterative reduction

Theorem. [Reg05]

For any modulus  $q < 2^{\text{poly}(n)}$ , solving the decision LWE problem is at least as hard as quantumly solving GapSVP, and SVP, on arbitrary  $n$ -dimensional lattices.

# **General working of the cryptography algorithms**

# Encryption



**A and b:** Receiver's public key which is available to sender

**s':** Sender's secret key

**e':** Random error

**q:** Random prime number

**b' and c:** Cipher text sent from sender to receiver

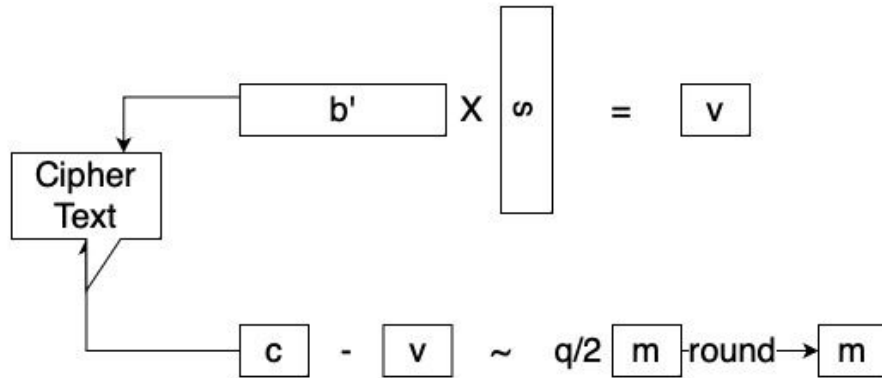
# Decryption

**b' and c: Cipher text received by receiver**

**s: Receiver's secret key**

**q: Random prime number**

**m: Deciphered/Original text**



# Lattice based cryptography methods

## NewHope

NewHope is based on Ring Learning  
with Error

## Kyber

Kyber is based on Modular Learning  
with Error

# New Hope RLWE

4	1	11
2	4	1
12	2	4
9	12	2
11	9	12
1	11	9

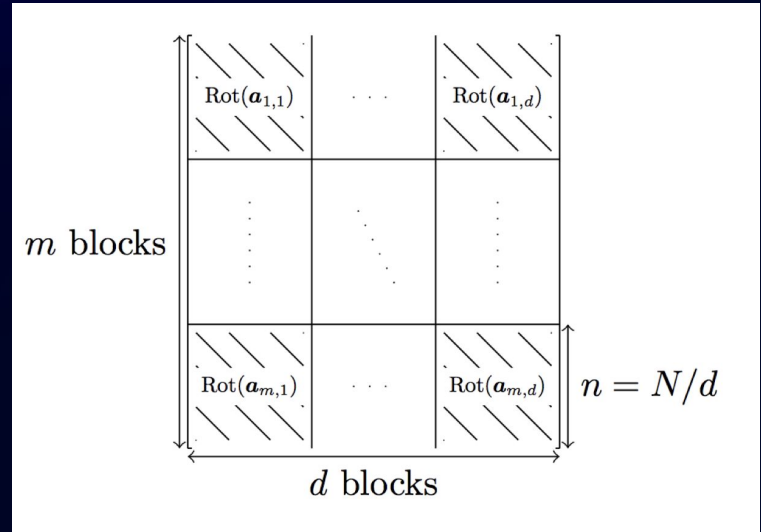
Each row is the cyclic shift of the row above

Wrapping rule is  $x$  maps to  $-x \bmod q$

$q$ : Random prime number

So we need to just send the first row

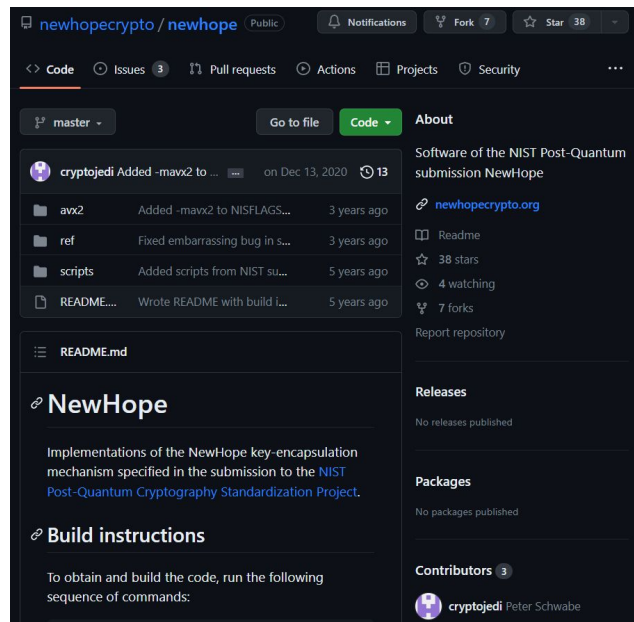
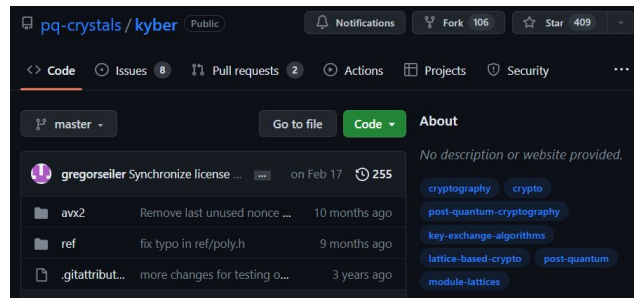
# Kyber MLWE



Every matrix entry is polynomial in  $\mathbb{Z}_q[x]/(x^n + 1)$

# Implementation

- Submissions to NIST's challenge are Open-Source, on GitHub
  - Each contain /ref folder to run tests on any hardware
  - Over a dozen measured metrics based on clock cycle

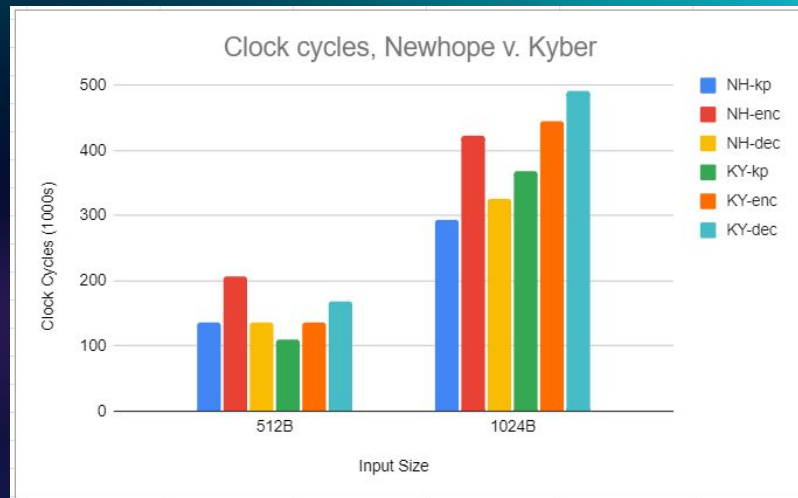
The NIST logo is displayed in a large, bold, black font on a white background. The letters are stylized and blocky.A screenshot of the GitHub repository page for 'newhopecrypto/newhope'. The repository is public and has 7 forks and 38 stars. The main content area shows a commit history with entries for 'avx2', 'ref', 'scripts', and 'README...'. The README section is visible, titled 'NewHope', and describes it as 'Implementations of the NewHope key-encapsulation mechanism specified in the submission to the NIST Post-Quantum Cryptography Standardization Project.' There are also sections for 'Build instructions' and 'Contributors'.A screenshot of the GitHub repository page for 'pq-crystals/kyber'. The repository is public and has 106 forks and 409 stars. The main content area shows a commit history with entries for 'avx2', 'ref', and '.gitattrib...'. The README section is visible, titled 'kyber', and describes it as 'Implementations of the Kyber key-encapsulation mechanism specified in the submission to the NIST Post-Quantum Cryptography Standardization Project.' There are also sections for 'Build instructions' and 'Contributors'.

# Findings

- Analysis on Cipher Size
  - Kyber has smaller ciphers at same size

Input Size (Bytes)	NewHope	Kyber
512	1104	768
1024	2184	1568

- Analysis on Clock Cycles
  - Different input sizes yield different optimal algorithms





# Conclusion

- Quantum computers can crack RSA encryption using Shor's algorithm in polynomial time.
- The security of lattice based cryptography algorithms is based on the hardness of the LWE problem which is in turn based on the hardness of the SVP problem. This makes lattice based algorithms an interesting candidate for many PQC applications.
- NewHope and Kyber are two lattice based algorithms which are based on LWE problem. NewHope takes longer time for encryption as compared to Kyber, but lesser time for decryption.

**Thank You**