Quantum Computing Project Report 2

PL3001 - Quantum Computing

Eshwar SK, Tanya Sravan

Project Report 2

23^{rd} April 2024

Contents

1 What is Post Quantum Cryptography

Post quantum cryptography(PQC) is a response to the potential threat posed by the emergence of a large fault free quantum computer with the ability to run algorithms like Shor's. The main aim of PQC is to protect classical systems from attacks using mathematically hard problems to create classical encryption keys.

2 Why is PQC Relevant

Traditional cryptography relies on the difficulty of problems like integer factorization (IF) and unstructured search. However, quantum algorithms exploit quantum mechanics to achieve significant speedups, posing a threat to existing cryptosystems. Two such algorithms are the Shor's algorithm and the Grover's algorithm.

2.1 Grover's Algorithm

Grover's algorithm tackles unstructured search, aiming to find a specific element ("marked state") within an unsorted database. Classically, this requires checking each element (bit string) with a time complexity of $O(N)$.

Grover's power lies in utilizing superposition. It prepares a quantum register of n qubits in a uniform superposition state:

Grover's Algorithm

State	Representation $\|\rangle$
Superposition	$(\|0\rangle + \|1\rangle) \otimes n$

Here, n represents the number of qubits in the quantum register, and $\otimes$ denotes the tensor product representing the superposition across all qubits.

This state represents all possible bit strings simultaneously. Subsequently, Grover applies a sequence of rotations (Oracle Operator and Grover Diffusion Operator) that amplifies the probability of the marked state while suppressing others.

Oracle Operator (O): This operator differentiates the marked state. Mathematically, for an input state $|x\rangle$:

$$O|x\rangle = \begin{cases} |x\rangle & \text{if } f(x) = 1 \text{ (marked)} \\ -|x\rangle & \text{otherwise} \end{cases}$$

where $f(x)$ is the function used to identify the marked element.

Grover Diffusion Operator (D): This operator reflects the state around the average, amplifying the marked state. It involves the Hadamard transform (H) and a reflection about the uniform superposition:

$$D = 2|\psi\rangle\langle\psi| - I$$

where:

- $|\psi\rangle$ is the current state of the quantum register.
- I is the identity operator.

2.2 Shor's Algorithm

Shor's algorithm tackles integer factorization, a critical problem for public-key cryptography (RSA). Factoring large numbers classically is computationally expensive, but Shor's algorithm achieves polynomial time complexity (roughly $O(log^3 N)$).

Shor's algorithm leverages superposition to create a period finding subroutine. It starts with an integer N and a smaller integer a coprime to N. It then puts a superposition of all possible values (0 to $N-1$) in a quantum register and applies a modular exponentiation operation involving a and N.

The resulting state encodes the period (p) with which a^p repeats modulo N. This period finding is achieved using the Quantum Fourier Transform (QFT), which exploits the relationship between addition and multiplication modulo N.

This algorithm utilizes the QFT to find the period in the modular exponentiation. The QFT operates on a superposition of states and relates addition and multiplication modulo N. Here's a simplified representation (the actual QFT involves complex number manipulations):

$$\text{QFT}\left(\frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|j\rangle\right) = \frac{1}{\sqrt{N}}\sum_{j=0}^{N-1}|j\rangle \exp\left(\frac{2\pi i j k}{N}\right)$$

Here:

- N is the integer to be factored.
- k represents an index iterating through possible values (0 to $N-1$).

By finding the period (p), Shor's algorithm can efficiently compute the factors of N. This poses a significant threat to RSA encryption, as breaking the public key (based on large factorized numbers) allows decryption of messages.

3 Lattice Based Cryptography

A lattice can be defined as the linear combination of n independent vectors, $L = z_1 b_1 + z_2 b_2$. What makes lattice problems hard is that it is modeled after the closest vector problem. Suppose we have a lattice and a point. The hard problem is finding a point on the lattice that is the

closest point to the given point.

When the vectors b_1 and b_2 have an angle of around 90^o in between them then finding the closest point is not hard however when the angle between the vectors are close to 0 then finding the closest point is hard. In lattice based cryptography the primary method of encryption is learning with errors (LWE). There are 3 different systems of LWE, the base model LWE, then the one with further optimization Ring-LWE and the final one optimized for increasing security without changing the dimension size of the matrices, Module LWE.

Here is a breakdown on how encryption and decryption using LWE works.
Suppose Alice and Bob were trying to communicate with each other. First Bob generates a public key and a private key. Bob generates 3 matrices, A, S and ϵ. He then does $AS + \epsilon = t$ to get t. Now S is his private key and the concatenation of A, t is the public key, we'll call this P. Now Alice uses Bob's public key and performs RP and adds some random vectors e_1 and e_2 with the message m to provide the encrypted text of u and v.

u and v are sent to Bob to recompute the message m

$$
\begin{aligned}
&v = r \times t + e_2 + m \\
&v = r\,(As + e) + e_2 + m \\
&v = r\,(As) + e' + m \quad \text{(here } e' \text{ is some small noise)} \\
&us = (rA + e_1)S \\
&= rAS + e' = v - m
\end{aligned}
$$

So now Bob can decrypt to obtain $v - us = m + e'$ and so the message can be obtained by removing the noise somehow.

So why is LWE Hard. This is because for the ϵ matrix. Say for example ϵ has m coefficients and is binary (typically in the sizes of 256 upwards). This means that one would have to do 2^m trials to find the values of t. Thus making this whole system hard.

4 Crystal-Kyber

Kyber is a lattice-based cryptosystem, specifically inspired by the Learning With Errors (LWE) problem in module lattices (MLWE) (The breakdown on the different types of LWE is on the repo). One of its primary design purposes is to establish a shared secret between two communicating parties (which then continue using symmetric cryptography) in a secure way without an attacker being able to decrypt it (and thus future communications between the parties).

Kyber offers three sets of parameters, named Kyber512, Kyber768, and Kyber1024, with the aim of achieving 128, 192, and 256 bit security levels, respectively. An important distinction between Kyber's design and other Ring-LWE encryption schemes is that these schemes need to change the lattice dimensions (n), and modulus (q), to increase the security levels, however Kyber fixes these values and achieves a higher security by increasing a scaling parameter, k, to in-turn attain a higher lattice dimension. This means operations within Kyber, such as multiplication, can all be done using the same fundamental operations, using the same parameters, except that more repetitions of these same operations are required to achieve a higher level of security.

The hardness of this algorithm comes from the fact that t is not simply a matrix product but also offset with a random noise vector e. figuring out the value of the secret key s by just looking at (A, t) is not trivial. (If there were no noise offset, one can use row reduction and Gaussian elimination tricks to solve for s.).

More specifically, solving for s from (A, t) can be reduced to solving an instance of the problem of Module Learning With Errors (M-LWE), which is believed to be hard even for quantum computers (similar proof as why is LWE hard shown above). Kyber gains quantum-resistance - by linking its core underlying public-key scheme with a mathematically hard problem (M-LWE) that cannot be efficiently solved by a quantum computer.

A code implementation is also provided with analysis of the code on the git repo under Kyber implementation.

5 Code Based Cryptography

Code based cryptography is heavily based on the concept of error correction codes. Digital media is often exposed to memory corruption and so we use error correction methods where we store k bits in n bits, meaning the redundancy is $n - k$. Checksums are a basic illustration of an error-detection code. The objective is to maximize the likelihood of data transmission accuracy while decreasing the volume of additional information added. If good codes are used then one can correct multiple errors.

The key concept of Code based cryptography is the use of linear codes, where we have a binary code, c, with length n and dimension k. The core concepts of Code based cryptography revolves around the generator matrix. When we apply the generator matrix to the code we are able to increase the distance between the valid codes making it easier to correct in incase there is an error. The parity check matrix then tells us on the receiving end whether or not there is an error and tells us what errors to fix.

Further notes available on the git repo under CBC notes

6 Classic McEliece Cryptography

The McEliece cryptosystem is based on the concept of error-correcting codes. The McEliece system offers a unique approach to encryption by leveraging the difficulty of decoding in linear binary codes (further proof to this is provided in the git repo under CBC efficiency proof). McEliece offers a robust and practical approach to encryption. Its security is based on well-established mathematical principles, and it has withstood decades of cryptanalysis without significant vulnerabilities being discovered. This system is also versatile and adaptable to various settings and security requirements. It can be used for secure communications in a wide range of applications, from secure messaging to protecting critical infrastructure.

Key Generation

Let C be a random binary Goppa code of length n and dimension k. Generate a random invertible $k \times k$ binary matrix S. Compute the public key as $P = S^{-1} \times G \times P_{\text{pub}}$, where G is the generator matrix of C and P_{pub} is a random permutation matrix. The private key is S.

Encryption

To encrypt a message m, convert it into a binary vector of length k. Generate a random binary vector r of length $n-k$. Compute the ciphertext as $c = m \times P + r$.

Decryption

To decrypt c, compute $c \times S$ to obtain m.

Example

Let's consider $n = 7$ and $k = 4$ for simplicity.

Key Generation

Assume C and G are:

$$C = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Let P_{pub} be a random permutation matrix.

Encryption

Let the message $m = [1, 0, 1, 0]$. Generate a random error vector $r = [1, 0, 1, 1, 0, 1]$. Compute the ciphertext:

$$c = m \times P + r$$

Decryption

To decrypt c, compute $c \times S$ to obtain m.

Now to explore the complexity of the algorithm

Selecting the Goppa code: This involves choosing a Goppa polynomial $g(z)$ of degree t over F_{2^m}, and n distinct field elements $\alpha_1, \ldots, \alpha_n$. The complexity is $O(nt)$.

Computing the public key: This involves computing the $k \times n$ public key matrix $\hat{G} = SGP$, where S is a random $k \times k$ invertible matrix, G is the generator matrix of the Goppa code, and P is a random $n \times n$ permutation matrix. The complexity is $O(kn)$.

Encoding the message: This involves multiplying the message vector m of length k by the public key matrix $\hat{G}$. The complexity is $O(kn)$.

Adding the error vector: This involves adding a random error vector e of weight t to the encoded message. The complexity is $O(n)$.

Decoding the ciphertext: This involves using the Patterson algorithm to decode the received vector c and recover the error vector e. The complexity is $O(n \log^2 n)$.

Removing the error: This involves subtracting the recovered error vector e from the received vector c to obtain the encoded message. The complexity is $O(n)$.

Recovering the message: This involves multiplying the recovered encoded message by the inverse of the matrix S. The complexity is $O(k^2)$.

The key generation and encryption algorithms have a complexity that is quadratic in the code length n, while the decryption algorithm has a complexity that is quasilinear in n. This makes the Classic McEliece cryptosystem efficient for practical applications, especially for decryption which is the most computationally intensive operation.

The security of the system relies on the hardness of decoding a random linear code, which is an NP-hard problem. The best known attacks have a complexity that is exponential in the code length n, making the system secure for appropriately chosen parameters.

Further explanation of the Goppa Codes follows:

Goppa codes are error-correcting codes defined by an irreducible polynomial $g(z)$ of degree t over F_{2^m}, and a set of n distinct elements $\alpha_1, \ldots, \alpha_n$ from F_{2^m}. The code consists of all vectors $c = (c_1, c_2, \ldots, c_n)$ such that $\sum_{i=1}^{n} \frac{c_i}{z - \alpha_i} \equiv 0 \pmod{g(z)}$.

Goppa codes have a minimum distance of at least $2t + 1$, allowing them to correct up to t errors.

6.1 Patterson's Algorithm

Patterson's algorithm is an efficient decoding algorithm for binary Goppa codes. It works as follows:

1. Compute the syndrome $s(x) = \sum_{i=0}^{n-1} \frac{v_i}{x-\alpha_i} \pmod{g(x)}$, where $v = (v_0, v_1, \ldots, v_{n-1})$ is the received vector.

2. Calculate $v(x) = \sqrt{s(x)^{-1} - x} \pmod{g(x)}$, assuming $s(x) \neq 0$.

3. Use the Extended Euclidean Algorithm (EEA) to find polynomials $a(x)$ and $b(x)$ such that $a(x) = b(x)v(x) \pmod{g(x)}$.

4. Compute the error locator polynomial $\sigma(x) = a(x) + xb(x)^2$, which should be equal to $\prod_{i \in B}(x - \alpha_i)$, where $B = \{i : e_i \neq 0\}$ and $e = (e_0, e_1, \ldots, e_{n-1})$ is the error vector. 5. Find the roots of $\sigma(x)$, which correspond to the error positions.

6.2 Security against Quantum Attacks

The security of the classic McEliece cryptosystem relies on the hardness of decoding a general linear code, which is an NP-hard problem. Even with the advent of quantum computers, the best known quantum algorithms, such as Shor's algorithm, do not provide a significant speedup for this problem.

Shor's algorithm is designed to solve the integer factorization and discrete logarithm problems, which are the basis for the security of RSA and elliptic curve cryptography. However, it does not directly apply to the decoding problem in code-based cryptography.

The combination of efficient decoding algorithms like Patterson's algorithm and the inherent hardness of the decoding problem makes the classic McEliece cryptosystem a promising candidate for post-quantum cryptography.

7 Kyber implementation breakdown

We have implemented Kyber for 3 security levels kyber-512, kyber-768, kyber-1024. This is controlled by parameters like n,k,q,eta_1 and eta_2, du and dv. here n is the degree of polynomials, k is the number of polynomials in a vector, q is the modulus, eta_1 and eta_2 are the error parameters, and finally du and dv which act as compression parameters. By this we mean, Kyber uses polynomial arithmetic, and the polynomials involved have coefficients that can take large values. Directly transmitting these polynomials would result in large data sizes, which is not efficient for practical implementations. To address this, Kyber uses compression techniques to reduce the size of the transmitted data. This compression works by mapping the coefficients of the polynomials to a smaller range. This is done by reducing the precision of the coefficients, effectively quantizing them to fewer bits.

Then we set up functions for key generation encryption and decryption, like described above.

Some key points to note is that when we are generating keys we use a separate script that implements a deterministic random bit generator using the AES-256 cipher. This is used instead of the os.urandom to achieve a higher level of control, predictability. This is because DRBG generates the same sequence of random numbers given the same seed, making it useful for scenarios where reproducibility is essential. This is better that the inbuilt random call because this is optimised for ctryptographic implementations.

Apart from this we have implemented scripts to perform polynomial arithmetic and also matrix math.

We performed Known Answer test for all the 3 kyber security levels and our implementation pass all the KAT tests. These tests were taken from a reference repository.

Some basic bench marking to see the time taken for the algorithm to run. For Kyber 512 it takes approximately 10.8s for keyGen, 12.3s for Encryption and 18.2 secs for decryption, for keber 768 it took 12.4,16.01,27.93 and finally 18.14,24.34,37.22 for Kyber 1024. These times were recoreded on an intel i5 processor. Further speed ups can be observed with more efficient implementations of polynomial and matrix operations.

Comparison of Kyber and Classic McEliece

Pros and Cons of Kyber

Pros

- Uses quasi-cyclic (QC) or quasi-dyadic (QD) codes, which have a structured generator matrix. This allows for a more compact representation of the public key, reducing its size by up to 50% compared to Classic McEliece.
- The structured nature of the codes used in Kyber enables more efficient key generation and encapsulation algorithms, as the structured generator matrix can be represented compactly and manipulated efficiently.
- Kyber has a smaller ciphertext expansion factor compared to Classic McEliece, meaning the ciphertext is closer in size to the plaintext.

Cons

- The use of structured codes in Kyber introduces potential security risks. The structure of the codes may leak information about the secret key, potentially leading to attacks. This is a concern because the security of code-based cryptography relies on the hardness of decoding a random linear code, which is an NP-hard problem.
- Kyber requires careful selection of security parameters to balance efficiency and security. The choice of code parameters, such as the code length and dimension, affects both the efficiency and the security of the scheme.
- The security of Kyber is less well-studied compared to Classic McEliece, as it is a relatively newer scheme. Classic McEliece has been studied extensively since its introduction in 1978, and its security has withstood cryptanalysis for over 40 years.

Pros and Cons of Classic McEliece

Pros

- Uses random binary Goppa codes, which are a well-studied class of error-correcting codes. The use of random codes provides a higher level of security compared to structured codes, as they do not leak information about the secret key.
- Classic McEliece has a strong security track record, with the original 1978 version remaining unbroken. This makes it a more conservative choice for applications that prioritize security over efficiency.
- The decryption algorithm in Classic McEliece involves only error correction, which is an efficient operation. This results in efficient decryption compared to other code-based cryptosystems.

Cons

- Classic McEliece has larger public key sizes compared to Kyber due to the use of random codes. The public key size grows linearly with the code length, which is typically larger than the code length used in Kyber.
- Key generation and encapsulation in Classic McEliece are more computationally intensive compared to Kyber, especially at higher security levels. This is because the key generation and encapsulation algorithms involve operations on larger matrices and vectors.

Use Cases

- Kyber is suitable for applications where public key size is a critical constraint, such as in resource-constrained devices or when minimizing communication overhead is important. However, the use of Kyber should be carefully considered due to the potential security risks associated with structured codes.
- Classic McEliece is a good choice for applications that prioritize security over efficiency, such as in high-security communication systems or when protecting sensitive data. Its strong security track record and the use of random codes make it a more conservative choice.

Summary

In summary, Kyber offers a trade-off between security and efficiency, while Classic McEliece prioritizes security over efficiency. The choice between the two depends on the specific requirements and constraints of the application, with Kyber being more suitable for applications where public key size is critical and Classic McEliece being more suitable for applications that prioritize security.

Post Quantum Cryptography offers a variety of techniques and the choice of algorithm for encryption/decryption usually comes down to a tradeoff between security requirement, available compute, possible weakness attacks and the postulated fundamental np hardness degree of the problem. there exist multiple techniques apart from the areas explored here like multi-variate, energy based, etc. all these lead us to believe that we aren't quite doomed as soon as quantum algorithms like shor's algorithm becomes a reality.

GitHub Repo Link