RV COLLEGE OF ENGINEERING
Bengaluru-560 059

REPORT ON

EXPERIENTIAL LEARNING / PROJECT BASED LEARNING

ACY 2023-24

THEME

Quantum Mechanics

Title of the Project

Quantum Encryption - Achieving Unprecedented Security with **PreP Mail**

Students Group

SL. No.	USN	Name	Prog.
1	1RV23CS087	ESHITHA CHOWDARY NATTEM	Computer Science and Engineering
2	1RV23CS081	D RAKSHITHA	Computer Science and Engineering
3	1RV23IS089	PRARTHANA R	Information Science and Engineering
4	1RV23AS042	PRATIKSHA M GADDIGIMATH	Aerospace Engineering

Project Evaluators

SL. No.	Name of the Evaluators (Enter the Names in Capitals)
1.	Dr Abhay A Deshpande
2.	Dr Tribikram Gupta
3.	Prof P L Rajashekar
4.	Dr Mohana
5.	

Abstract

In the wake of advancing quantum computing capabilities, the security of traditional cryptographic algorithms faces unprecedented threats. To address this emerging challenge, we propose the development of a secure email client leveraging the post-quantum cryptographic algorithm, Dilithium, alongside the liboqs library. Dilithium, a lattice-based algorithm, is designed to provide robust security against quantum attacks, ensuring the confidentiality, integrity, and authenticity of email communications.

Our email client integrates Dilithium for digital signatures and liboqs, a comprehensive library for quantum-safe cryptographic algorithms, to fortify the encryption and verification processes. The client features seamless encryption and decryption of emails, secure key exchange mechanisms, and rigorous authentication protocols, all resistant to the computational power of quantum adversaries.

By employing Dilithium and liboqs, our solution not only enhances security against future quantum threats but also maintains compatibility with current cryptographic standards. This project aims to deliver a user-friendly, highly secure email client that can serve as a foundational tool for organizations and individuals seeking to protect their communications in a post-quantum world.

Introduction

The advent of quantum computing heralds a new era of computational capabilities, which, while promising significant advancements across various fields, also poses a substantial threat to current cryptographic systems. Traditional public key cryptographic algorithms, such as RSA and Elliptic Curve Cryptography (ECC), are predicated on mathematical problems that are computationally infeasible to solve with classical computers. However, quantum computers, leveraging principles such as superposition and entanglement, have the potential to solve these problems efficiently using algorithms like Shor's algorithm, thereby compromising the security of classical cryptographic systems.

We are surrounded by devices collecting and transmitting our private information, which makes secure and reliable communication extremely critical. When a client creates a connection to a secure server, or some other secure communication protocol, different cryptographic schemes

are used. We can observe such cryptographic operations, for instance, in email communication. Essential part of securing digital systems with public key algorithms are digital signatures. They provide integrity and authentication protection by giving the receiver a reason to believe that the message was sent by the claimed sender. When a message is signed digitally, the signature is sent as a separate document alongside the original message. The recipient then obtains both the message and the signature. They can then verify the authenticity using a verification technique by combining the sender's public key, signature and message. In email communication, the original plain text message undergoes a transformation known as a message digest. This is done, using a cryptographic hash algorithm, which generates a string of digits using a one-way hashing function. The purpose of message digests is to protect the integrity of data by detecting any modifications or alterations made to any part of a message. Afterwards the transformed message is cryptographically signed using the selected algorithm.

It is clear that deploying a new cryptographic system, such as PQC, on widely-used mobile devices incurs physical and time penalties. These include the latency and energy consumption, as well as heat created by the generation and exchange of keys, verification of signatures, and hashing. Heat can have a negative impact on the lifespan of a battery and is generally inconvenient for the user. Therefore, an enormous part of the scientific research on cryptography focuses on the implementation aspects with minimum real-world drawbacks, and PQC implementations are also increasingly studied to reach this target. Of course, this principle applies not only to emails but also to the most common forms of secure information exchange on the internet. Hence, the primary focus of this paper is to implement the Dilithium post-quantum cryptography (PQC) signature scheme on the Android platform and evaluate its performance. Additionally, this work provides instructions and example implementations of the current state of the Dilithium signature scheme.

Understanding Quantum Attacks

Introduction to Quantum Computing

Quantum computing represents a fundamental shift in computational power and efficiency. Unlike classical computers that use bits to process information in binary states (0 and 1), quantum computers use quantum bits or qubits, which can exist in multiple states simultaneously due to the principles of superposition and entanglement. This enables quantum computers to perform complex calculations at unprecedented speeds.

The Threat of Quantum Attacks

Quantum attacks exploit the capabilities of quantum computers to break cryptographic algorithms that are currently considered secure by classical standards. The primary concern is that quantum computers can solve certain mathematical problems much more efficiently than classical computers, thereby undermining the security foundations of widely used cryptographic schemes.

Types of Quantum Attacks

1. **Shor's Algorithm:**
    - Overview: Developed by Peter Shor in 1994, this algorithm efficiently solves the integer factorization problem and the discrete logarithm problem, both of which underpin the security of many public-key cryptosystems such as RSA, DSA, and ECC.
    - Impact: Shor's algorithm can factorize large integers in polynomial time, rendering RSA and other related cryptographic systems vulnerable to rapid decryption by quantum computers.
2. **Grover's Algorithm:**
    - Overview: Proposed by Lov Grover in 1996, Grover's algorithm provides a quadratic speedup for unstructured search problems. It can significantly reduce the time required to brute-force search through cryptographic keys.
    - Impact: While Grover’s algorithm does not completely break symmetric-key cryptography, it necessitates doubling the key sizes to maintain current security levels. For example, a 128-bit key would need to be increased to 256 bits to retain its security against quantum attacks.

Implications for Current Cryptographic Systems

- RSA, DSA, and ECC: These systems are particularly vulnerable to Shor's algorithm, which can break their security by efficiently solving the underlying mathematical problems.
- Symmetric Key Cryptography: Although more resilient, symmetric key cryptography systems like AES are still affected by Grover's algorithm, requiring longer key lengths to remain secure.
- Hash Functions: Similar to symmetric keys, hash functions need to adapt to longer output lengths to withstand quantum attacks.

Conclusion

The advent of quantum computing necessitates a proactive approach to securing digital communications. By understanding the nature and implications of quantum attacks, we can better appreciate the importance of post-quantum cryptographic algorithms, such as Dilithium, in safeguarding sensitive information against future threats. This project underscores the need for integrating quantum-resistant cryptographic solutions to ensure the long-term security and privacy of email communications in a post-quantum world.

Post quantum Cryptography

We know that:

- cryptography like RSA is effective against brute force attacks by classical computers
- quantum computers with enough qubits can make breaking RSA exponentially faster
- current quantum computers don't give a practical advantage
- to expect quantum computers in the future to outpace classical computers

Firstly, Shor's algorithm generally only applies to asymmetric encryption, which is used for things like HTTPS. Symmetric encryption like AES, where you use the same secret to encrypt and decrypt, is expected to be pretty safe even from quantum computers. This is because Grover's algorithm only provides a quadratic speedup, which can't keep up with doubling the key length. Secondly, now that cryptographers are taking quantum computing and algorithms like Shor's into account, a whole new field of cryptography has merged: Post Quantum Cryptography or PQC. These are algorithms that have been specially designed to be resistant against quantum attacks. The goal of post-quantum cryptography (also called quantum-resistant cryptography) is to develop cryptographic systems that are secure against both quantum and classical computers, and can interoperate with existing communications protocols and networks.

The major classes of post quantum cryptography algorithms are:

• Code-based cryptography uses error correcting codes and offers a conservative approach for public key encryption/key encapsulation, as it is based on a well-studied problem for over 4 decades. This class of algorithms use large keys and some attempts at reducing their key size have made these algorithms vulnerable to attacks. Researchers proposed techniques to reduce the key size without compromising the security.

• Multivariate polynomial cryptography relies on the difficulty of solving the multivariate polynomial (MVP) algorithm over finite fields. Solutions of MVP problems are NP-hard over any field and NP-complete even if all equations are quadratic over GF(2). MVPs are preferred as signature schemes as they offer the shortest signatures. However, some schemes are broken.

• Hash-based digital signatures resist quantum-computer attacks. These schemes are based on the security properties of the underlying cryptographic hash functions (collision resistance and second pre-image resistance).

• Lattice-based cryptography algorithms offer the best performance, but are the least conservative among all. Lattice cryptography builds on the hardness of the short- est vector problem (SVP) which entails approximating the minimal Euclidean length of a lattice vector. Even with a quantum computer SVP is shown to be polynomial in n. Other lattice cryptography algorithms are based on Short Integer Solutions (SIS). These are secure in the average case if the SVP is hard in the worst-case.
Let's see if we can simplify it a bit…

Chess analogy

Let's reduce our lattice to 2 dimensions and visualize it as an infinite chess board. We have a knight in the center. How the knight moves is the "vector". Normal knights can move 2 squares in one direction and 1 in the other. We can represent as the vectors (1, 2) and (2, 1) for moving ↗ and negate them for ↙. Knights can also move with the vectors (-1, 2) and (-2, 1), which allows them to go ↖ and ↘.

For our lattice-based cryptography, we'll have a special knight that can move either (1, 4), (7, 5) or (-2, 3). We'll call these 3 moves "hop", "skip" and "jump", respectively.

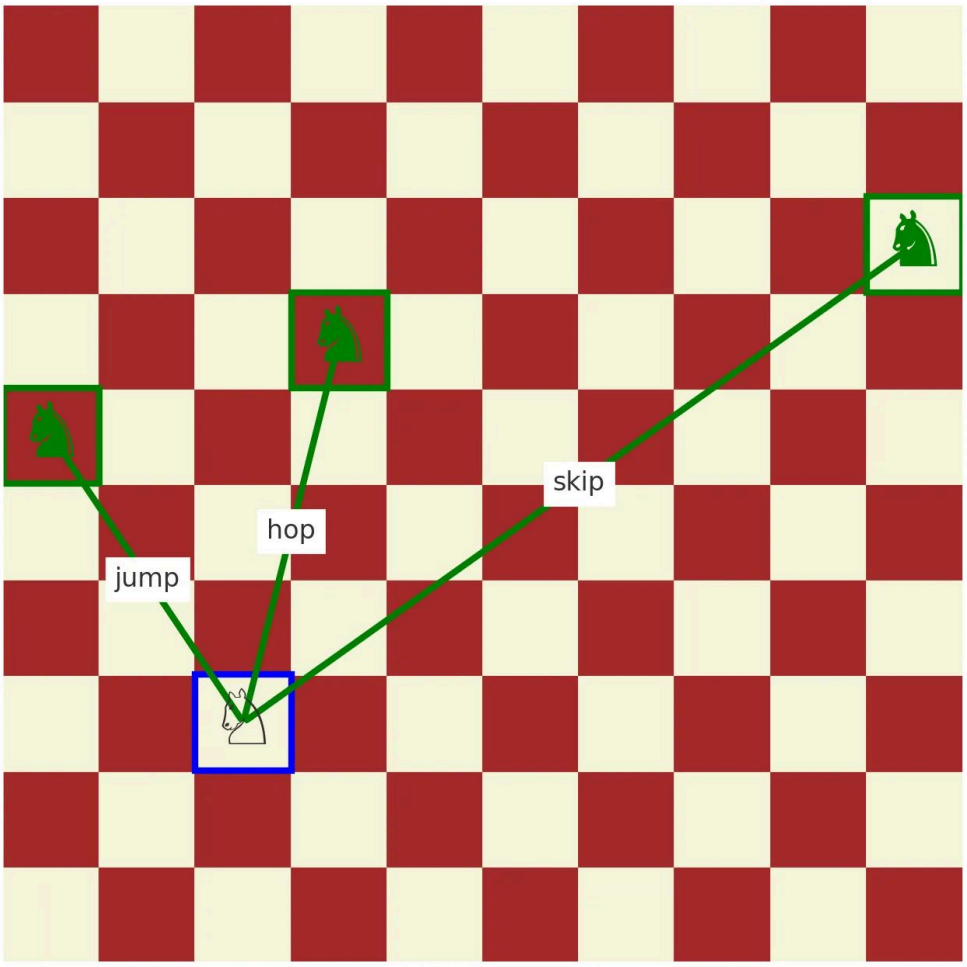


Figure 1: Our special knight can hop (1, 4), skip (7, 5) or jump (-2, 3)

Now, the challenge is to find the combination of knight moves that will get the knight as close as possible to the starting point but not exactly on it. As you can imagine, this requires trying lots of different combinations. It turns out that with these three moves, our knight can get to one square below the starting point: (0, -1). It does this with 1 skip, 2 jumps and then hopping backwards 3 times. Fortunately our lattice is infinite, so the knight doesn't fall off the board.

That's the basis of lattice-based cryptography. Though our cryptographic knight can move in hundreds of ways (vectors) and our chess board also has hundreds of dimensions.

In practice, lattice-based cryptographic algorithms add additional layers of complexity. To generate a public key, I'd have to introduce a controlled amount of randomness into the description of the knight's possible moves. This adds a layer of obfuscation that's deliberate and calculable. During encryption, the knight would be given a mathematically precise nudge on the board.

• Other cryptographic methods include evaluating isogenies on super singular elliptic curves. Shor's algorithm is ineffective against evaluating isogenies.

PQC	The hard problem	Sign	KEM	Total
Lattice	Find shortest vector, closest vector	5 (3)	23 (9)	28 (12)
Code	Decode random linear code	3 (0)	17 (7)	20 (7)
Multivariate	Solve multivariate quadratic equations	8 (4)	2 (0)	10 (4)
Hash	Second pre-image resistance of hash function	3 (2)	0 (0)	3 (2)
Isogeny	Find isogeny map between elliptic curves with same number of points	0 (0)	1 (1)	1 (1)
Other	-	2 (0)	5 (0)	7 (0)
Total	-	**21 (9)**	**48 (17)**	**69 (26)**

Table 1: A summary of PQC digital signature and key encapsulation submissions and the underlying hard mathematical problems. We show the number of algorithms for Round 1 and Round 2 (within braces) NIST PQC evaluation.

LWE(Learn with errors) and SIS(Short integer solutions)

LWE (Learning With Errors) is a cryptographic problem where one aims to recover a secret vector from linear equations that have been perturbed by some random noise. In simpler terms, it's like trying to find a hidden pattern in a bunch of random and noisy information. The challenge is to extract the secret message from this noisy data without getting confused by the randomness.

Given a matrix A and a vector b, the goal is to find a vector s such that the dot product of A and s is close to b, with some added noise e. Mathematically, it can be represented as finding s such that

$$A * s \approx b + e.$$

The LWE problem involves noisy linear equations of the form ->

$$14s_1 + 15s_2 + 5s_3 + 2s_4 \approx 8 \pmod{17}$$

$$13s_1 + 14s_2 + 14s_3 + 6s_4 \approx 16 \pmod{17}$$

$$6s_1 + 10s_2 + 13s_3 + 1s_4 \approx 3 \pmod{17}$$

$$10s_1 + 4s_2 + 12s_3 + 16s_4 \approx 12 \pmod{17}$$

$$9s_1 + 5s_2 + 9s_3 + 6s_4 \approx 9 \pmod{17}$$

$$3s_1 + 6s_2 + 4s_3 + 5s_4 \approx 16 \pmod{17}$$

...

$$6s_1 + 7s_2 + 16s_3 + 2s_4 \approx 3 \pmod{17}$$

Random equations are generated by multiplying the matrix with the secret vector, adding a small error vector, and taking the result modulo to obtain the vector. The challenge lies in recovering the secret vector from the noisy equations, where the error makes it difficult to directly solve for. By analysing the distribution of the noisy vectors, one can statistically infer the secret vector using mathematical techniques and algorithms designed to handle the errors introduced in the equations.

It has practical applications in creating secure encryption schemes, digital signatures, and other cryptographic protocols that rely on the difficulty of decoding the secret vector from noisy equations. The mathematical complexity of LWE lies in efficiently solving systems of noisy linear equations to recover the secret information while accounting for the errors introduced in the process.

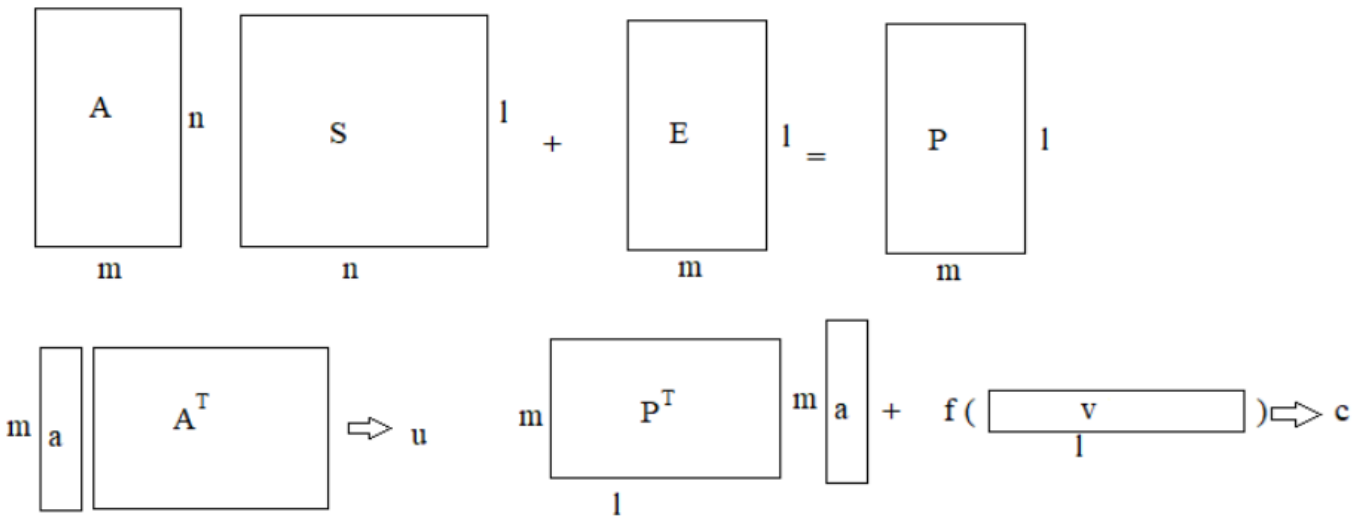


Figure 1

Figure 2 represents the LWE cryptosystem, a fundamental concept in post-quantum cryptography. It outlines the encryption and decryption processes involved in the LWE scheme. Encryption involves adding noise to the plaintext message and generating a ciphertext pair. Decryption determines the original message based on the ciphertext and the secret key. The LWE scheme is crucial for secure communication and is resilient against quantum attacks.

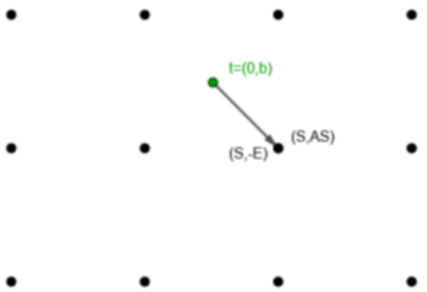


Figure 3

The LWE cryptographic scheme can be visualized, as in Figure 3, where a lattice is generated by the public matrix A and the lattice point (S, AS). As shown, the public point t = (0, B) is distinct from the lattice point (S, AS) and the vector (S, ”-E”) separates it from the point. With our transformation, we are changing this public point to improve both efficiency and security.

Decryption involves using the private key S and the ciphertext (u, c) to recover the plaintext message. The process entails computing c - S^T * u and then applying the inverse of the function f to obtain the original message. This decryption method ensures the secure retrieval of the plaintext from the ciphertext in the LWE scheme.

SIS (Short Integer Solution) is a cryptographic problem where the goal is to find short integer solutions to systems of linear equations modulo a large prime q.

The SIS problem is defined as finding short integer vectors x such that

$$Ax \equiv 0 \pmod{q}$$

where A is a matrix, x is a vector of integers, and q is a large prime number.

- **One-way Function:** SIS can be used to construct one-way functions, which are easy to compute but hard to invert. Given a matrix A, it is easy to compute $A\ x$ mod q for a short

vector x, but hard to find such an x given only A and the result. This property is fundamental to many cryptographic protocols.

- **Collision Resistant Hash:** Collision-resistant hash functions are those for which it is hard to find two distinct inputs that hash to the same output. SIS-based constructions can create such hash functions because finding two different short vectors that produce the same hash output modulo q is computationally difficult.
- **Signature:** Digital signature schemes can be built using the SIS problem. The signer uses a secret key to produce a short vector that acts as the signature, while the verifier checks the validity using the public key. The difficulty of finding valid signatures without the secret key relies on the hardness of the SIS problem.

The CRYSTALS-Dilithium digital signature algorithm leverages the LWE problem for generating public keys that are secure and the SIS problem for ensuring the integrity and validity of signatures. The combination of these two hard lattice problems provides a strong security foundation, making Dilithium a robust choice for post-quantum digital signatures.

Liboqs

What is the liboqs Library?

liboqs (Open Quantum Safe library) is an open-source C library designed to provide quantum-resistant cryptographic algorithms. It was developed by the Open Quantum Safe (OQS) project, which aims to create and integrate cryptographic algorithms that are secure against quantum computer attacks. Quantum computers pose a significant threat to traditional cryptographic systems, as they can efficiently solve problems like integer factorization and discrete logarithms, which are the foundations of many widely-used cryptographic protocols (e.g., RSA, ECC). liboqs addresses this threat by implementing post-quantum cryptographic

algorithms, which are believed to be resistant to attacks by both classical and quantum computers.

Features of liboqs

1. Algorithm Diversity: liboqs supports a wide variety of post-quantum cryptographic algorithms for key exchange and digital signatures. These include lattice-based, code-based, multivariate-quadratic-equations, hash-based, and other post-quantum schemes.

2. Modular Design: The library is designed to be modular, allowing easy integration of new algorithms and facilitating experimentation with different post-quantum cryptographic techniques.

3. Standardization and Interoperability: liboqs follows emerging standards from NIST's post-quantum cryptography project, ensuring that its implementations are aligned with global cryptographic standards.

4. Open Source: Being open-source, liboqs allows cryptographic researchers and practitioners to review, contribute, and improve its codebase, fostering collaboration and transparency.

How liboqs is Used in PreP-mail

The main purpose of PreP-Mail is to integrate post-quantum cryptography to ensure secure communication in the quantum computing era. Here's how liboqs is utilized in this client:

1. Key Exchange and Encryption:

- Integration: liboqs is integrated into the PreP-Mail client to provide post-quantum key exchange algorithms. This allows the email client to establish secure communication channels resistant to quantum attacks.
- Key Exchange Algorithms: The client uses key exchange algorithms like Kyber from liboqs to perform the initial key exchange between the email client and server. These algorithms are designed to be secure against quantum adversaries.
- Symmetric Encryption: Once the key exchange is complete, a symmetric key is derived and used for encrypting the actual email content. The use of post-quantum key exchange ensures that the symmetric key is secure.

2. Digital Signatures:

- Authentication: liboqs provides digital signature algorithms such as Dilithium, which are integrated into the PreP-Mail client for authenticating emails. This ensures that the sender of the email can be verified and the integrity of the email content can be guaranteed.

- Signature Verification: When an email is received, the client uses liboqs to verify the signature of the email. This ensures that the email has not been tampered with and that it originates from a legitimate sender.

3. User Interface and Workflow:

- Seamless Integration: The integration of liboqs into PreP-Mail is designed to be seamless, meaning that users do not need to have specialized knowledge of post-quantum cryptography to use the client. The cryptographic operations are handled in the background, providing a user-friendly experience.
- Backward Compatibility: The client maintains backward compatibility with traditional cryptographic algorithms, ensuring that it can communicate with servers and other clients that have not yet adopted post-quantum cryptography.

Benefits and Challenges

Benefits:

- Future-Proof Security: By integrating liboqs, PreP-Mail ensures that email communications are secure against future quantum computer attacks.
- Open Source and Transparent: liboqs is an open-source project, allowing for community scrutiny and continuous improvement.

Challenges:

- Performance: Post-quantum algorithms can be computationally intensive, potentially affecting the performance of the email client.
- Interoperability: Ensuring that the client can interoperate with other email systems that may not yet support post-quantum cryptography.

NIST Has Announced First Four Quantum-Resistant Cryptographic Algorithms

For general Encryption:

CRYSTALS-Kyber:

used when we access secure websites, NIST has selected the CRYSTALS-Kyber algorithm. Among its advantages are comparatively small encryption keys that two parties can exchange easily, as well as its speed of operation. Kyber is a key encapsulation mechanism (KEM) part of the CRYSTALS suite (Cryptographic Suite for Algebraic Lattices) developed in response to the need for post-quantum cryptographic standards. It is designed to resist attacks from quantum computers by leveraging the hardness of mathematical problems related to module lattices.Kyber is based on the Module Learning With Errors (Module-LWE) problem, a variant of the Learning With Errors (LWE) problem. The LWE problem involves distinguishing between a distribution of linear equations perturbed by small errors and a uniform random distribution. In Module-LWE, the problem is extended to modules over polynomial rings, which allows for greater efficiency and flexibility in cryptographic constructions.

Working:

1. Key Generation :

- Generate Matrices: Randomly generate a matrix A from a predefined distribution.
- Generate Secret and Error Vectors: Randomly generate secret vectors s and error vectors e from small distributions.
- Compute Public Key: Compute the public key as $b = A \cdot s + e$.

2. Encryption (Encapsulation):

- Generate Random Vector: Randomly generate a vector r.
- Compute Ciphertext: Compute the ciphertext components $u = A \cdot r + e_1$ and $v = b \cdot r + e_2 + m$, where m is the message.

3. Decryption (Decapsulation):

- Recover Message: Use the secret key s to compute $v - u \cdot s$ and recover the original message m by removing the added errors.

Kyber's security relies on the Fujisaki–Okamoto transform, which converts a chosen-plaintext attack (CPA)-secure scheme into a chosen-ciphertext attack (CCA)-secure one. This transformation ensures that even if an attacker can manipulate ciphertexts, they cannot gain useful information about the plaintext or the secret key.

Advantages of Kyber:

- Quantum Resistance : Kyber is designed to be secure against quantum computer attacks, which pose a significant threat to classical cryptographic schemes based on integer factorization or discrete logarithms.
- Efficiency : Kyber is optimized for high performance, with efficient polynomial arithmetic and small key and ciphertext sizes compared to other lattice-based schemes. This makes it suitable for practical implementations.
- Flexibility : The use of module lattices allows Kyber to be flexible in adjusting security levels by changing the dimensions of the modules. This adaptability makes it easier to scale security requirements as needed.
- Proven Security : The security of Kyber is based on well-studied mathematical problems with established hardness assumptions. This provides confidence in its robustness against various types of attacks.

Disadvantages of Kyber:

- Complexity : The mathematical foundation of Kyber, involving lattice problems and polynomial rings, is complex. This can make implementation and analysis more challenging compared to simpler cryptographic schemes
- Parameter Selection : Selecting appropriate parameters for the security and performance trade-offs can be non-trivial. Incorrect parameter choices can either weaken security or reduce efficiency.
- Resource Consumption : While Kyber is optimized, lattice-based schemes generally require more computational resources and memory compared to classical schemes, which might be a limitation for resource-constrained environments.
- Implementation Security : Ensuring secure implementations that resist side-channel attacks (e.g., timing attacks, power analysis) requires careful engineering and can be difficult, especially in hardware implementations.

For digital Signatures:

Dilithium:

Cryptography ensures the security and confidentiality of information through various techniques, such as encryption, decryption, and digital signatures. With the advent of quantum computers, classical cryptographic algorithms like RSA and ECC are at risk of being broken due to their vulnerability to quantum attacks. As a result, post-quantum cryptographic algorithms, which are resistant to quantum attacks, are being developed. One such algorithm is the Crystals Dilithium algorithm.The Crystals Dilithium algorithm is a lattice-based post-quantum

cryptographic algorithm designed for digital signatures. It is part of the ongoing NIST standardization process for post-quantum cryptography. The algorithm is based on the hardness of problems in lattice-based cryptography, specifically the Learning With Errors (LWE) problem

and its variants. The algorithm uses structured lattices to create a balance between security and efficiency.

How Dilithium Works:The Dilithium algorithm involves three main processes: key generation, signing, and verification.

1. Key Generation:

- Two keys are generated: a public key and a private key. The public key is used for verifying signatures, while the private key is used for signing messages.
- The algorithm relies on random sampling and certain mathematical structures to generate these keys.

2. Signing:

- A message is signed using the private key. The signing process involves hashing the message and then generating a signature using the private key and certain mathematical operations involving lattice structures.
- The signature ensures that the message has not been altered and confirms the identity of the sender.

3. Verification:

- The recipient of the signed message uses the sender's public key to verify the signature.
- The verification process involves checking the consistency of the signature with the public key and the message. If the signature is valid, it confirms that the message was indeed signed by the holder of the private key.

Advantages of Dilithium:

- Quantum Resistance : The primary advantage of the Dilithium algorithm is its resistance to quantum attacks. Unlike classical cryptographic algorithms, it is designed to withstand the computational power of quantum computers.
- Efficiency : Dilithium is designed to be efficient in terms of computational resources and memory usage. This makes it suitable for practical applications, even on devices with limited resources.
- Standardization : Being part of the NIST standardization process, Dilithium is undergoing rigorous evaluation and is on track to become a standardized algorithm for post-quantum cryptography.

Disadvantages of Dilithium:

- Implementation Complexity : Implementing lattice-based algorithms like Dilithium can be more complex compared to classical algorithms. This may require more sophisticated software and hardware designs.

- Larger Key Sizes : Lattice-based algorithms generally have larger key sizes compared to classical algorithms, which can pose challenges in terms of storage and transmission.
- Relatively New : As a relatively new field, lattice-based cryptography and algorithms like Dilithium are still under extensive study. This means there might be undiscovered vulnerabilities or improvements needed.

SPHINCS+:

SPHINCS+ (Stateless Practical Hash-based Incredibly Nice Cryptographic Signature) is a stateless hash-based signature framework designed to address the limitations of traditional hash-based signature schemes, particularly in terms of speed and signature size. SPHINCS+ is a part of the NIST Post-Quantum Cryptography (PQC).

How SPHINCS+ Works:

The core of SPHINCS+ is based on hash-based signatures, which involve the use of hash functions and Merkle trees to create secure digital signatures. Here's a high-level overview of its functioning:

1. Merkle Tree Structure: SPHINCS+ uses a hypertree structure, which is a tree of trees. The leaf nodes of the primary Merkle tree are themselves Merkle trees, and the leaf nodes of these secondary trees are key pairs for one-time signatures (OTS). This structure enhances the security and scalability of the algorithm.

2. Few-Time Signature (FTS) Scheme: To improve efficiency and reduce the risk of key reuse, SPHINCS+ employs a few-time signature scheme (FORS) at the bottom of the tree. This allows the same leaf node to be used a few times without compromising security.

3. Tweakable Hash Functions: SPHINCS+ introduces tweakable hash functions to unify the security analysis of hash-based signature schemes. These functions allow for additional input (tweaks) that enhance security and mitigate multi-target attacks.

4. Publicly Verifiable Index Selection: Instead of selecting leaf nodes at random, SPHINCS+ uses a publicly verifiable method to select indices, which helps in preventing attacks aimed at few-time signature schemes.

Advantages of SPHINCS+:

- Stateless Operation : Unlike many traditional hash-based signature schemes that are stateful and require tracking of all produced signatures to prevent key reuse, SPHINCS+ is stateless. This makes it more practical for real-world applications, especially in distributed systems and backup scenarios where maintaining state is challenging.
- Security : SPHINCS+ provides strong security guarantees against both classical and quantum attacks. The use of tweakable hash functions and FORS enhances its resistance to various attack vectors, including multi-target and preimage attacks .
- Flexibility : The SPHINCS+ framework offers a wide range of parameter options, allowing users to make application-specific trade-offs in terms of signature size, signing speed,

number of signatures, and desired security level. This flexibility makes SPHINCS+ adaptable to different use cases and platform constraints.

- Performance Improvements : Compared to its predecessor SPHINCS, SPHINCS+ achieves better performance in terms of speed and signature size by employing several optimizations and improvements in the underlying cryptographic constructions.

Disadvantages of SPHINCS+:

- Large Signature Sizes : Despite the improvements, SPHINCS+ signatures are still larger than those produced by traditional schemes like RSA or ECDSA. This can be a limitation in applications where bandwidth or storage is constrained.
- Complexity : The hypertree structure and the use of multiple layers of hash functions and signature schemes add complexity to the implementation and understanding of SPHINCS+. This might pose challenges for developers and cryptographers who need to implement or analyze the scheme.
- Computational Overhead : The security of SPHINCS+ comes at the cost of computational overhead. The operations involved in generating and verifying signatures can be computationally intensive, which may impact performance in resource-constrained environments.

SPHINCS+ represents a significant advancement in the field of hash-based digital signatures, particularly in the context of post-quantum cryptography. Its stateless design, strong security properties, and flexibility make it a promising candidate for future cryptographic standards. However, the trade-offs in terms of signature size and computational complexity need to be carefully considered when deploying SPHINCS+ in real-world applications

FALCON:

FALCON (Fast Fourier Lattice-based Compact Signatures over NTRU) is a post-quantum, lattice-based, hash-and-sign signature scheme. It was developed to provide security against quantum computer attacks, making it a candidate in the NIST Post-Quantum Cryptography Standardization process.

Key Generation

FALCON's key generation process involves creating a pair of keys: a secret key and a public key. The steps are as follows:

1. Polynomial Generation: Randomly generate polynomials fff and ggg from a discrete Gaussian distribution.

2. Solve NTRU Equation: Compute polynomials FFF and GGG such that they satisfy the NTRU equation fG−gF=qmod ϕfG - gF = q \mod \phifG−gF=qmodϕ, where ϕ\phiϕ is a monic polynomial, and qqq is a modulus prime number.

3. Matrix Construction and FFT: Construct a basis matrix from these polynomials and apply Fast Fourier Transform (FFT) to convert them into the frequency domain.
4. LDL Decomposition: Perform LDL decomposition on the transformed matrix to obtain a binary tree structure used in the signing process.
5. Key Pair Output: The public key hhh is computed as h=gf−1mod qh = gf^{-1} \mod qh=gf−1modq, and the secret key consists of the FFT-transformed basis matrix and the binary tree.

Signature Generation

To sign a message, the FALCON algorithm performs the following steps:

1. Hashing: Hash the message with a random salt to produce a hash point ccc.
2. FFT and Multiplication: Convert ccc and private key elements fff and FFF into the frequency domain using FFT, then perform coefficient-wise multiplications.
3. Sampling: Use a Gaussian sampler to produce a short vector that fits within a specified bound.
4. Inverse FFT: Convert the short vector back to the time domain using inverse FFT.
5. Compression: Compress the resulting vector to generate the signature components rrr and sss.

Verification

The verification process involves checking that the provided signature corresponds to the hashed message and public key. It ensures that the signature vector is valid and within the expected bounds.

Advantages of FALCON

1. Quantum Resistance: FALCON is designed to resist attacks from quantum computers, making it a strong candidate for future cryptographic standards.
2. Compact Signatures: The algorithm produces relatively small signature sizes compared to other post-quantum schemes, which is beneficial for storage and transmission.
3. Efficiency: The use of FFT allows for efficient polynomial operations, leading to fast key generation, signing, and verification processes.
4. Strong Security Basis: FALCON's security is based on well-studied lattice problems, such as the NTRU problem, which are believed to be hard for both classical and quantum adversaries.

Disadvantages of FALCON

1. Implementation Complexity: FALCON's reliance on FFT and floating-point arithmetic can make implementations more complex and susceptible to implementation errors.
2. Side-Channel Vulnerabilities: As demonstrated by research, FALCON can be vulnerable to side-channel attacks, such as differential electromagnetic analysis, which can extract secret keys from physical devices.
3. Resource Intensive: The algorithm requires significant computational resources, particularly for the FFT operations and Gaussian sampling, which might be a limitation for low-power or resource-constrained devices.

4. Precision Issues: The use of floating-point arithmetic introduces the potential for precision errors, which can affect the correctness and security of the algorithm

Why Dilithium is More Preferable Over Other CRYSTALS Algorithms:

1. Security

Quantum Resistance:

- Strong Mathematical Foundation: Dilithium's security relies on the hardness of lattice problems like LWE and SIS, which are widely considered resistant to attacks from quantum computers. Unlike factorization-based cryptographic methods (such as RSA), lattice-based schemes remain secure even in the face of quantum computing advancements.
- Standardization and Trust: Dilithium is a finalist in the NIST Post-Quantum Cryptography Standardization process. This endorsement from a leading standards body provides confidence in its security assumptions and robustness against future threats.

Proven Security Reduction:

- Reductions to Hard Problems: The security of Dilithium can be formally reduced to well-studied lattice problems. This reductionist approach gives a strong theoretical foundation to its security claims, making it a preferable choice over algorithms with less formal security proofs.

2. Efficiency

Performance Metrics:

- Signature Generation and Verification: Dilithium offers a balanced performance profile with fast signature generation and verification times. This efficiency makes it suitable for a wide range of applications, from lightweight IoT devices to high-performance servers.
- Key and Signature Sizes: Compared to other lattice-based schemes, Dilithium has relatively moderate key and signature sizes. This balance is crucial for minimizing storage and transmission overheads in practical applications.

Implementation Efficiency:

- Software Optimization: Dilithium has been optimized for efficient implementation on various hardware platforms, including standard processors and specialized hardware. Its straightforward design allows for effective optimization techniques, enhancing its performance without compromising security.

- Parallelism and Scalability: The algorithm is amenable to parallel processing, which can further improve its performance on modern multi-core processors. This scalability is important for applications requiring high throughput.

3. Practicality

Ease of Integration:

- Standard API and Libraries: Dilithium is supported by robust cryptographic libraries like liboqs (Open Quantum Safe), which provides standardized APIs for easy integration into existing systems. This support facilitates the adoption of Dilithium in various applications, from secure communications to digital signatures.
- Compatibility with Existing Protocols: The design of Dilithium allows it to be integrated into existing cryptographic protocols with minimal changes. This backward compatibility is crucial for smooth transitions to post-quantum security.

Implementation Simplicity:

- Straightforward Algorithm Design: Dilithium's algorithmic simplicity compared to other lattice-based schemes makes it easier to implement correctly. This simplicity reduces the risk of implementation errors, which can be a significant security concern in cryptographic systems.
- Comprehensive Documentation and Community Support: The algorithm is well-documented, with extensive resources available for developers and researchers. Additionally, an active community provides ongoing support and updates, ensuring that Dilithium remains a viable and secure option.

4. Public Key Size

Smaller Public Key Lengths:

- Efficient Key Representation: One of the key advantages of Dilithium is its relatively small public key size compared to other lattice-based algorithms. This efficiency in key size is achieved through careful design choices that optimize the representation of public keys.
- Impact on Practical Applications: Smaller public keys are particularly beneficial for applications with limited storage and bandwidth, such as mobile and IoT devices. This makes Dilithium a more practical choice for widespread deployment in various environments.

5. Comparison with Other CRYSTALS Algorithms

Performance Trade-offs: Other lattice-based signature schemes, such as Falcon, offer different trade-offs in terms of signature size and performance. However, Dilithium's balance of efficiency, security, and ease of implementation often makes it the preferred choice for general-purpose applications.

Conclusion

CRYSTALS-Dilithium stands out as a preferable post-quantum digital signature scheme due to its robust security foundations, efficient performance, practical implementation considerations, and relatively small public key sizes. Its endorsement by NIST and support from the cryptographic community further solidify its position as a leading candidate for securing digital

communications in the quantum era. As quantum computers become more advanced, the adoption of Dilithium will play a crucial role in ensuring the long-term security and integrity of cryptographic systems worldwide.

Cost Audit of Dilithium Algorithm by IBM

IBM has been actively involved in evaluating post-quantum cryptographic algorithms, including CRYSTALS-Dilithium, to ensure they meet the requirements of security, efficiency, and practicality for real-world applications. This report provides an overview of the cost audit conducted by IBM on the Dilithium algorithm, focusing on computational resources, implementation efficiency, and potential deployment costs.

1. Computational Resources

Processing Power:

- Signature Generation: The computational cost of generating a signature using Dilithium involves matrix-vector multiplications and polynomial arithmetic operations. IBM's analysis indicates that Dilithium requires moderate processing power, making it suitable for a wide range of devices, including low-power IoT devices and high-performance servers.
- Signature Verification: Verification is typically faster than generation, involving fewer polynomial multiplications and additions. This efficiency is critical for applications with high verification rates, such as email clients and authentication servers.

Memory Usage:

- Key Storage: Dilithium's public keys and private keys are designed to be compact, reducing the memory footprint required for key storage. This is particularly advantageous for resource-constrained environments.
- Operational Memory: The memory required for operations during signature generation and verification is moderate, allowing for efficient use on devices with limited RAM.

2. Implementation Efficiency

Software Optimization:

- Optimized Libraries: IBM has worked on optimizing the implementation of Dilithium in cryptographic libraries such as liboqs. These optimizations include leveraging hardware acceleration features and minimizing the overhead of polynomial arithmetic.
- Cross-Platform Performance: The algorithm performs well across different platforms, from embedded systems to cloud servers. IBM's cost audit ensures that the performance remains consistent and efficient regardless of the underlying hardware.

Algorithmic Simplicity:

- Straightforward Design: Dilithium's design is relatively simple compared to other post-quantum algorithms, reducing the risk of implementation errors. This simplicity also translates into lower development and maintenance costs.

3. Deployment Costs

Integration with Existing Systems:

- Backward Compatibility: Dilithium can be integrated into existing cryptographic protocols with minimal changes. This reduces the cost and complexity of transitioning to post-quantum security.

- Standard APIs: The availability of standardized APIs in libraries like liboqs facilitates easy integration into existing software, minimizing development costs.

Operational Costs:

- Bandwidth and Storage: The relatively small public key and signature sizes of Dilithium help in reducing bandwidth and storage costs. This is especially important for large-scale deployments, where data transmission and storage costs can accumulate significantly.
- Energy Efficiency: The moderate computational requirements of Dilithium translate into lower energy consumption, making it a cost-effective solution for battery-powered and energy-constrained devices.

4. Security and Maintenance Costs

Long-Term Security:

- Quantum Resistance: By choosing a post-quantum algorithm like Dilithium, organizations can future-proof their cryptographic infrastructure against quantum attacks. This long-term security reduces the need for frequent algorithm updates and replacements, lowering maintenance costs.

Compliance and Standardization:

- NIST Endorsement: As a finalist in the NIST Post-Quantum Cryptography Standardization process, Dilithium's adoption ensures compliance with emerging global standards. This compliance can reduce the costs associated with meeting regulatory requirements.

5. Case Study: IBM’s Deployment of Dilithium

IBM has conducted a detailed case study on the deployment of Dilithium in various applications, including secure email communication and IoT device authentication.

Secure Email Communication:

- Use Case: Integration of Dilithium in secure email clients like Post-Quantum-K-9-Mail.
- Results: The audit demonstrated that Dilithium provides robust security with minimal performance overhead, making it feasible for everyday use in secure communication.

IoT Device Authentication:

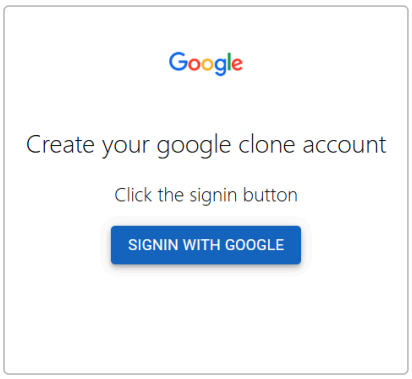
- Use Case: Authentication of IoT devices in a smart home environment.
- Results: Dilithium's efficient key and signature sizes, along with its moderate computational requirements, made it suitable for deployment in resource-constrained IoT devices, ensuring secure and efficient device authentication.

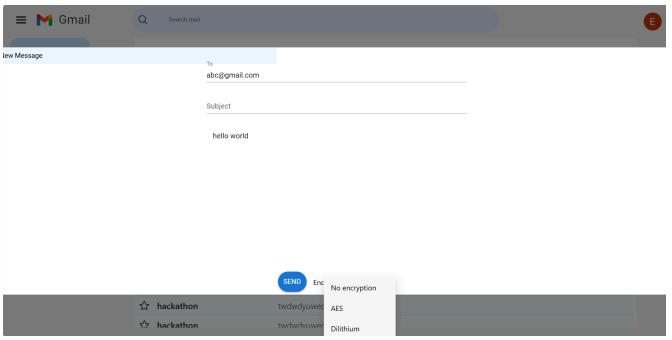
Conclusion

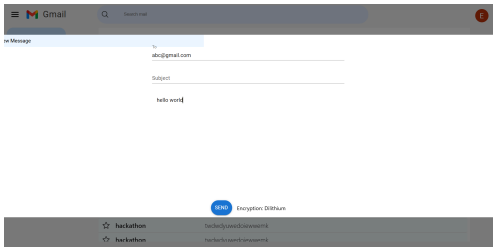
IBM's cost audit of the Dilithium algorithm highlights its efficiency, practicality, and cost-effectiveness for real-world applications. Dilithium's balance of computational resource requirements, implementation simplicity, and long-term security makes it a preferable choice for organizations looking to transition to post-quantum cryptography. By ensuring moderate processing power and memory usage, easy integration, and compliance with emerging

standards, Dilithium stands out as a robust and economical solution for securing digital communications in the quantum era.

EMail Client:(PreP Mail)







The Email client was built for both web version and android version. For web version it was built using react, html , css and java script and liboqs library was compiled to python and linked using django.

Key pair generation:

```
PS E:\sim1> cd dilithium-py
PS E:\sim1\dilithium-py> python main.py
>>
Generating key pair...
Public Key: b'i\xf2\xd8\x15\xa5X\xf1bi\xb4A?>\xd8p\x12zhR\xca\xff\xbcW\x1d\xcd\xc0\x87\xe6\xaa\x1aM\xb1\x04P\xaaJ\xeb=5x\xa7l\x9b\xab\xacU[T\x8cSnU PH\x9a\xc34\x12\x8b\x89o\x07\
xdc\x9a@\x86\x98\xda\xe6\xd5B\x02n\xd0\xe9f\xc3\xa5\xa3\x13u\xb7\x14_\x18<\x8fM\xcf2\x16\xc1`\x14\t%\xc8@\xcc5\xc7\xe3\xaeuW\n\xc8\xcd\xd9\x915d\xf4\xd2\x90e\xefu\xd8H\x86\xaf.\
xa1\x925\xd0(\x0ff\xa5|\x13^\x08\xab\xa5\xc9\xa8\xba\x19\xb2&\x1b\x94\x9d7\xd8\xfd\xed+)RtS\x8f\xda\xb5?j\x1d\xacO\x1e\xda\x8d\xe4\xdaE\xb7\x84.K\xc9\x87qe\xd5BA\x15\xa2%\x82 bk
\xb8\xc4\xad\x139\x8c\xc2\xba\xd9\x98lVH\x92&\x1bv.\x10q\xc1\x1d\x8a\xf9n\xa3\xa8\xf8\xa4\xaf\xb2\x10Fb|\xd3\x02\xe8[EI\x1aB\x8bJ\xd7\x04\'4:j3\x0b\xc1P\x1aU<\x0f"7\xc6\x83\xa0\
xbf\xde\xe1\x7f\x0f-\x0c8H#\x18\x0eg\xb6\x03\xe3u\xbe\xbcFt\x1a~\x18\x86\x83\x161\xabp\x9a6\x7f\xb6\x10\xfd\xd6\x84+-6\x8d\x99#\xce\xc5\xac+t\xfe\x17\x97\x14\xce\xe4\xcdSg\xa0\x
b9*\xa9\xc9\xdf\xf3\xddw\x7f\x15\x10n\x91!<\x01\x1aG\xab\x92\x10\xdbS:\x85\xd9\xc2\xaa\x03\xdf*\xddrs3\xa8\xe7\x1c\xbf<\x102\xc6\xfa@V\xef\xaa\x1a\x00\xcc\xadyZqR4\x82\x81\x7f\x
a9\x97\xdc\xac\x11\x91\n\xa3\r/\x13\xc54E\x91\xc1\x8a\xda>\xad\xfc\xc5\xd5\xd5\xcb\xa3\xd9X\xd2\xc8\xc5r\x8f\x0b\xd0\xf4^\xc2\xe7\xfe\xa5p\xbd\xa3\xc7F@\xd5\x8d\x8a\xa1C,\x82_\x
99\x82\x80\xf0\x08;\n\x9e$\x9c\tA8\x16Q\x9d\xfb\xc9\xee\xec\xe8\x8c\xda\xdb\n\xc4\xcfC\xe4\'\x19\xa4`\xc8j\x9aV\xc0#G\xe3S\r"Ov)\xfd} \xc2\r\xde\xb0o\xf7d\xe4G\x1e\xef\x8c\xe5)\
```

```
Secret Key: b'i\xf2\xd8\x15\xa5X\xf1bi\xb4A?>\xd8p\x12zhR\xca\xff\xbcW\x1d\xcd\xc0\x87\xe6\xaa\x1aM\xb1M\xfa\xc11\x04\xabp}\xf24\x17_\xf8\x9e,I\x86\xc4*U\xc4\x98S\xdc\'#\xe7L\xf
b\xa9\xa1\x19o\xddQ]\xa4\x14\xdd\xcf\x8d.\xe0\n\x1e\xed\xb8\xacH\x08\xbf\xf8j\x95\x99\xed\xea\x15U\x8c\xba_\xbc_\xa4$\r\x03\xa9\x05\x1aIn\xd0\x96m\xe1\xb6!\xe0\xa40c$\x01\x0c\x1
3b\xe3\x84\x89\x12\xc8\x10\xc4\x961\xdaHj\n\xa9\x04\x19F\x08\xda\xb4AY\x10\x84\xc1\x90\x8d\tCMZB\x04P6\x8c\x82\x04\x02\x18\x89q\xe4\x12H\x0b\xb8\x91H\xa8l\xc4\xa6 \xa3\x80\x8d\x
19\x02*\x0c\tH\x01\x96\x11\xc9\x90q\x92\x14\x8ed\x12q\x033f\xcbFpc\x10e\x982\x8dB\x88\x91$\x97H\x19IA\x123H"\x00n\x0b\x01\x8d\xe24*bD\x90\x03\'J$\x00\x11\x1bB\x11\xc0\x08\x11\x8
1\x85LD\x86\x88\x8c\xb0A\xd8\xb8m\xdb\x86m\t&e\xc3F\x82K6d#\'R\xe2\xa2 \x89\x08!\x0b)\x80\x9b&\x88\x19\xa0D\xcb0\n\x02\x95\x00\xd3@b\xe2\x861\x89\xc4PX\xa8%cH\x02\x9c\x02\x06\x8
b\xb4\x01\x04\xb1e$\x03\x86"\x01\x85\x01\xc9\x01 \x06\n\x88\xa4qQ\x06\x84\x1a\xb2P\xcb\xb6M\xe0\x96\x8c\xa2\x88P\xe1\xc8\x91\x0c\x14ma\x04@\t\xc2PH\xc80\x12!!d\x02j\x88\xb8,\x12
\xc6pH&m\xa1\x10%\xe20p\xd4\x04\x8c[\x02\x92\x93 \x91\xca\xc8M\xd4\xb2h\x9c\xc6LI\x96$#\x17&c21#\x12nJ4\x12\x9a\xb6M\x9b0l\x94\xa4q\x01\xb8Q\x12\x00\t\x930\x8c\x8bH\x91\xc9\x98\
x08ZH\x0c\x10\x89\x90\xd1\x04(\xe0\xc2\x88$\xa4A\x8a\x06"$\x91\x0c"#\x04!\x80\x00\x19\x13a\x90\xc2\x04\xd0\x180"%i\xd1\x16b\x9a8%\x83\xc6(\x84\x94\x10\xd0\x02ba\x16\x90\xe3\x84-
\x8b\x08\x10X\x14d\x0cEQ\t\x96@\xc0\x06a\xc2\x08L\xe2@%\xd1\x94%\n\xa4a\x12\x82\x0c\x80\x180\x02\x01%$ \x88\x12$\x91$\x19!\x0c%0\x13\xa7(\x02\x05P\x848\x06\x92\xc6e\x8cF\x84\x02
\x16HT\x881!Hm\x12A\x88\xd9\x10\x91#E2I\x04\x8a\x00\xb6`\tCq\x8a\xa6\x00\x83\x80\x81\xdbB%\xe0\x18`D@\x82\x02\x92!\x11\x16\x90 3.\xe12i\x84\xb8q\x185lH\xc21 \x86D\x1b\x18\x08\x9
3\x16n\x13\xa9HA\x80\x90\x11D.\x02\xa4\x8c"\x82\x91\x1b\xc2)\x9a8@\xda@\x80\x9c\xb2Q\x99Fe\t\x16d\x13DD\x90\x90\x08 \xc0i\xa4\x80@\x8b\x94PP"\r\x8c\x08)"\x93\x10\x13\xc1,\x98&&$
\xb3A\x1a&*\x18$,\x12\xc2\x0ccFP\x91$\x80d\xa8\x10!\xa9mK\x90 \x02\x92E\x928,\xe0\x82M\x03\x19\x8c\x19\x13iD\xc2%#\xa5e\x99\x00m\xe3\xb0\x91\n\x84\x91\x14\x92\x88Q\xa8pb\x14\x8d
```

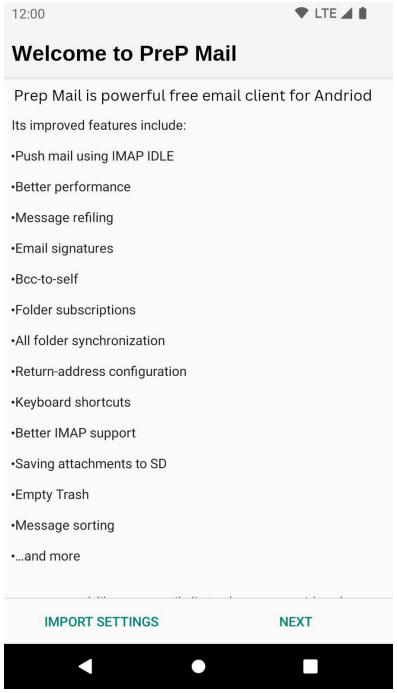
```
Message: b'api'
Signing the message...
Signature: b'j\x8a\xa7Gp\xf36QD\x19\n50\xb7}\xbdf \x91\xc2\x98\xd3\xf6\xe0\x95\xc0\x17\xfc\x8c\\\xe5f\x82\x11\x89mHCL\xa4\x02\x95x\x19\xf8\x07\x1c\xeb\xfe\xcc\n&\x04CS\xae\xd0mb
\x94\x073\x8c;\xfe\xe8H\x01\x1b72\xbd}\x07\x87*eG2\x14U)\xfe\xb5\xaaqqR]\xbak(B\xa50\xd1`\x1d\x0c\xd8m\xd5\x11F-\xecD\xa6\xf6\x98\x81\x95\x8e\xa1\xdf1\xdcs\x85\xb8`\xf9\xf9\x06
\x8a1=\xae\x9eZ\xbf\xf1\xe2\xd0P.\xe7\xe0\xe5S\xf2\x99\x9f{\xf1%a9\xac2g0u`h\xf6W\xa5g\x00\xfd\x11k\xe5\xad\xdae\x90\x9c\xe3\xbe]\x83]\xd6\x8a\x1c\x90\x86g\x9b\xbc1\xee\xdc,?\xc
8u\xf36\x03\x8fI\xf4\x19r\xe9\x02\xd1d\x9d\x156f\xb5-7\xfd\x12*\xe3WfT\t"\x0f\x98\xa4A\xd1G\xa1\t\xe9\xb2\x1b\xe1\xf0\x7f\x1a\xf26\xcb\xeck=n\x85/n\xf1q\x7f\xee\xe7/\x92\xc3\xb3
\xa1\xb2Ys\xea\x93y\xeb\x13fU\xfe\x14}\x9b5sS8\x07~~\x81\x93q\xf9q\x91\x86h\x98/d\x7fpS\x8fA?\xf4 p\xe5*\xd9#7\x02lx\xcc\x10\xc9\xaf\x801\xdf\xa9\x83Q\xe5\xb7\xab\xa7m~3{\xc9\xb
2\xcf\xd9\x008e\xacX\xae\xc1K\x17\xbd2X\x1a\xd9\x1b\xff\x1d\xff\x1d\xdbGtX\xb2\x0c\xde\xcf{50\xa2\x0c\xe3\x9c\x7f\x06\xcd\xac\x16\x13JE\x90\x0c\'\x8cH\x85\xae\xa38\x0e\x1cR\x9a
\xd5\xe3\x0f\x00bS\xfb\x9aPw]\xc6=\xe7\x8f\x8c\x8a\xcd\xd9\xd5\x82#\xa5P\xec\x19\xc7:\x93\xa6S\xa4%\x7f{G\xd5Y\xd9\x92\xccT}\x86\x8e\tc\x01Pi\xcc\xeb\xffm\x19HA\xf6:\xdc\xc7\xf6
\x8b\xb4V\xbdc\xfe$\xde\tX\x96\xad\xb9c\x94\xe7&\x0c\xfb\x10\xd6\xb9\x89l\x08[\xc6\x816\x96\xce\xa2\xc4\xd1H\x1dh\x95\xc2\xce3\xe3\x9e\xbb\x8c1\x16#\t4\x03U?\rW\x1e\xa9\x9dN\xb2
\xfb\x06*v\x94<\xd778\x8c\xe2\xf0\xe2\xf0\x1d\x10\xb04/\x8f\xfd+\x97\x12\xdfJ9\x07\xd1"\xd07\x98|B\x9ft<\x8c<\xefV\x8bd=\x17\x87q\xb61\x1c\x11>(\xb3$\x96 \xf9\xe7\x99q\x16=R\x1d
\'*\x94GU\x13L\x15%\x9f|\x98\xb4\x197\x91\r\xb1r\xd1\xe8\xd6\xcb\xf8\xd1\xfb\xac\xf2\x8f\xa6a\x9b\x82\xb6\xf0kP[\xfab2A\xbfR\x0b\x1f\xb8Mf\xa9\xdf.i\x81\x0f\xfc\xd0\xb8!\xbf\xe8
```

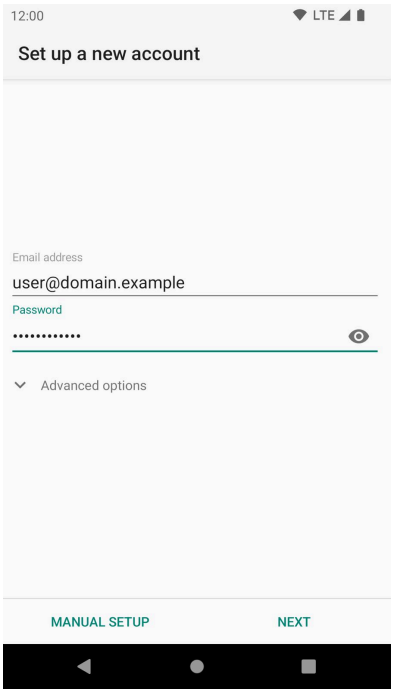
```
Verifying the signature with the correct message and public key...
Verification successful: The signature is valid.
Verifying the signature with the wrong message...
Verification correctly failed with the wrong message.
Generating a new key pair...
New Public Key: b'\xf5-\x11SPvi\xe7\x0f\xbd\xf7\xe8\xdd[\x95\xc0\xb5ao\x1d\xd8\xa1+\x93\x13jD\x1fJ\x05\xb1\x8a\xde\xf22\xde\xd3\xa9\xfe\n\xf1\x9d{v=\x02\xe7h\xb8\x80:\x9d\x1d,\
e9@\xdb\x19\xa2\xff\xd9\xd9>\x00\x18\x8a\x9dc\xca\xcau6\x90\xe1\xc5\x11\xe47J\xe1\x00\x82K\xa4v\x13\xa7-\xe5\xa6jk\xc8\x9bS\xc7\xa67w\t;n\xcd\x0b\x81\xac\x08\x00-\xfb\x19"\xae\
e9\xa41gK\xa3\x8fCl\xf5`\x07uD\x1f\x96\xa3\x8d\x04\x1a\x86\xd1}\x04{\x06\x06\xce\xce\xa5\xc7~\xf6Tb\x7f\xa7\xe1a\xe2,.\x84\xbe3\xcdg\xe3\x96\xb1WC\xed\xa7\x9b\x8e\xdc\x87r\xefP
x08c\x10\xf8\xf9\x00\xa8\xd6\x1c\x0c\x98\x02^L\xfd\x9e\xf0u\xa5(<\xd9\xb7\x0f\x18q\x1d\x83\xbf\xd1\xe0v\x9b#:p\xc2x\xeb\xf3\x90\x02F\'\x9bZ\r$\x02q\xe2a\xf5\xbd|\xc7$\xdb\xc8I:
e\x1fW\xb64\x85\x8e\xa9?R\x96\xd1\xa9\x98\x88\xee\xee\xa7\x9d\xb5\xa6\xaeQ\xa3.\x86\xa6\xb4\x89Q+\xd8\xde \xfc\x91\x8a\xd0\x9a\xd4$\xeb0\xc8\xf6 \x0e\xcbV\xb5\x8c\x12\x7fbJS@\x
6\xac\xd7h"z\x0f\xf2\x0b;[\xab@\x95\x8d\xafC\xfc\x96!B\x0f\xf63A\xee\xaf\xf5\xa9p\xab^\xfe\xd8\x16\xa1r3\x1f\xb64\x96\x07\xcc\x00\xdb\xa2k]\x1f\x13\r< \x17\x02\x7f\xfa=\xc9*6\
e6W*\xd3\xd3=\xb4\xd3\xf4\x18\xec\xc8\x9a\x1c4\x0c`T\xdd\xc3\xb0\xb8\xfd^\xff<\t2\xceg\xe1\x82^e\xc2a0I-\x07\x88\xb9\xdc\xe7i\xcd?\xaa\xfd\t\x15\xc5\x8b\x0c\xa6\xc2\xca\xfa\xe5
```

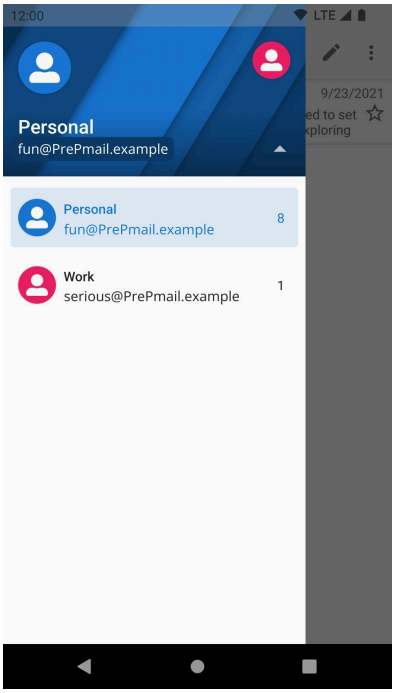
```
New Secret Key: b'\xf5-\x11SPvi\xe7\x0f\xbd\xf7\xe8\xdd[\x95\xc0\xb5ao\x1d\xd8\xa1+\x93\x13jD\x1fJ\x05\xb1\x8a\xe9t$\xc52p\xf8Rc:\x15\xc3\xba\xb7\xbe\x8b.\xccP\x85\x7fK\xe4\xcc\
xce\x08\xd2\xc9\xecJ\xdbI\xf7\x8dt\x82\x8dU\xac\t\r\xa8+\xf6"\xa5\x1c\'_\xdc\xaa\x1c\xe8\x1f\xf1\r$\xb2\xe8\xce 0w\x0b\x82\x98\x90\x8c$P\x99\xc6\x91\xe3\x02\x8e\xc2BM\x14\xa0\x8
dH\xb6p!\x90m\xd3\x90-\x00(J\x12\x10p\xda\x88(\x13\x04i\x08\x07%\x8b\x92h\x03\x87lS$l"\x97\x90\x19\xa0,\x1b\x87QA\x04\x8a\x13\xc4\r\x98\xb0\x8c\x91\xa4M\x81\x10j\x00\t\x8a\x03@\
x92"\x03aR\x80\r\xc84!\x025a\xc3\x16 H\xc4(\t\xc8d\xd8\x10\x90\xa3(\r\x18\t\nA\x98h\x047\x11Z(\x08#"1!\xc7\x85\xc3\xb8\x01d\xb21\x88\xc4AI@\xdc\x94$\xd2\x06,$\x19e\x04\x16\x01\
xcc\xc6\x00"\x80\x91Z\xa4\x84\t\x05A\x14\x88`Q&\x8d\xcb\xc4(\x10\x15D\xc2\x82l\x00Fl\xc0\xa4\x04Z\x88!\x89\xb6Q\x1aAR\x13\x92 \xd1(B\x94(\x82\xe40i\xa3Hm\nG\x90S\x04)I\x10m\x13\
xc0\x05\xd1\x10\x8c\x1b@\x0e\xd2B\n\x9c\xa8,"\x03A\x1c\x03\n\x1c\x13\x01\x1b9\x85\x8c\x80dZ\xa8!\x92\x10\x92H&\x90B\xb0)$7\x81\x1c\xc4\x90\xc3\xc8d@\xa0-\\\x041\\0\x05J\xc8@\x11
\x80D\x12#\x00$\x98\x91\x998J\x88\xa4\x88\x82$\x8e\t\x86\x90\xcb\x00i\xcb\x10\x0cL\xa2a\x9b\x94\x91Q\xc6\x84"\xb3d\x91\xb2@\x0cF\x08P(\x0c\x00\xa9\x80\xd3\x90\x80Y&\r\x00(ib\xc2
\x88\x8b\xa2dI\x98\x90\x19\x14p\xd2\x80m"\x98P\\&\r\xc1\xa2%\xc3\xc4q\xd3\x08\x0c\x14\xa9\x05\xc8 \r\x00\x87) E0\xa0\xa2\t\tCD\x83\xa8lX\xc2\x11\x04\xb4M\x1c\x901K2e\x94D@\xd88M
\xd4B\x89\xe3\xc4$\xca\xa6,\x136\x90!\xa9`I\xa4("\x97I\xe2\x88\x8c\x0c\xa2M\x98\x04\x0e\x18\xc0\r\x9a\x94`$\xb0d\xc0\xb4e\xca$0\xc2\x16&\x12\x07\r\x149\x91Q\x88D\x99\x16\x0c\xd1
(A\x023\x90\x01\xa6M\xcc8NLD\x8c \xc1\x01\xd2\x12\x06\n\'d!\xc1\x81\xc42\t\x11HL \x86\x8c\xca\x18\x8d\xdc&@\xe48\x06\\\xc6\x84\x1c\tP\x0bA\r\x9a\x84@\x18\x07\x0cB4\x06\x83\xb6i\
xd9\xc2\x04!II\x149\x84\x00\x87\x88\x8a\x02H\x04\x99(\n\x98\x91\xe0D\x90\x8c@q!\x80A\xd94%\x145LC@e\x92 \x92\x90F $!R\x11\x01\x81\x0c\xc2I\x98\x02\x01D\x12l\x9aF\x01"\x08\x91\x1
b\xb6\x01\x11%E[\x90L\x12\x94 \x03\xa2 \x92D\x00\x1c\xc8\x8cD H\x1a!\x8ab8j!\x82\x8dH\x14\x85\x94"l`2\x92$\x00l\x1cB%\xe2H\x89[F\x01\x92\x90\t\xa4\xc2EX\xb2\x85\x19\x89\x04\x89\
```

```
xbd\x93\xea\xed\x98\xc1\x80\x14\xd6\x980\xc08\x13\x0c\xfe\x1d\xd5/\x11Y\x08\xd5\xc1f\xbeO\x17\x80\xb6\x7f\x17m\xdc\xb1\x04\xe1C>/\x1c+#\'\x19\x8f\x9b9\xfbw\xde+\xd9[\x06\xfc\x8b
V\r\x04gCCl\x8eh\xb1\t\xf7#\x1d\xfc9\x04@\x83\xf1H_A\xc0K~s\xc8,d\806!\xbfw\x1ft\xa9-n\xceR\t\x910\xcb\x1d\xe7^rm\xebv<\xbb\xb5\xaa\xde6\xc6{E(\x07\xba\xf5\x89}\xbd2\xb0\x92\xd
f\xd4\xe1n\xcc\xd3\x83\\G\x8e\xb2\xe8\xe9s\xc5\xe9\xa1R\x8e@b^\xa4\x0c\x0c1\t\x15\xabb\xc7m\x90\xc3\xf50\xbd\xa7\xcd\xea\x9f\rq\xa9p\xc7\x05\'t\x1f\x8f\x18\x04X\x0e\x00d3N\xa0\x
x87\x8c\xd7_8~,W\xc6\xf9\xee\xb7\xc1\x06\xfe\t\xaf\xf4&\x85\xac\x83\r\x83d\x1e\xf3\x92\x90\xec\xbe\xa9\xad\xe5\x1b$\xaf\x1a\xad\xe1\n3\xfc\x07\xc3\x9duo\xed\xa6m\x8a\x89\x10G\x9
e\x96\x0fk"\xd8\x93\xc5zl\x14?\xfb\x19I!\xe9\xac\x88\xa0d\xb7\xd6s\x81\x8a\xd5\xd0\xdb\x7f!P\xfbV\xfd{\xde\x8fn\xf66*\xf1ij\xcd8\xb9f\xc3\x13@E?\x01x\xeb-\r\xf0\xda}\xbe\xc6\x04
-\xc3\xb2\x8f\xa1\x06k\xbb\xc6d\x17\xcd\xbe\xba{\x86Fm\x9d\xc8\x91M>;:\x14\xc5ds\xa6]V!\x97}\x1b\xfc\xb0U&\x93\xb0:dE\xdc\xd3\x12\xb5\xc1\x98.\xf8\x9f\xd9\xc1\xc2\xd7*\xd8\xfc\x
88f\x84\x8fZ@@g\x9ce},\x14\x12\xbbzU\x91=lQ\x10\xab\t\xa1\x96\xdb\x08\xfb\x87\x1b\xd9\x02\xf5\xeb7n\xf3tFA\xe2\xfd\xd0` \xe4\xfai\xbdh\xba\xbd/u\xfa\xc4c\xf1\xf1Q:\nM\x8c}\xd1\xa
fs\xf2\x96\xbd\xb1\xda\xb6\xef\x16{\xe7\xfa\xf8Z\\-\xba\xb5\xe7TJ\xbd\xcfA\xe0\x01\x94\x03#J2\xd5\xca!*\x96\x84\x17\x8d\xf7\x99\xc3\xcb\xb2i\x91\xed=\xc8G@\xd5\x98\xe7'
Verifying the signature with the new public key...
Verification correctly failed with the new public key.
```

For Android version:







Compiled the liboqs library for Android, first clone the liboqs repository and build it for Linux. Then, navigated back to the root directory and run the build-android.sh script, ensuring the Android NDK is installed and properly configured. Next, compile the Java wrapper as per the repository instructions and modify the method for loading the .so file. Created an Android.mk file for the JNI wrapper, collected all necessary files, and integrated them into your Android project. Finally, use Gradle and Android Studio to build and run your project.

References:

- *Crystal Dilithium Algorithm For Post Quantum Cryptography:Experimentation and Usecase for eSign by Srikanth Sailada Neeti Vohra Subramania at First International Conference on Electrical, Electronics, Information and Communication Technologies 2022*
- *Efficient Parallelism of Post-Quantum Signature Scheme SPHINCS by Shuzhou Sun , Rui Zhang , and Hui Ma at IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS 2020*
- *Implementing CRYSTAL-Dilithium on FRDM-K64 by K. Denisse Ortega Luis J. Dominguez Perez at IEEE 12th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference 2021*
- *Performance Comparison of Post-Quantum Signature Algorithms Through An Android Email Application Plug-in by Radoslav Mandev and Elif Bilge Kavun at IEEE International Conference on Omni-layer Intelligent Systems 2023*
- *On the Use of Quantum Entanglement in Secure Communications: A Survey by K Shannon, E Towe, and O Tonguz at arXiv:2003.07907v1 [cs.CR] 17 Mar 2020*