DESIGN AND IMPLEMENTATION OF QUANTUM-SAFE BLOCKCHAINS

By: Nujud Alyami & Enas Aljuhani
ID: 202209920 & 202210840

Supervisor: Dr.Sultan Almuhammadi

Submitted in partial fulfillment of the
requirements for the degree of
Master's Degree

at

King Fahd University of Petroleum and Minerals
Dhahran, Saudi Arabia
May 2024

Table of Contents

List of Tables

List of Figures

Abstract

The integration of quantum computing into the technological mainstream presents unprecedented risks to the cryptographic underpinnings of current blockchain technologies. Quantum capabilities, particularly algorithms like Shor's, pose threats to the integrity of digital signatures critical to transaction security on blockchain platforms. Addressing these vulnerabilities, our research explores the development of a quantum-safe blockchain designed to resist these emerging threats. This study lays the foundation for a blockchain architecture that incorporates a robust quantum-resistant digital signature scheme based on the Learning with Errors (LWE) problem, utilizing lattice-based cryptography. This approach not only provides an alternative to the conventional Elliptic Curve Digital Signature Algorithm (ECDSA) but significantly enhances transactional security against quantum threats.

The Key Findings of This Project:

1. Baseline Blockchain Implementation: We initiated a standard blockchain model to identify and understand potential vulnerabilities.
2. Integration and Assessment of ECDSA: Within our blockchain model, we implemented and evaluated the effectiveness of ECDSA against quantum threats.
3. Investigation of Lattice-based Cryptography: Our focus was on post-quantum signatures to fortify blockchain security.
4. Design and Application of LWE-based Signatures: We developed and integrated a new signature scheme based on lattice structures.
5. Performance Evaluation: We assessed the viability and performance implications of the new quantum-resistant blockchain architecture, comparing it with ECDSA and other quantum-resistant schemes.

This comprehensive analysis underscores the necessity of advancing blockchain technology to counteract quantum threats, ensuring robust security and functionality in the emerging quantum era.

Acknowledgements

First and foremost, we would like to express our sincere gratitude and thanks to Allah who is most compassionate for keeping us in good health and blessing us with knowledge. We are profoundly thankful to our families for their unwavering support and encouragement during our academic journey. Their love and patience have been our rock.

Our heartfelt appreciation goes to our esteemed supervisor, Dr. Sultan Almuhammadi, for his invaluable guidance, constant support, and constant encouragement. His expertise and dedication have played a pivotal role in the successful completion of this project.

Special thanks are also due to Dr. Muhammad Felemban for his consistent availability and assistance whenever needed. May Allah bless each and every individual who has played a part in this endeavor, and may their kindness and support be rewarded manifold.

We acknowledge with gratitude all individuals who have contributed to our growth and success, and we pray for their continued blessings from Allah. The unique MX program in Quantum Information and Computing at KFUPM has provided us with unparalleled opportunities and experiences that have shaped us into the individuals we are today.

Enas and Nujud

Chapter 1

Introduction

Blockchain technology has swiftly emerged as a pioneering innovation, poised to transform digital transactions through its decentralized, transparent, and immutable framework. Fundamentally, a blockchain operates as a distributed ledger that securely and immutably records transactions across multiple computers. Each ledger entry is encapsulated within blocks, which are time-stamped and linked to preceding blocks, thereby forming an unbroken chain.

Central to the blockchain paradigm is the principle of decentralization. Traditional systems rely on centralized authorities to validate and record transactions, which introduces challenges related to trust, security, and transparency. Blockchain addresses these challenges by enabling transactions to be verified by network participants, known as nodes, through a consensus process.

1.1 Key Components of Blockchain Technology

- **Blocks:** The fundamental units of data storage in blockchain, each containing a cryptographic hash of the previous block, thus ensuring data integrity.
- **Cryptographic Hash:** A function that transforms input data into a fixed-size string of characters, unique to the original data. Any alteration in the input results in a completely different hash, ensuring tamper-proofing.
- **Distributed Ledger:** A ledger shared and synchronized across multiple nodes in a network, ensuring that all participants have access to the same data.
- **Consensus Mechanism:** Processes used to achieve agreement among network participants on the validity of transactions. Notable mechanisms include Proof of Work (PoW) and Proof of Stake (PoS).

- **Cryptographic Primitives:** Algorithms that secure the network, including digital signatures and encryption, ensuring the confidentiality and integrity of transactions.

The motivation behind blockchain technology is its potential to rectify the deficiencies of traditional systems, such as centralized control, single points of failure, and lack of transparency. By decentralizing trust, blockchain offers a more secure, transparent, and efficient method for recording and verifying transactions.

1.2 The Process of Adding a Block to the Blockchain

- **Transaction Broadcasting:** Transactions are broadcast to the network and collected by nodes.
- **Block Creation:** Nodes collect transactions and form a block, which includes a list of transactions and a reference to the previous block's hash.
- **Block Hashing:** The block's content is hashed to create a unique identifier.
- **Proof of Work:** Miners compete to solve a complex mathematical puzzle to validate the block.
- **Block Addition:** Upon solving the puzzle, the block is added to the blockchain, and the process repeats for subsequent blocks.

1.3 Mining and Blockchain

Mining is the process of adding new blocks to the blockchain. Miners solve complex mathematical puzzles, with the first to solve the puzzle earning the right to add the block to the blockchain and receive a reward. Mining is crucial for maintaining the security and integrity of the blockchain network.

1.4 Proof of Work

Proof of Work (PoW) is the process through which blocks are mined and added to the blockchain. It involves significant computational resources and energy consumption. Miners are incentivized by earning mining and transaction fees.

1.5 Immutability of Blockchains

In a blockchain, each block is linked to its predecessor through cryptographic hashes. Altering a block necessitates recalculating all subsequent blocks, demanding immense computational power, thus ensuring immutability. Once a block achieves six or more confirmations, it is practically irreversible, securing the data stored in the blockchain.

1.6 Detailed Blockchain Structure

A blockchain block comprises a header and a list of transactions. The block header contains the hash of the previous block, a timestamp, a Merkle root, a nonce, and the network difficulty target. Full nodes in a blockchain network hold a copy of the entire blockchain, validating every block and transaction. Light nodes, using simplified payment verification (SPV), can verify transactions without downloading the entire blockchain.

1.7 Rewriting the Blockchain Code from Scratch

To deepen our understanding, we decided to rewrite the blockchain code from the ground up on our local machines. This hands-on approach allowed us to engage deeply with the fundamental aspects of blockchain operation, including block creation, transaction handling, and network synchronization. By developing the code from scratch, we were able to explore each component's function and its interaction within the broader system architecture. An overview of blockchain technology as presented in "Beginning Ethereum Smart Contracts Programming" by Wei-Meng Lee outlines the key principles and technologies that underpin blockchain systems, notably its decentralized nature and immutability. The discussion focuses primarily on these foundational aspects, emphasizing their significance in the development and security of blockchain networks.

1.8 Experience with Geth

This section provides a comprehensive overview of our extensive work and experience with the Go Ethereum (Geth) client. By following Chapters 3 and 4 from Wei-Meng Lee's book *Beginning Ethereum Smart Contracts Programming*, we have meticulously set up, configured, and interacted with the Ethereum blockchain. Here is a comprehensive overview of our work and results:

Downloading and Installing Geth

We successfully downloaded and installed the Geth platform, which serves as the backbone for interacting with the Ethereum blockchain. Connected to the main Ethereum network and initiated the syncing process to download the entire history of Ethereum transactions and maintain an up-to-date copy of the blockchain. We used various Geth commands to ensure proper installation and connection, such as `geth --datadir  /.ethereum-testnet`, `geth --goerli --datadir  /.ethereum-testnet --http --http.port 8552 --ipcpath /tmp/geth.ipc`, and `eth.blockNumber`.

Commands Used:

- `geth --datadir  /.ethereum-testnet`
- `.\geth.exe attach`
- `geth attach http://127.0.0.1:8552`
- `eth.syncing`
- `eth.blockNumber`
- `geth --syncmode "full"`

Establishing a Private Ethereum Test Network

Creating a Custom Genesis Block: We created a custom genesis block to initiate the private test network, allowing us to experiment without using real Ether or impacting the main Ethereum network. Configured our private network's parameters to suit our testing needs, ensuring a controlled environment.
Commands such as `geth --datadir "C:\Users\96650\Desktop\MyTestNet\data\node1" init "C:\Users\96650\Desktop\MyTestNet\genesisblock.json"` were used to initialize the network with nodes.

Initializing the Network: Initialized the network with nodes using commands like `geth --datadir  /MyTestNet/data/node1 console > C:\Users\96650\console1.txt 2>&1` and `geth --datadir  /MyTestNet/data/node2 --port 30304 --nodiscover --networkid 2345 console 2>"C:\Users\96650\Desktop\console2.log.txt"`.

Node Configuration and Management

Setting Up Nodes: Set up two nodes in our private test network, each with separate data directories to maintain a clear distinction between their respective blockchains and states. Started up each node and ensured they were functioning correctly using commands like `geth --datadir  /MyTestNet/data/node1 console 2>console1.log`.

Managing Nodes: Used various administrative commands such as `admin.nodeInfo`, `tasklist | findstr geth`, and `taskkill /F /PID <PID>` to manage and monitor node activities.

Account Creation and Balance Management

Creating New Accounts: Created new accounts (wallets) on each node using the Geth client. This involved generating new cryptographic key pairs, which were securely stored in each node's data directory. Used the command `geth account new --datadir /MyTestNet/data/node1` to create accounts on our private test network.

Checking Balances: Learned to check account balances using Geth commands. This involved querying the blockchain to determine the amount of Ether each account held. Verified that balances were zero initially, as no transactions or mining had yet occurred.

Commands Used:

- `eth.accounts`
- `eth.getBalance(eth.accounts[0])`
- `web3.fromWei(eth.getBalance(eth.accounts[0]), "ether")`

Node Interaction and Network Connectivity

Connecting Nodes: Stopped one node and started another, gathering information about each node's status and connectivity. Successfully paired our nodes, enabling them to communicate within our private network. This simulated a real-world scenario where multiple participants interact on the Ethereum blockchain.

Adding Peers: Used commands like `admin.addPeer("enode://
bfa5e38ee8ffe501a79e30e13d443b57b0d92df7a8
e9bed74be0e9013f2a1807cc41565f336a9929
409b24d6ec82db301849de72e9560ad4dcff271cf985f736@192.168.100.13:30304?discport=0")` to add peers and ensure connectivity.

Transaction Execution

Performing Transactions: Sent transactions between accounts using Geth commands. For instance, we used `eth.sendTransaction({from: "0xSenderAddress", to: "0xRecipientAddress", value: web3.toWei(1, "ether")})` to transfer Ether between accounts. Verified transactions on the blockchain by querying the transaction details and ensuring they were recorded accurately.

Mining Rewards: Configured mining rewards to be credited to specific accounts. This involved setting the coinbase account using commands like:
`miner.setEtherbase("0xAccountAddress")`. Observed mined blocks including transactions rewarding the miner, ensuring that mining incentives were correctly implemented.

Note: Our commands listed are working on Windows 11, whereas the commands in the book are for macOS. Specific paths and command syntax might differ, such as using backslashes in Windows paths and forward slashes in macOS paths, as well as possible differences in environment setup and IPC path definitions.

Chapter 2

Literature Review on Quantum-Safe Blockchain Design

The rapid advancements in quantum computing necessitate the design of blockchain technologies that are resistant to quantum attacks. Recent research has provided foundational approaches and new methods for integrating quantum-resistant cryptographic systems into blockchain technology, highlighting both practical implementations and theoretical frameworks.

2.1 Quantum-Resistant Cryptographic Frameworks

The vulnerability of current cryptographic protocols used in blockchain, such as RSA and ECC, to quantum computing threats has led researchers like Alghamdi Almuhammadi (2023) to advocate for the immediate integration of post-quantum cryptographic algorithms to secure blockchain infrastructures. This integration ensures the continued efficacy of blockchains in providing secure and tamper-proof systems for digital transactions and contracts.

2.2 Post-Quantum Signature Schemes in Blockchain

Among the significant contributions to quantum-resistant blockchain design is the adaptation of post-quantum digital signature schemes. Chand et al. (2021) proposes a modification of the Dilithium signature scheme, emphasizing its unforgeability based on the hardness of lattice problems such as Learning with Errors and Short Integer Solution. This scheme not only enhances security but also maintains the essential qualities needed for blockchain operations—confidentiality, authenticity, integrity, and non-repudiation of transactions.

2.3 Compression Techniques for Efficient Blockchain Design

In their effort to design more practical and secure blockchain systems, Bai and Galbraith (2014) explore improved compression techniques for signatures based on learning with errors (LWE), significantly reducing the size of digital signatures. Their work focuses on reducing the transmission data size, which is crucial for maintaining the efficiency and scalability of blockchain systems. Such advancements in compression techniques facilitate the implementation of robust, quantum-safe blockchain architectures that are both secure and performant.

2.4 Quantum Vulnerabilities and Post-Quantum Solutions

Allende et al. (2023) detail the critical need for post-quantum cryptographic solutions in blockchain architectures to counter vulnerabilities exposed by quantum computing capabilities. Their framework for enhancing quantum resistance involves innovative uses of quantum entropy for secure key generation and establishing quantum-resistant TLS connections, demonstrating a practical approach to adapting existing blockchain systems without complete redevelopment.

2.5 Towards Post-Quantum Blockchain by Fernández-Caramés & Fraga-Lamas

Fernández-Caramés and Fraga-Lamas (2020) provide a comprehensive review on the state of quantum-resistant cryptosystems and their application to blockchain technology. They discuss the vulnerabilities of current blockchain encryption and hashing methods under quantum threats, and detail various post-quantum schemes that could secure blockchain from such advanced attacks. Their work is pivotal in directing future efforts towards the development of quantum-resistant blockchain systems, ensuring their longevity and security in the face of evolving quantum computing technologies.

2.6 High-Performance Hardware Implementation of Lattice-Based Digital Signatures

Recent work by Beckwith, Nguyen, and Gaj presents a high-performance hardware implementation of lattice-based digital signatures, focusing on CRYSTALS-Dilithium and FALCON. These implementations target FPGA platforms and are aimed at providing secure digital communications resistant to quantum attacks. Their work includes the first reported FALCON hardware implementation for signature verification and offers insights into optimizing these cryptosystems for better performance and efficiency.

2.7 Future Directions in Blockchain and Quantum Computing

Looking forward, the blockchain community must continue to engage with the developments in quantum computing to anticipate and mitigate potential security threats. The ongoing standardization efforts by bodies such as the National Institute of Standards and Technology (NIST) are critical in this regard, as they guide the development of quantum-resistant cryptographic standards. Moreover, as quantum technology continues to evolve, blockchain systems must adapt dynamically to incorporate new quantum-safe protocols to ensure long-term security and reliability.

Chapter 3

Hash Function in Quantum Era

In this section, we will discuss the essential role of hashing in blockchain technology and the specific challenges it faces in the era of quantum computing. Hashing serves as a critical cryptographic tool for ensuring the integrity and security of blockchain data, providing a unique fingerprint for each block in the chain. However, the advent of quantum computing introduces new threats to traditional hashing algorithms, such as SHA-256 and SHA-3, which are vulnerable to quantum attacks. We will explore the impact of quantum algorithms, like Grover's algorithm, on hashing functions and the potential strategies to mitigate these threats, including the development of quantum-resistant hash functions. Understanding these challenges is vital for safeguarding the security and immutability of blockchain systems in the quantum era.

3.1 The Essential Function of Hashing in Blockchain Technology

Hashing is a fundamental cryptographic technique that underpins the security and structural integrity of blockchain technology. Each block in a blockchain is uniquely identified by a hash value, which is generated by applying a cryptographic hash function to the block's contents. This hash value acts as a digital fingerprint, uniquely identifying each block and securely linking it to the previous one in the chain.

The hash function used in blockchain technology has several key characteristics:

- Determinism: The same input will always produce the same hash output, ensuring consistency across all users.
- Fixed-size Output: The hash function outputs a hash of a constant length, regardless of the size of the input data, making it easier to store and compare hash values.
- Pre-image Resistance: It is computationally infeasible to reverse-engineer the original input from its hash output, protecting against unauthorized modifications.

- Collision Resistance: It is highly unlikely for two different inputs to produce the same hash output, ensuring each block remains unique.
- Avalanche Effect: Small changes to the input result in major, unpredictable changes in the hash, making it difficult to predict new hash values.

These properties are crucial for maintaining the immutability and integrity of the blockchain ledger. By linking blocks through their hash values, any attempt to alter a block would require recalculating the hashes for all subsequent blocks—a computationally prohibitive task.

3.2 Quantum Threats to Hashing Algorithms

While traditional hashing algorithms like SHA-256 and SHA-3 are designed to be secure against attacks using classical computers, quantum computing presents new challenges. Quantum algorithms, such as Grover's algorithm, can search an unsorted database in $O(\sqrt{N})$ time. This could potentially compute pre-images and find collisions with quadratic speed up, meaning that tasks that take classical computers a long time could be accomplished much quicker with quantum technology.

For blockchain technology, this represents a significant threat; a quantum-enabled attacker could feasibly find the preimages of hash functions used in blockchain and alter a block and quickly recompute the hashes for all following blocks, effectively maintaining an intact chain that appears valid. This capability could fundamentally undermine the blockchain's immutability and security.

3.3 The Limitations of Doubling Hash Output as a Countermeasure

To mitigate quantum threats, one proposed approach is to double the size of hash output from n-bits to $2n$-bits to increase computational difficulty and enhance security against quantum attacks. For example, instead of using a 256-bit hash output, a 512-bit output could be used to increase the difficulty of finding collisions or pre-images. This increase in output size effectively doubles the security level against quantum attacks, as Grover's algorithm would require $O(2^{256})$ operations to find a collision, which is computationally infeasible. However, this approach is not without its problems:

- Scalability and Performance: Larger hash sizes require more storage and processing power, potentially impacting system performance and scalability.
- Evolving Quantum Capabilities: As quantum computing advances, merely doubling the hash size might not be sufficient. Continuous improvements in quantum technology might necessitate further increases in hash output size, leading to an ongoing cycle of adjustments.
- Fundamental Security Gaps: Increasing the hash output size does not address the core vulnerabilities introduced by quantum computing. It serves as a temporary fix rather than a long-term solution.

The emergence of quantum computing necessitates the development of new quantum-resistant hash functions. These advancements are crucial for ensuring the long-term security and integrity of blockchain technology in the face of evolving computational capabilities.

To better protect blockchain hashing against quantum threats, researchers are exploring the development of quantum-resistant cryptographic algorithms that offer more durable security solutions.

Chapter 4

Digital Signatures in Quantum-Safe Blockchains

Digital signatures are cryptographic protocols that provide authentication, integrity, and non-repudiation in digital communications and transactions. In blockchain technology, digital signatures serve as a foundational security mechanism, facilitating trust and ensuring that transactions cannot be altered retroactively. This chapter explores the use of digital signatures in blockchain, the impact of quantum computing on their security, and the development of quantum-resistant cryptographic techniques to safeguard future blockchain technologies.

4.1 Principles of Digital Signatures

4.1.1 Asymmetric Cryptography

Asymmetric cryptography, also known as public key cryptography, uses a pair of keys for encryption and decryption: a public key and a private key. The keys are mathematically linked—whatever is encrypted with a public key can only be decrypted by its corresponding private key, and vice versa. This type of cryptography is fundamental to many digital security protocols, including those used in blockchain technologies.

Key Features of Asymmetric Cryptography:

- Key Distribution: Public keys can be freely distributed without compromising security, while private keys must be kept secret by the owner.
- Encryption and Decryption: Ensures that messages are encrypted in a way that only the intended recipient can decrypt them, using their private key.

Digital Signatures: Asymmetric cryptography is used to create digital signatures, which verify the authenticity and integrity of a message or document. Examples: RSA, ECDSA (used in blockchain), and Elliptic Curve Cryptography (ECC).

4.2 Digital Signatures in Blockchain

Digital signatures are used to verify the authenticity of a transaction by proving that the transaction was created by a known sender (authentication) and was not altered in transit (integrity). In blockchain, this is crucial as it helps maintain trust in a decentralized environment.

4.2.1 Implementation in Blockchain

Digital signatures secure blockchain transactions by ensuring that each transaction is signed by the sender's private key. This signature is then verified by other network participants using the sender's public key, linked to their digital identity on the blockchain.

Identity Verification and Non-Repudiation: Digital signatures provide a cryptographic proof of origin, ensuring that transactions are permanently linked to their creator's identity and cannot be repudiated once confirmed.

4.3 Popular Algorithms Used

4.4 RSA (Rivest-Shamir-Adleman)

RSA is one of the earliest public-key cryptosystems and is a widely used digital signature algorithm that forms the basis of many secure communication protocols. It derives its security from the computational difficulty of factoring large composite numbers. The algorithm involves several key steps, including key generation, signing, and verification.

Algorithm 1 RSA Key Generation

```
1: Select two large prime numbers, $p$ and $q$.
2: Compute the modulus, $n = p \times q$.
3: Select a public exponent, $e$, typically a small prime.
4: Compute the private exponent, $d$, such that $(e \times d) \mod ((p - 1) \times (q - 1)) = 1$.
5: Public key: $(n, e)$, Private key: $(n, d)$.
```

Algorithm 2 RSA Signing

```
1: Calculate the hash value of the message, $H(m)$.
2: Compute the signature, $s = H(m)^d \mod n$.
```

Algorithm 3 RSA Verification

```
1: Recover the hash value, $H'(m) = s^e \mod n$.
2: If $H'(m)$ matches the hash of the received message, the signature is valid.
```

4.4.1 RSA Usage in Blockchain

RSA is not typically used in blockchain primarily because of its computational and space inefficiencies compared to other algorithms like ECDSA. RSA requires larger key sizes (often 2048 bits or more), which can be cumbersome in a blockchain environment where efficiency and speed are crucial.

4.5 ECDSA (Elliptic Curve Digital Signature Algorithm)

4.5.1 Mathematical Overview of ECDSA in Blockchain

ECDSA employs elliptic curve cryptography to ensure secure digital signatures, crucial for transaction integrity and authentication in blockchain networks. Here, we'll break down the mathematical operations and principles that underpin ECDSA, providing a clear understanding of how it functions within a blockchain context.

Algorithm 4 Key Generation

```
1: Private Key:
2: A private key $d$ is a randomly selected integer from the set $\{1, 2, ..., n-1\}$, where $n$ is the order of the base point.
3: Public Key:
4: The public key $Q$ is computed as $Q = dG$, where $G$ is the base point on the curve, and $d$ is the private key. The operation involves scalar multiplication of the base point $G$ by the private key $d$.
```

Algorithm 5 Signing Process

```
1: To sign a message $m$, the sender performs the following steps:
2: 1. Select a random integer $k$ from $\{1, 2, ..., n-1\}$.
3: 2. Compute the point $(x_1, y_1) = kG$ on the curve.
4: 3. Calculate $r = x_1 \mod n$. If $r = 0$, select a different $k$.
5: 4. Compute $s = k^{-1}(H(m) + dr) \mod n$, where $H(m)$ is the cryptographic hash of the message $m$. If $s = 0$, select a different $k$.
```

Algorithm 6 Verification Process

```
1: A signature $(r, s)$ on the message $m$ is verified as follows:
2: 1. Compute $w = s^{-1} \mod n$.
3: 2. Calculate $u_1 = H(m)w \mod n$ and $u_2 = rw \mod n$.
4: 3. Compute the point $(x_1, y_1) = u_1G + u_2Q$, where $Q$ is the signer's public key.
5: 4. The signature is valid if $r = x_1 \mod n$.
```

4.5.2 ECDSA Usage in Blockchain

ECDSA is used extensively across various blockchain platforms, including Bitcoin and Ethereum. Bitcoin addresses, for instance, are generated by hashing the public key derived from an ECDSA key pair, providing both security and efficiency.

4.6 Quantum Threats to Digital Signatures

Despite their strong security features, both RSA and ECDSA are susceptible to quantum computing attacks. In particular, Shor's Algorithm, a quantum algorithm, presents a significant threat to blockchain platforms that utilize RSA and elliptic curve cryptography (ECC), including ECDSA, for encryption and digital signatures. This vulnerability underscores the urgent need for blockchain technologies to adapt and integrate quantum-resistant cryptographic methods.

Breaking RSA: RSA encryption relies on the difficulty of factoring large composite numbers into their prime factors. Shor's algorithm can efficiently factorize large integers on a quantum computer, potentially compromising the security of RSA-based encryption used in many blockchain systems.

Breaking ECC: ECC, including its use in ECDSA for digital signatures, relies on the hardness of computing discrete logarithms in a finite field or elliptic curve group. Shor's algorithm can efficiently solve this problem on a quantum computer, compromising the security of ECC-based systems.

Quantum Advantage: Shor's algorithm's time complexity for factoring an n-bit integer and computing a discrete logarithm in a group of order n is $O((\log n)^3)$. This complexity makes it significantly faster than the best-known classical algorithms, which require exponential time for RSA and sub-exponential time for ECC.

Impact on Blockchain Security: In blockchain, where ECDSA is prevalent for generating digital signatures, the ability of Shor's algorithm to efficiently derive private keys from public keys threatens the security of transactions. This could lead to the compromise of digital signatures, allowing attackers to forge transactions and potentially disrupt the integrity of the blockchain.

Mitigating Measures: To counter the threat of Shor's algorithm, post-quantum cryptographic algorithms are being developed. These algorithms aim to provide security against attacks from both classical and quantum computers, ensuring the security of future blockchain systems.

4.6.1 Post-Quantum Digital Signatures

Post-Quantum Cryptography (PQC) is an essential field of research driven by the development of quantum computers, which pose significant threats to the security of classical cryptographic systems. Three principal cryptographic methods are being explored:

Lattice-based Cryptography: This method uses lattice problems, such as the Shortest Vector Problem (SVP) and Closest Vector Problem (CVP), which are considered hard to solve even with quantum computers. Its key strength lies in its resistance to quantum attacks, making it a robust choice for securing systems against future quantum threats. In the next chapter, we will delve further into the intricacies and applications of lattice-based cryptography, exploring how it forms the cornerstone of post-quantum security measures and why it is considered one of the most promising approaches for safeguarding our digital future against quantum threats.

Code-based Cryptography: This type of cryptography relies on the hardness of decoding a random linear code, which is generally considered to be a computationally infeasible task for both classical and quantum computers. The McEliece cryptosystem is a notable example, known for its security against quantum attacks. It operates on the principle that while encoding (and thus encrypting) is computationally efficient, decoding (decrypting without the key) is not feasible due to the NP-completeness of the decoding problem for general linear codes.

Hash-based Cryptography: This method leverages the one-way property of hash functions, which are easy to compute in one direction but hard to invert. Hash-based signatures, for example, can provide security assurances based on the difficulty of finding collisions in a hash function. These systems are particularly noted for their potential resistance to quantum computing attacks since they don't rely on number theory assumptions, which quantum algorithms like Shor's can break.

Each of these cryptographic methodologies—lattice-based, hash-based, and code-based—plays a vital role in advancing the security frameworks used in today's digital and blockchain environments. They offer unique benefits and face distinct challenges, but collectively contribute to the development of more secure, quantum-resistant cryptographic systems that are essential for the future of secure communications and data integrity in an increasingly quantum-aware world.

Lattice-based cryptography, in particular, stands out due to its versatility and robustness. It underpins many post-quantum cryptographic protocols and is the foundation for quantum-resistant encryption schemes, digital signatures, and even advanced applications like fully homomorphic encryption. The inherent complexity of lattice problems like SVP and CVP makes it exceedingly difficult for quantum algorithms to solve them efficiently, providing a strong defense against potential quantum attacks.

Chapter 5

Design and implementation

5.1 Lattice Concepts in Cryptography

Lattices have been studied extensively since the 18th century, with significant contributions from mathematicians like Joseph Louis Lagrange, Carl Friedrich Gauss, and Hermann Minkowski. Their work laid the groundwork for using lattices in cryptographic systems, particularly visible in the construction of a lattice by Carl Jacobi in 1833, which played a foundational role in cryptographic applications.

Definition5.1: A lattice in R^m (an m-dimensional Euclidean space) consists of all possible linear combinations of n linearly independent vectors $b_1, ..., b_n$ with integer coefficients. Specifically, a lattice $\mathcal{L}(b_1, ..., b_n)$ can be defined as:

$$\mathcal{L}(b_1, ..., b_n) = \left\{ \sum_{i=1}^{n} x_i b_i \mid x_i \in Z \right\}$$

where $m \geq n$ typically.

Rank and Dimension:

- **Rank**: The rank of a lattice $\mathcal{L}(b_1, ..., b_n)$ in R^m is n, indicating the number of basis vectors.
- **Dimension**: The dimension of the lattice is m, indicating the dimensional space the lattice spans.
- A lattice where $m = n$ is termed a *full-rank lattice*, indicating that the lattice spans the entire dimensional space.

Matrix Representation:

- The basis vectors $b_1, ..., b_n$ of the lattice can be organized into a matrix form as $B = [b_1, ..., b_n]$ in $R^{m \times n}$.

- Therefore, the lattice can also be expressed through the matrix product as:

$$\mathcal{L}(B) = \{Bx \mid x \in Z^n\}$$

This matrix notation is convenient for computational purposes and helps in visualizing the structure and operations within the lattice.

Definition5.2 the minimum distance for a lattice, represented as $D(\mathcal{L})$, is determined by:

1. The minimum norm of any lattice vector that is not zero:

$$D(\mathcal{L}) = \min\{\|x\| : x \in \mathcal{L} \text{ and } x \neq 0\}.$$

2. The shortest distance between any two different lattice points:

$$D(\mathcal{L}) = \min\{\|x - y\| : x, y \in \mathcal{L} \text{ and } x \neq y\}.$$

When addressing computational aspects of lattices, these structures are often illustrated using a basis matrix B, with integer components. This matrix defines the lattice as intersections of an infinite grid in an n-dimensional space.

Definition 5.3 An n-dimensional lattice $\mathcal{L}$ in R^n satisfies the following properties:

1. It is an additive subgroup, which means:
   - The zero vector, 0, is included in $\mathcal{L}$.
   - For any vectors x and y in $\mathcal{L}$, both $x + y$ and $-x$ are also elements of $\mathcal{L}$.
2. It is discrete, ensuring that every point x in $\mathcal{L}$ is isolated by a neighborhood free from other lattice points.

5.2 Hardness Problems in Lattices

5.2.1 Short Vector Problem (SVP)

The challenge of the Short Vector Problem (SVP) lies in finding the shortest nonzero vector in a specified full-rank lattice $\mathcal{L}$. The magnitude of this minimal vector is denoted by $\lambda_1(\mathcal{L})$ and is known as the first successive minimum of the lattice. The SVP is notorious for its computational difficulty, which escalates with the increase in lattice dimensionality. This complexity is crucial for the security of lattice-based cryptographic systems, as it is universally acknowledged by cryptographic experts to be a formidable problem, particularly in high-dimensional settings.

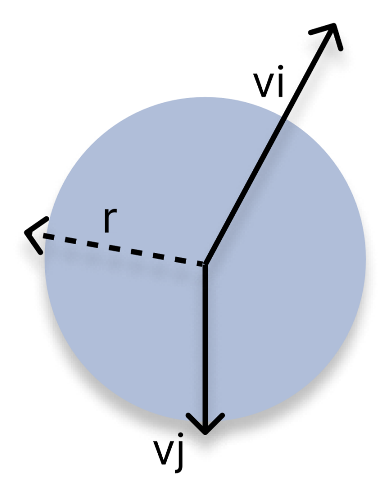

Figure 5.1: If $\lambda_k(\mathcal{L})$ represents the smallest radius r, such that the lattice $\mathcal{L}$ includes at least k linearly independent vectors, each with a length no greater than r.

5.2.2 Closest Vector Problem (CVP)

The Closest Vector Problem (CVP) involves identifying the nearest lattice point to a given non-lattice point t in R^m. This task is generally considered to be more challenging than the Short Vector Problem (SVP). The problem can be mathematically formulated as follows:

$$\text{CVP}(\mathcal{L}, t) = \min_{x \in \mathcal{L}} \|t - x\|$$

where the goal is to minimize the distance between t and x, with x being a point in the lattice $\mathcal{L}$.

5.3 The Learning With Errors (LWE) Problem

The Learning With Errors (LWE) problem has become a fundamental component in modern cryptographic systems due to its average-case hardness and its connection to worst-case hardness assumptions in lattice problems. Here is a detailed formulation of the LWE problem, defined over a set of parameters in the field Z_q:

Given parameters:

- A random matrix A in $Z_q^{m \times n}$,
- A secret vector s in Z_q^n,
- An error vector e in Z_q^m.

We receive pairs (a_i, b_i) **where:**

- a_i is a vector selected at random from the rows of A.
- b_i is calculated as $b_i = a_i^T s + e_i \mod q$, where e_i represents small noise terms added to each result.

The objective is to recover the secret vector s despite the noise introduced by e_i.

This task is challenging in cryptography because solving the LWE problem is not only difficult on average but also tied to the worst-case scenarios involving complex lattice structures.

5.4 Reduction from Shortest Vector Problem (SVP) to Learning with Errors (LWE)

In addressing the reduction from the Shortest Vector Problem (SVP) to the Learning with Errors (LWE) problem, we need to introduce the concept of the dual lattice's basis, symbolized by B^{-1}. This basis plays a key role in comprehending the reduction process.

The dual lattice's basis, denoted by B^{-1}, is essentially the transpose of the original lattice's basis B. This basis comprises a set of vectors that serve as the foundation for the dual lattice, defined as:

$$\mathcal{L}^* = \{y \in R^n : \forall x \in \mathcal{L}, x^T y \in Z\}$$

In this context, $\mathcal{L}$ represents the original lattice, and y is a vector within the dual lattice. The dual lattice's basis B^{-1} helps define the index a_i of $y_i \mod p$'s sub-parallelepiped, indicating how many small tiles are required to reach $y_i \mod p$ from the origin in each dimension.

In the reduction procedure, we sample x once and y_i multiple times, where each pair (x, y_i) used in the LWE instance involves a different y_i. To establish this reduction, we define a_i and b_i as follows:

- Let $a_i = qB^{-1}(y_i \mod p)$, where a_i signifies the index of $y_i \mod p$'s sub-parallelepiped. Each dimension of a_i denotes the number of small tiles necessary to reach $y_i \mod p$ from the origin in the respective direction.

- Let $b_i = q(y_i \cdot x) + \text{noise} \mod q$, where b_i is calculated as the dot product of y_i and x, scaled by q, and then added with noise modulo q.

We aim to demonstrate the relationship between s and b_i in the LWE problem. The calculation of b_i in terms of s is given by:

$$b_i = q(y_i \cdot x) + \text{noise} \mod q$$

By defining $y_i = qB^{-1}(y_i \mod p) + e$, where e represents small error terms, we can rewrite the equation above as:

$$b_i = q(y_i \cdot x) + \text{noise}$$

$$= q((qB^{-1}(y_i \mod p) + e) \cdot x) + \text{noise} \mod q$$

$$= q(qB^{-1}(y_i \mod p) \cdot x + e \cdot x) + \text{noise} \mod q$$

$$= a_i \cdot s + \text{noise} \mod q$$

Here, s is the secret vector we aim to recover, and we observe that b_i is approximately equal to $a_i \cdot s \mod q$. This demonstrates that if we have an efficient algorithm for approximate LWE, we can determine the s value. Using s, we can compute x by multiplying it with B^{-1}. Therefore, if the Closest Vector Problem (CVP) is difficult, then the LWE problem is also challenging, establishing the reduction from SVP to LWE.

5.5 Design of an LWE-Based Lattice Cryptographic Signature Scheme

5.6 Key Generation (KeyGen)

The key generation process is a fundamental step in establishing the security framework of the cryptographic scheme. This process involves creating a public key from a matrix A and vectors S (the secret key) and E (the error vector). Both S and E are selected to have small norms to ensure the security properties of the scheme. The computation of the target vector T is performed as:

$$T = A \cdot S + E \mod q$$

This ensures that the secret key is mixed with noise, effectively concealing it within the public key.

5.7 Signing (Sign)

In the signing phase, random vectors y_1 and y_2 of small norm are chosen to generate a unique, secure signature. The vector v is computed as:

$$v = A \cdot y_1 + y_2 \mod q$$

incorporating the public matrix A. A challenge c is then derived from v and the message μ. The final signature vectors z_1 and z_2 are adjusted using the challenge to include the secret key and error contributions.

5.8 Verification (Verify)

The verification process is essential for ascertaining the authenticity and integrity of the signature against the provided public key and the message. The vector v' is reconstructed as:

$$v' = A \cdot z_1 + z_2 - c \cdot T \mod q$$

using the public key components (A, T) and the signature elements (z_1, z_2, c). This computation aims to match v' with v used during the signing phase.

Algorithm 7 Signing Algorithm

```
1: **Inputs:** $A$, $E$, secret key $S$, message $\mu$
2: Choose two random vectors $y_1, y_2$ of small norm.
3: Compute $v = A \cdot y_1 + y_2 \mod q$.
4: Compute challenge $c = \text{hash}(\mu, v)$, e.g., $c = (\mu + \sum v) \mod 19$.
5: Compute $z_1 = y_1 + c \cdot S \mod q$.
6: Compute $z_2 = y_2 + c \cdot E \mod q$.
7: **Output:** Signature $(z_1, z_2, c)$.
```

Algorithm 8 Key Generation Algorithm

```
1: **Input:** Parameters $n$, $m$, $q$
2: Generate an $m \times n$ matrix $A$ over $Z_q$.
3: Generate an $n \times 1$ vector $S$ with entries chosen according to a small norm.
4: Generate an $m \times 1$ error vector $E$ with entries chosen according to some small norm.
5: Compute $T = A \cdot S + E \mod q$.
6: **Output:** Public key $(A, T)$.
```

Algorithm 9 Verification Algorithm

```
1: **Inputs:** Public key $(A, T)$, signature $(z_1, z_2, c)$, message $\mu$
2: Compute verification vector $v' = A \cdot z_1 + z_2 - c \cdot T \mod q$.
3: Compute challenge $c' = \text{hash}(\mu, v')$.
4: **Output:** "Accept" if $c = c'$ and the norms of $z_1$ and $z_2$ are sufficiently small, otherwise "Reject".
```

Proof of $v = v'$

- v is computed as $v = A \cdot y_1 + y_2 \mod q$.

- Substitute z_1 and z_2 from the signing step into v':

$$v' = A \cdot (y_1 + c \cdot S) + (y_2 + c \cdot E) - c \cdot T \mod q$$

  where $T = A \cdot S + E \mod q$.

- Expand and simplify the expression for v':

$$v' = A \cdot y_1 + A \cdot c \cdot S + y_2 + c \cdot E - c \cdot (A \cdot S + E) \mod q$$

- Note that terms involving c will cancel out:

$$v' = A \cdot y_1 + y_2 \mod q$$

- The simplified expression for v' matches exactly the expression for v:

$$v' = A \cdot y_1 + y_2 \mod q = v$$

This proof confirms that the verification step correctly reconstructs v using the public key and components of the signature, ensuring the integrity and authenticity of the signature under the LWE-based lattice cryptographic scheme. The cryptographic soundness of this equality is predicated on the intractability of the underlying LWE problem and the correct implementation of cryptographic operations, such as hashing and modular arithmetic.

If $v' = v$ and the recalculated challenge c' matches the original c, along with the norms of z_1 and z_2 being within acceptable limits, the signature is accepted. This verifies that the signature has not been altered and was indeed created using the stated secret information, preserving the integrity and confidentiality of the communication.

5.9 Example of LWE-Based Lattice Cryptographic Signature Scheme

In this example, we present a simplified version of an LWE-based lattice cryptographic signature scheme.

Parameters:

- $m = 5$
- $n = 4$
- $q = 13$
- $\mu = 123$

5.9.1 Key Generation:

Matrix A and Secret Vector S are chosen:

$$A = \begin{bmatrix} 4 & 2 & 10 & 5 \\ 0 & 8 & 7 & 9 \\ 3 & 0 & 0 & 4 \\ 10 & 10 & 2 & 7 \\ 12 & 10 & 7 & 4 \end{bmatrix}, \quad S = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}$$

Error Vector E is selected:

$$E = \begin{bmatrix} 2 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Target Vector T is computed as $T = A \cdot S + E \mod q$, which results in:

$$T = \begin{bmatrix} 5 \\ 2 \\ 3 \\ 10 \\ 3 \end{bmatrix}$$

5.9.2 Signing:

Two Random Vectors y_1 and y_2 are chosen:

$$y_1 = \begin{bmatrix} 0 \\ 0 \\ 2 \\ 1 \end{bmatrix}, \quad y_2 = \begin{bmatrix} 2 \\ 0 \\ 2 \\ 1 \\ 2 \end{bmatrix}$$

Compute Vector v as $v = A \cdot y_1 + y_2 \mod q$ yielding:

$$v = \begin{bmatrix} 1 \\ 10 \\ 6 \\ 12 \\ 7 \end{bmatrix}$$

The Hash Message:

$$C = (\mu + \sum v) \mod 19 = (123 + 1 + 10 + 6 + 12 + 7) \mod 19 = 7.$$

The Signature Vectors z_1 and z_2 are then calculated as follows:

$$z_1 = \begin{bmatrix} 7 \\ 7 \\ 9 \\ 1 \end{bmatrix}, \quad z_2 = \begin{bmatrix} 3 \\ 0 \\ 2 \\ 8 \\ 2 \end{bmatrix}$$

5.9.3 Verification:

Compute the Verification Vector v' as $v' = A \cdot z_1 + z_2 - c \cdot T \mod q$, which results in:

$$v' = \begin{bmatrix} 1 \\ 10 \\ 6 \\ 12 \\ 7 \end{bmatrix}$$

The Prime Hash Message c' is calculated:

$$c' = (\mu + \sum v') \mod 19 = (123 + 1 + 10 + 6 + 2 + 12 + 7) \mod 19 = 7.$$

Since $c = c'$, the verification process accepts the signature as valid.

5.10 Implementing LWE-Based Cryptographic Signature Scheme

This document details the implementation of an LWE-based digital signature scheme, highlighting key components and ensuring security through proper parameter selection and hash functions.

5.11 Class Implementation

5.11.1 Initialization

```python
class LWE_Scheme:
    def __init__(self, n, m, q, mu=123):
        self.n = n
        self.m = m
        self.q = q
        self.mu = mu
```

In the initialization method, we set up the parameters for our LWE scheme:

- `n`: Dimension of the secret vector S.
- `m`: Number of rows in matrix A.
- `q`: A large prime number used as the modulus.
- `mu`:Message whose signature is being generated or verified(default value is 123).

These parameters define the size and scope of the cryptographic scheme.

5.11.2 Hashing the Message

```python
def hash_message(self, data):
    if isinstance(data, str):
        data = data.encode('utf-8')
    hash_digest = hashlib.sha256(data).digest()
    return int.from_bytes(hash_digest, 'big') % 19
```

This method takes an input message, converts it to a SHA-256 hash, and then reduces it modulo q:

- `data.encode('utf-8')`: Converts the message to bytes if it is a string.
- `hashlib.sha256(data).digest()`: Computes the SHA-256 hash of the data.
- `int.from_bytes(hash_digest, 'big') % self.q`: Converts the hash to an integer and takes it modulo q.

This ensures that the hash value fits within the range $[0, q-1]$.

5.11.3 Checking Linear Independence of Matrix A

```python
def check_linear_independence(self, A):
    rank = np.linalg.matrix_rank(A)
    if rank < min(A.shape):
        raise ValueError("Matrix A is not linearly independent. The program will terminate.")
```

This method verifies that matrix A is linearly independent:

- `np.linalg.matrix_rank(A)`: Computes the rank of matrix A.
- If the rank is less than the smaller dimension of A, it raises an error indicating linear dependence.

Ensuring A is full rank is necessary for the cryptographic scheme to function correctly.

5.11.4 Checking Small Norm of Vectors

```python
def check_small_norm(self, vector, threshold):
    norm = np.linalg.norm(vector)
    if norm >= threshold:
        raise ValueError(f"Vector {vector} is not a small norm vector (norm: {norm}).")
```

This method checks if a vector has a small norm:

- `np.linalg.norm(vector)`: Computes the Euclidean norm (length) of the vector.
- If the norm exceeds the specified threshold, it raises an error.

This ensures that vectors S, E, $y1$, and $y2$ have small norms, which is crucial for the security of the scheme.

5.11.5 Key Generation

```python
def keygen(self):
    A = np.random.randint(0, self.q, size=(self.m, self.n))
    self.check_linear_independence(A)

    S = np.random.randint(0, self.q ** 0.05, size=self.n)
    E = np.random.randint(0, self.q ** 0.05, size=self.m)

    self.check_small_norm(S, threshold=self.q)
    self.check_small_norm(E, threshold=self.q)

    T = (A @ S + E) % self.q

    return (A, T), (S, E)
```

This method generates the public and private keys:

- A: A random matrix with elements in $[0, q-1]$ and verified for linear independence.
- S and E: Small norm vectors generated with elements in $[0, q^{0.05} - 1]$.
- $T = (A \cdot S + E) \mod q$: Computes T by taking the matrix-vector product and adding E, then reducing modulo q.

The method returns the public key (A, T) and the private key (S, E).

5.11.6 Signing a Message

```python
def sign(self, A, S, E, message):
    y1 = np.random.randint(0, self.q ** 0.05, size=self.n)
    y2 = np.random.randint(0, self.q ** 0.05, size=self.m)

    self.check_small_norm(y1, threshold=self.q)
    self.check_small_norm(y2, threshold=self.q)

    v = (A @ y1 + y2) % self.q
    c = self.hash_message(message.encode('utf-8') + v.tobytes())
    z1 = (y1 + S * c) % self.q
    z2 = (y2 + E * c) % self.q

    return (z1, z2, c), (y1, y2, v)
```

This method signs a message:

- $y1$ and $y2$: Small norm vectors generated randomly.
- $v = (A \cdot y1 + y2) \mod q$: Computes v by taking the matrix-vector product and adding $y2$, then reducing modulo q.
- c: Hashes the concatenated message and v.
- $z1 = (y1 + S \cdot c) \mod q$ and $z2 = (y2 + E \cdot c) \mod q$: Computes the signature components by combining $y1$, $y2$, S, E, and c.

The method returns the signature $(z1, z2, c)$ and the intermediate values $(y1, y2, v)$.

5.11.7 Verifying a Signature

```python
def verify(self, A, T, signature, message):
    z1, z2, c = signature
    v_prime = (A @ z1 + z2 - T * c) % self.q
    c_prime = self.hash_message(message.encode('utf-8') + v_prime.
   tobytes())
    return c == c_prime
```

This method verifies a signature:

- $v\_prime = (A \cdot z1 + z2 - T \cdot c) \mod q$: Recomputes v_prime using the signature components and public key.
- c_prime: Hashes the concatenated message and v_prime.

- Compares c and c_prime to determine if the signature is valid.

The method returns `True` if the signature is valid, otherwise `False`.

5.12 Performance Analysis of an LWE-Based Cryptographic Signature Scheme

In this section we will presents a detailed analysis of the performance metrics associated with an LWE-based cryptographic signature scheme. The analysis includes key generation, signing, and verification times across varying parameter sizes.

Timing Results for Different Values of n

The script runs a performance test for different values of n (from 1000 to 10000, incremented by 1000 each time). The n parameter corresponds to the dimension of the lattice used in the LWE-based scheme, which affects the security and computational complexity of the cryptographic operations. For each n, the script reports the total time required to complete both signing and verification processes, measured in milliseconds (ms). The times increase with n, indicating higher computational costs for larger parameters.

n	Public Key Size (bytes)	Secret Key Size (bytes)	Signature Size (bytes)	Avg Signing Time (ms)	Avg Verification Time (ms)
1000	4,008,004	8,004	8,008	3.52359	3.31375
2000	16,016,004	16,004	16,008	12.0352	11.9175
3000	36,024,004	24,004	24,008	24.533	24.334
4000	64,032,004	32,004	32,008	40.7494	41.2427
5000	100,040,004	40,004	40,008	69.5441	69.3755
6000	144,048,004	48,004	48,008	155.167	151.901
7000	196,056,004	56,004	56,008	199.923	200.436
8000	256,064,004	64,004	64,008	225.373	226.513
9000	324,072,004	72,004	72,008	204.494	202.963
10000	400,080,004	80,004	80,008	285.684	285.782

To be revised

Table 5.1: Performance Metrics for Cryptographic Operations

Tabulated Data

The table displays detailed performance metrics for each tested value of n. These metrics include:

- **Public Key Size (bytes):** The combined size of the public keys in bytes. It increases significantly with n, reflecting more data needed to represent the keys as n grows.

- **Secret Key Size (bytes):** The size of the secret keys in bytes, which also increases with n.
- **Signature Size (bytes):** The size of the digital signatures produced by the scheme. Like the keys, the signature size increases with n.
- **Avg Signing Time (ms)** and **Avg Verification Time (ms):** These columns show the average time it takes to sign and verify a message, respectively, averaged over a number of iterations. Both times generally increase with n, indicating greater computational effort for larger parameter sizes.

5.13 Detailed Analysis

Increasing Complexity

As n increases, all aspects of the cryptographic system's resource usage also increase. This includes the time taken for operations and the memory required for storing keys and signatures.

Security vs. Performance Trade-off

The growth in key and signature sizes, along with increased computation times, suggest a trade-off between security and performance. Larger n values, which potentially offer higher security, require more resources and result in slower operations.

Practical Implications

For practical implementations, choosing the right value of n involves balancing the security level against the system's performance constraints. The data helps in understanding how different choices of n affect these factors, which is crucial for designing systems that need to operate efficiently while being secure against quantum computing threats.

Analysis of Results

The performance data indicate a clear increase in both the key sizes and the computational times for signing and verification as n increases. This trend is further illustrated in the figure below.

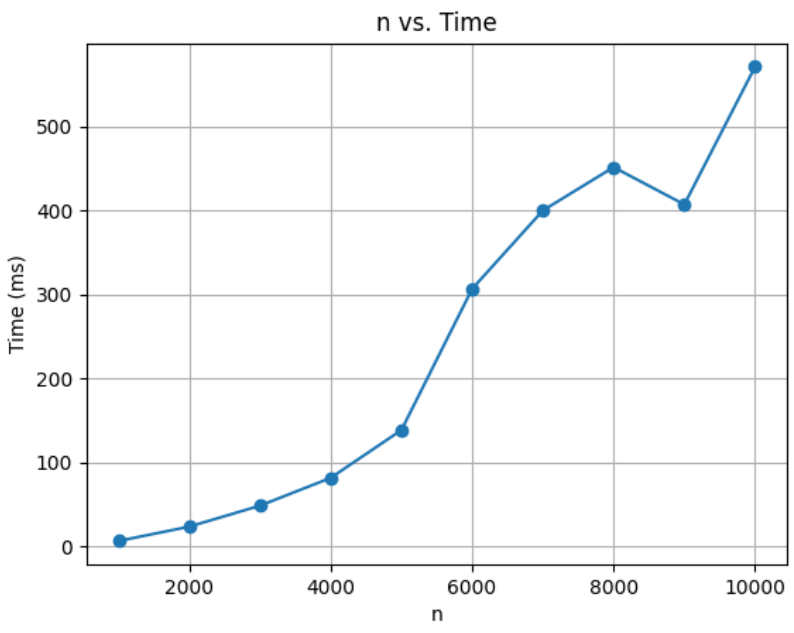

Figure 5.2: The computational time required for signing and verification processes as a function of parameter n.

As depicted in Figure 5.2, the computational time required for signing and verification processes increases nonlinearly with respect to the parameter n. The graph clearly illustrates that both the signing and verification times escalate significantly for larger values of n. This pattern is indicative of a quadratic time complexity, $O(n^2)$, which aligns with the observed steep increase in computational time.

The sharp rise in time, especially noticeable around $n = 6000$, can be attributed to the intensive matrix-vector multiplications and modular arithmetic operations involved in the cryptographic processes. These operations become more complex and time-consuming as the parameter n grows, thus leading to a quadratic growth in computational time.

Chapter 6

NIST's Post-Quantum Cryptography Candidates: Analysis and Performance

As the prospect of quantum computing becomes more tangible, the cryptographic community has intensified efforts to develop secure systems that are resilient against quantum threats. Digital signature schemes, vital for data integrity and authentication, are particularly crucial in this transition. Among the emerging solutions, three prominent schemes SPHINCS+, Falcon, and CRYSTALS-Dilithium demonstrate how different cryptographic foundations can provide robust security in a post-quantum world. This document examines these schemes, comparing their underlying technologies, performance, and suitability for various applications, to provide a clearer understanding of their roles in future cryptographic infrastructures.

Signature Scheme	**Type/Variant**	**Description/Features**
SPHINCS+	SPHINCS+-SHAKE256	Utilizes the SHAKE256 hash function, optimizing for robust security and efficient performance.
	SPHINCS+-SHA-256	Employs SHA-256, known for strong security and wide use.
	SPHINCS+-Haraka	Features fast processing and small output sizes with the Haraka hash function.
Falcon	Falcon-512	Offers fast operations and small key sizes, optimized for lower security levels.
	Falcon-1024	Provides higher security with larger keys and slower operations.
CRYSTALS-Dilithium	Dilithium2	Configured for basic security needs with balanced performance.
	Dilithium3	Enhances security, suitable for more sensitive environments.
	Dilithium5	The highest security level, intended for maximum security requirements.

Table 6.1: Comparison of Digital Signature Schemes

Feature/Quality	**SPHINCS+**	**Falcon**	**CRYSTALS-Dilithium**
Security Basis	Hash-based	Lattice-based	Lattice-based
Quantum Resistance	Strong (relies on hash functions)	Strong (based on lattice problems)	Strong (based on lattice problems)
Signature Size	Larger signatures	Most compact signatures among lattice-based schemes	Larger than Falcon, but configurable
Performance	Flexible configurations for balance between size and speed	High speed in signature generation and verification	Efficient key management, suitable for large-scale use
Versatility	Highly adaptable to different security needs	Versatile, suitable for identity-based encryption	Offers different security levels for NIST categorizations
Operational Efficiency	Moderate, depends on configuration	High, optimized for performance and minimal resource usage	High, efficient key generation and management
Ideal Use Case	Environments where quantum resistance and statelessness are critical	Applications requiring high performance and compactness	Environments needing robust security with flexible security levels

Table 6.2: Comparative Analysis of Post-Quantum Cryptographic Digital Signature Schemes.

The table provides a succinct comparison of three post-quantum digital signature schemes: SPHINCS+, Falcon, and CRYSTALS-Dilithium, each designed to offer security against potential quantum computational threats. SPHINCS+ utilizes hash-based methods, ensuring stateless operations and large keys, ideal for scenarios where maintaining state is impractical. Falcon, on the other hand, is built on NTRU lattice-based cryptography, emphasizing compactness and speed, making it suitable for high-performance applications that require efficient, compact cryptographic tools. CRYSTALS-Dilithium also employs a lattice approach through Module-LWE, offering configurable signature sizes and robust security, tailored for environments that demand stringent security measures. Collectively, these schemes illustrate diverse strategies in quantum-resistant cryptography, each aligning with specific operational efficiencies and use-case requirements.

6.1 Comparative Analysis of Post-Quantum Cryptographic Algorithms

This section presents a comprehensive analysis of several cryptographic algorithms, comparing traditional ECDSA with post-quantum candidates Dilithium2, Dilithium3, Dilithium4, Falcon-512, Falcon-1024, and SPHINCS+ variants. The metrics under comparison include key generation time, signing time, verification time, key sizes, and security levels.

The era of quantum computing presents a nuance of threats to classical cryptographic algorithms, such as ECDSA. Post-quantum signature schemes are being crafted to withstand quantum computing attacks. This study benchmarks the performance of Dilithium and Falcon, contrasting them with the classical ECDSA and another post-quantum contender, SPHINCS, to offer insight into their feasibility as quantum-resilient signature solutions.

6.1.1 Methodology

The analysis was conducted by running each algorithm for 100 iterations to ensure statistical significance, measuring key generation, signing, and verification times. Key sizes and security levels were also documented.

- Hardware: ASUS VivoBook 14X OLED
- Software: PyCharm, Python version 3.12

Evaluation Criteria:

- Key Generation Time (seconds)
- Signing Time (seconds)
- Verification Time (seconds)
- Key Sizes (bytes)
- Security Levels (bit security)

6.2 Performance Analysis

This section presents a comparative performance analysis of various cryptographic algorithms. It focuses on key metrics such as key generation time, signing time, verification time, and key sizes. Each metric evaluates the efficiency and security of the algorithms in a post-quantum cryptographic landscape.

Performance Rank	Average Key Generation Time (s)	Average Signing Time (s)	Average Verification Time (s)	Secret Key Size (bytes)	Public Key Size (bytes)
1	Dilithium2 (0.261×10^{-3})	Dilithium2 (0.447×10^{-3})	Dilithium3 (0.11×10^{-3})	ECDSA (48)	SPHINCS_SHA256_192f (48)
2	Dilithium3 (0.408×10^{-3})	Dilithium4 (0.532×10^{-3})	Falcon_1024 (0.16×10^{-3})	SPHINCS_SHA256_192f (96)	SPHINCS_SHA256_256f (64)
3	Dilithium4 (0.487×10^{-3})	Dilithium3 (0.732×10^{-3})	Falcon_512 (0.19×10^{-3})	SPHINCS_SHA256_256f (128)	ECDSA (96)
4	ECDSA (2.408×10^{-3})	ECDSA (2.292×10^{-3})	Dilithium4 (0.195×10^{-3})	Falcon_512 (1281)	Falcon_512 (897)
5	SPHINCS_SHA256_192f (4.135×10^{-3})	Falcon_512 (8.883×10^{-3})	Dilithium2 (0.85×10^{-3})	Falcon_1024 (2305)	Falcon_1024 (1793)
6	SPHINCS_SHA256_256f (10.115×10^{-3})	Falcon_1024 (15.41×10^{-3})	SPHINCS_SHA256_192f (5.473×10^{-3})	Dilithium2 (2800)	Dilithium2 (1184)
7	Falcon_512 (29.812×10^{-3})	SPHINCS_SHA256_192f (103.233×10^{-3})	SPHINCS_SHA256_256f (5.484×10^{-3})	Dilithium3 (3504)	Dilithium3 (1472)
8	Falcon_1024 (76.464×10^{-3})	SPHINCS_SHA256_256f (221.529×10^{-3})	ECDSA (9.334×10^{-3})	Dilithium4 (3856)	Dilithium4 (1760)

Table 6.3: Ranking Performance: A Comparative Analysis of Algorithms (1=Best, 8=Worst).

6.2.1 Average Key Generation Time Analysis

Dilithium2 demonstrates the fastest key generation time among the algorithms listed, indicating that it is the most efficient when it comes to setting up new cryptographic keys. This efficiency is due to its foundation on Ring Learning With Errors (Ring-LWE), which allows for streamlined arithmetic operations and avoids complex sampling methods. ECDSA also performs well, but Dilithium2 is roughly an order of magnitude faster. The Falcon variants take the longest time for key generation, as their process involves more intricate lattice-based computations.

6.2.2 Average Signing Time Analysis

Dilithium2 stands out again with the fastest signing time. Its rapid signing process is attributed to the use of Ring-LWE. Specifically, Dilithium avoids the use of Gaussian sampling, instead generating randomness uniformly from a power-of-2 range, which is computationally simpler and faster. ECDSA is slower than all the Dilithium variants but faster than Falcon and SPHINCS+ variants. SPHINCS+ has significantly longer signing times, which may be a limiting factor in certain applications. The straightforward and efficient nature of Dilithium2's signing process makes it a highly attractive option for applications requiring quick and reliable digital signatures.

6.2.3 Average Verification Time Analysis

Among the compared algorithms, Dilithium3 offers the fastest verification time by a considerable margin, making it highly efficient for scenarios where verification speed is critical. ECDSA has the slowest verification time, which can be a downside in applications requiring quick validation of signatures. Falcon variants, while slower in other metrics, provide extremely competitive verification times. Falcon's design, based on the NTRU lattice and the GPV framework, allows for efficient verification through compact and fast Fourier sampling methods. This design ensures that Falcon can maintain high performance in verification, making it a strong candidate for applications that demand quick and reliable signature validation.

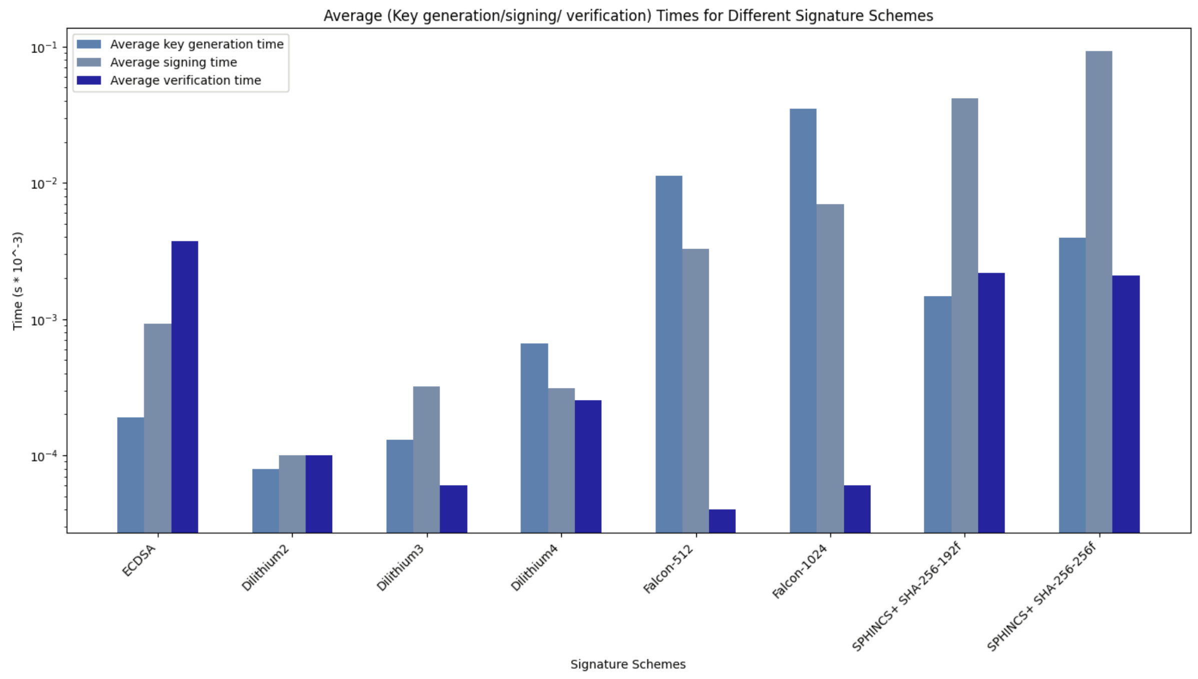

Figure 6.1: compares time metrics for key generation, signing, and verification across cryptographic schemes, with Dilithium variants and Falcon_512 having longer times, while ECDSA and SPHINCS variants show shorter durations in these processes.

6.2.4 Key Sizes Analysis

ECDSA has the smallest key sizes, which could be beneficial in terms of storage and transmission efficiencies. Dilithium variants have larger key sizes, reflecting their design to offer security against quantum computational attacks, which requires larger keys to provide equivalent classical security levels. Falcon also shows larger key sizes due to its post-quantum security properties. In contrast, SPHINCS+ exhibits small key sizes, especially in its 192-bit security level variant.

6.2.5 Security Levels Analysis

The security level indicates the amount of protection against attacks an algorithm provides. A higher security level typically corresponds to a greater resistance to attacks, with ECDSA providing a 192-bit security level. Dilithium3 matches this, while Dilithium4 offers an even higher 256-bit security level. Falcon-1024 and SPHINCS+ SHA256 256f

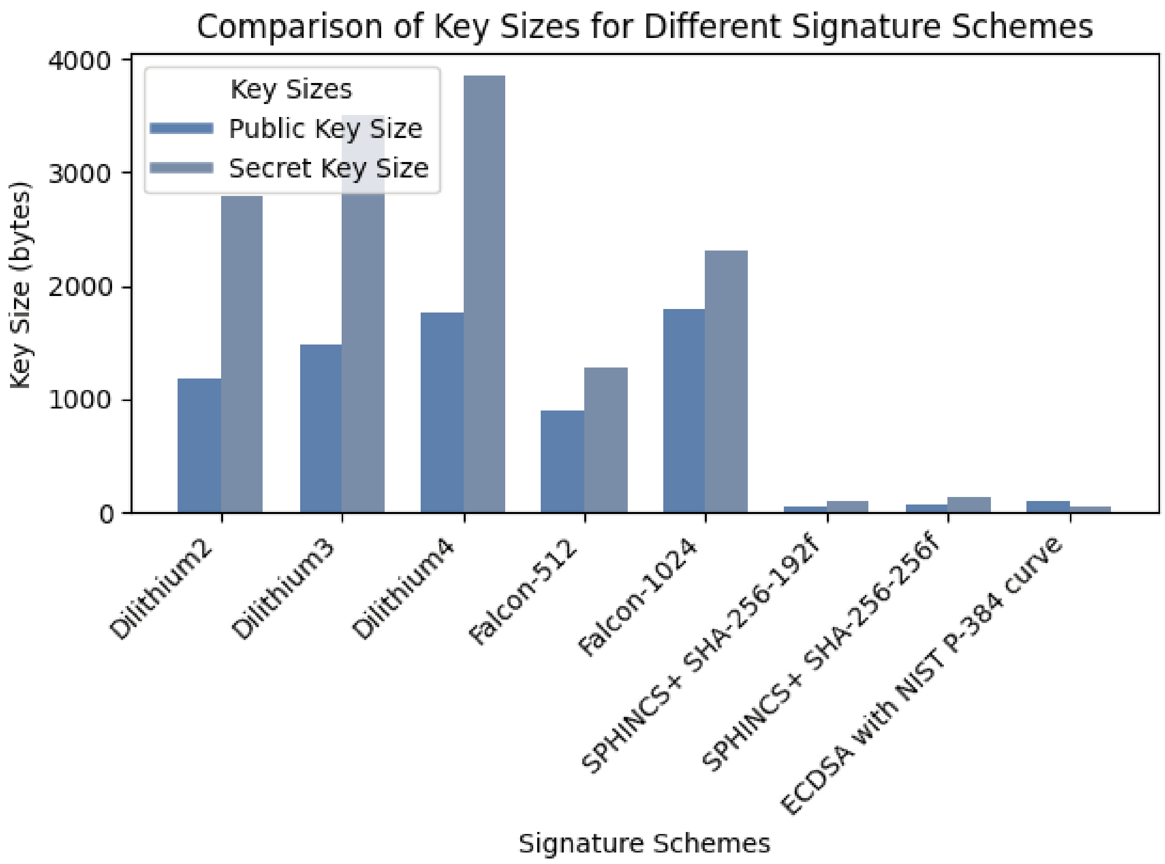

Figure 6.2: presents a comparison of secret and public key sizes in bytes for different cryptographic signature schemes, highlighting the variance between them for algorithms such as ECDSA, Dilithium, Falcon, and SPHINCS.

also provide a 256-bit security level, which indicates a higher degree of theoretical security assuming the hardness of the underlying problems holds against advancements in cryptanalysis and computational power, including quantum attacks for the post-quantum algorithms.

The comparative analysis of post-quantum cryptographic algorithms highlights the strengths and weaknesses of each algorithm in key areas such as key generation time, signing time, verification time, key sizes, and security levels. Dilithium2 emerges as a strong contender with the fastest key generation and signing times, while Dilithium3 excels in verification speed. ECDSA remains efficient in key sizes and offers a 192-bit security level, but is slower in key generation and verification. Falcon variants have longer key generation times and key sizes but competitive verification times. SPHINCS+ stands

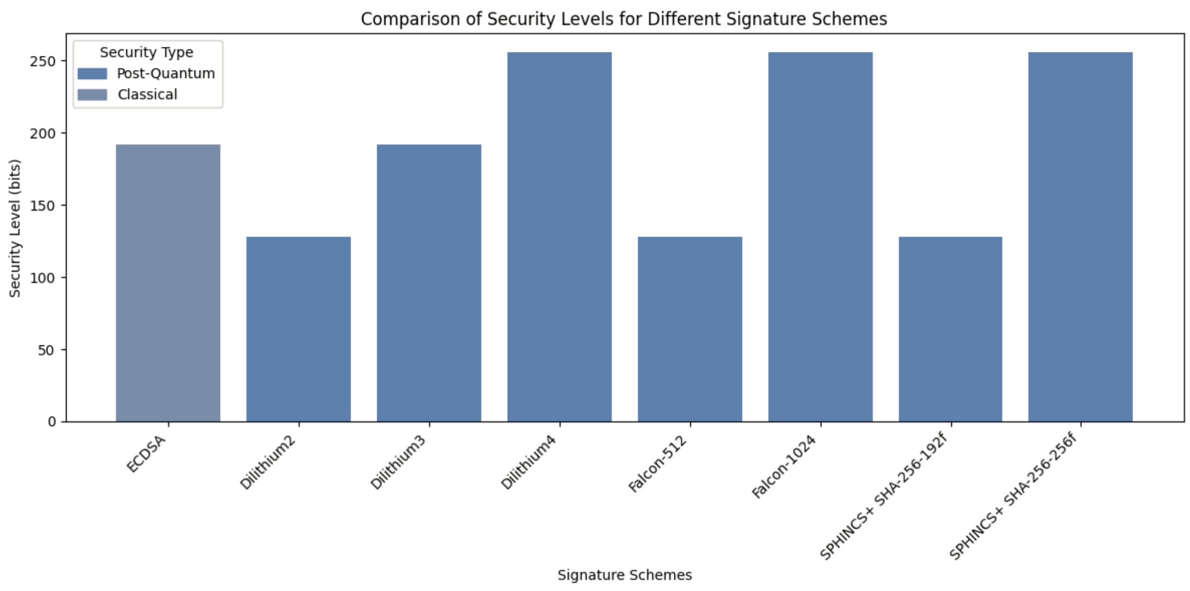

Figure 6.3: shows a bar chart ranking the bit security of cryptographic algorithms, from the lowest with ECDSA to the highest with Dilithium4.

out with small key sizes and varying security levels. Each algorithm has its own trade-offs and suitability for different use cases, emphasizing the importance of considering several factors when choosing a cryptographic solution in the era of quantum computing threats.

Metrics	**ECDSA**	**Dilithium2**	**Dilithium3**	**Dilithium4**	**Falcon-512**	**Falcon-1024**	**SPHINCS-SHA256-192f**	**SPHINCS-SHA256-256f**
Average Key Generation Time (ms)	2.408	0.261	0.408	0.487	29.812	76.464	4.135	10.115
Average Signing Time (ms)	2.292	0.447	0.732	0.532	8.883	15.41	103.233	221.529
Average Verification Time (ms)	9.334	0.85	0.11	0.195	0.19	0.16	5.473	5.484
Secret Key Size	48 bytes	2800 bytes	3504 bytes	3856 bytes	1281 bytes	2305 bytes	96 bytes	128 bytes
Public Key Size	96 bytes	1184 bytes	1472 bytes	1760 bytes	897 bytes	1793 bytes	48 bytes	64 bytes
Security Level	192-bit (3)	128-bit (1)	192-bit (3)	256-bit (5)	128-bit (1)	256-bit (5)	128-bit (1)	256-bit (5)

Table 6.4: Comparison of Different Signature Schemes

Chapter 7

Conclusion

This project delineates the design and implementation of a quantum-safe blockchain architecture employing lattice-based cryptographic techniques. The principal objective was to mitigate the potential threats quantum computing poses to the cryptographic underpinnings of existing blockchain technologies.

Initially, a conventional blockchain was developed to pinpoint vulnerabilities susceptible to exploitation by sophisticated quantum algorithms, notably Shor's algorithm. This exploration was pivotal in understanding the limitations of current cryptographic approaches when confronted with quantum capabilities. Subsequently, attention was directed towards post-quantum cryptographic algorithms, emphasizing lattice structures, resulting in the formulation of a signature scheme predicated on the learning with errors (LWE) problem.

This LWE-based signature scheme was incorporated into our blockchain framework. Our theoretical evaluation suggests that the scheme could effectively withstand quantum attacks. The performance assessment underscored the efficiency and resilience of the designed quantum-safe blockchain framework, proposing its applicability to practical scenarios.

In summation, this report offers a detailed framework for the integration of quantum-resistant cryptographic methods into blockchain technology. The theoretical viability and potential efficacy of the LWE-based signature scheme underscore its utility for crafting secure blockchain systems that are impervious to the advancements in quantum computing. It is essential to pursue ongoing research and development to adapt continually to the dynamic quantum technological landscape, thus safeguarding the enduring security and feasibility of blockchain infrastructures. This research not only addresses extant security challenges but also establishes a foundation for a secure and resilient blockchain ecosystem capable of countering future quantum threats.

Bibliography

[1] Gewu Bu, Serge Fdida, Maria Potop-Butucaru, and Bilel Zaghdoudi. *Blockchain-based decentralized identity system: Design and security analysis.* Clermont Ferrand University and Sorbonne University, May 2024. Available at: `https://example.com/path/to/document.pdf`.

[2] Shi Bai, Steven D. Galbraith. *An Improved Compression Technique for Signatures Based on Learning with Errors.* Department of Mathematics, University of Auckland, New Zealand, Published at CT-RSA 2014.

[3] Marcos Allende, Diego López León, Sergio Cerón, Adrián Pareja, Erick Pacheco, Antonio Leal, Marcelo Da Silva, Alejandro Pardo, Duncan Jones, David J. Worrall, Ben Merriman, Jonathan Gilmore, Nick Kitchener, and Salvador E. Venegas-Andraca. *Quantum-resistance in blockchain networks.* Scientific Reports, vol. 13, no. 5664, 2023. doi:10.1038/s41598-023-32701-6. Available at: `https://www.nature.com/articles/s41598-023-32701-6`.

[4] Atul Kumar and Arun Mishra. *LWE Based Quantum-Resistant Pseudo-Random Number Generator.* The International Arab Journal of Information Technology, vol. 20, no. 6, pp. 911-918, November 2023. doi:10.34028/iajit/20/6/8.

[5] Dan Boneh, Trevor Perrin. *ecdsa: ECDSA cryptographic signature library (pure python).* PyPI, 2023. Available at: `https://pypi.org/project/ecdsa/`.

[6] Kostas P. Demetriou. *Post-Quantum Cryptography (pqcrypto).* GitHub repository, 2023. Available at: `https://github.com/kpdemetriou/pqcrypto`.

[7] Luke Beckwith, Duc Tri Nguyen, and Kris Gaj. *High-Performance Hardware Implementations of Lattice-Based Digital Signatures.* George Mason University, 2022. Available at: `https://example.com/path/to/document.pdf`.

[8] Alghamdi, S., & Almuhammadi, S. (2021). The Future of Cryptocurrency Blockchains in the Quantum Era. In Y. Xiang, Z. Wang, H. Wang, & V. Niemi (Eds.), *Proceedings - 2021 IEEE International Conference on Blockchain, Blockchain 2021* (pp. 544-551). Institute of Electrical and Electronics Engineers Inc.. `https://doi.org/10.1109/Blockchain53845.2021.00082`

[9] Ravinesh Chand, Maheswara Rao Valluri, and MGM Khan. (2021). Digital Signature Scheme over Lattices. *Journal of Cyber Security and Mobility*, vol. 10, no. 1, pp. 81-106. Available at: `https://www.researchgate.net/publication/357723623_Digital_Signature_Scheme_over_Lattices`.

[10] Nivedita Dey, Mrityunjay Ghosh, and Amlan Chakrabarti. *Quantum Solutions to Possible Challenges of Blockchain Technology.* University of Calcutta and HCL Technologies, October 2021. Available at: `https://arxiv.org/abs/2110.05321v1`.

[11] Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Peter Schwabe, Gregor Seiler, and Damien Stehlé. *CRYSTALS-KYBER: Algorithm specifications and supporting documentation*, January 2021.

[12] M. Egli, B. Grobauer, and P. S. T. Brito. *Quantum-Resistant Blockchain: A Systematic Literature Review.* IEEE Access, vol. 9, pp. 119401-119414, 2021. doi: 10.1109/ACCESS.2021.9478087. Available at: `https://ieeexplore.ieee.org/document/9478087`.

[13] Zhongxiang Zheng, Anyu Wang, and Lingyue Qin. *Rejection Sampling Revisit: How to Choose Parameters in Lattice-Based Signature.* Mathematical Problems in Engineering, vol. 2021, Article ID 9948618, 12 pages, June 7, 2021. doi:10.1155/2021/9948618.

[14] Sergi Ramos-Calderer, Emanuele Bellini, José I. Latorre, Marc Manzano, and Victor Mateu. *Quantum Search for Scaled Hash Function Preimages.* Technology Innovation Institute, Universitat de Barcelona, and National University of Singapore, September 2020. Available at: `https://arxiv.org/abs/2009.00621v1`.

[15] Tiago M. Fernández-Caramés and Paula Fraga-Lamas. *Towards Post-Quantum Blockchain: A Review on Blockchain Cryptography Resistant to Quantum Computing Attacks.* IEEE Access, 2020. doi:10.1109/ACCESS.2020.2968985. Available at: `https://www.researchgate.net/publication/357723623_Digital_Signature_Scheme_over_Lattices`.

[16] Vikas Srivastava, Anubhab Baksi, Sumit Kumar Debnath. *An Overview of Hash Based Signatures.* National Institute of Technology Jamshedpur, Nanyang Technological University, Singapore, 2020.

[17] Arne Schönbohm. *Quantum-Safe Cryptography – Fundamentals, Current Developments and Recommendations.* Federal Office for Information Security (BSI), April 2020. Available at: `https://www.bundesfinanzministerium.de/Web/EN/Issues/Public-Finances/stimulus-package-for-everyone/stimulus-package-for-everyone.html`.

[18] National Institute of Standards and Technology (NIST). *Post-Quantum Cryptography Standardization.* 2022. Available at: `https://csrc.nist.gov/projects/post-quantum-cryptography`.

[19] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. *CRYSTALS-Dilithium: A Lattice-Based Digital Signature Scheme.* CWI, Ruhr Universität Bochum, SRI International, IBM Research

– Zurich, Radboud University, IBM Research – Zurich and ETH Zurich, ENS de Lyon, 2017.

[20] Luke Beckwith, Duc Tri Nguyen, and Kris Gaj. *High-Performance Hardware Implementation of Lattice-Based Digital Signatures.* George Mason University, USA, 2022. Available at: `https://example.com/path/to/document.pdf`.

[21] Brandon Rodenburg, PhD, and Stephen P. Pappas, PhD. *Blockchain and Quantum Computing.* MITRE Technical Report, June 2017. Project No.: 25SPI050-12. Available at: `https://www.mitre.org/publications/technical-papers/blockchain-and-quantum-computing`.

[22] Aviad Rubinstein, Nick Guo. *Lecture 14: Learning with Errors.* CS 354: Unfulfilled Algorithmic Fantasies, Stanford University, May 15, 2019.

[23] Wei-Meng Lee. *Beginning Ethereum Smart Contracts Programming: With Examples in Python, Solidity, and JavaScript.* Apress, Ang Mo Kio, Singapore, ISBN: 978-1-4842-5085-3, `https://doi.org/10.1007/978-1-4842-5086-0`, 2019.

[24] Martin R. Albrecht, Rachel Player, and Sam Scott. *On the concrete hardness of Learning with Errors.* Information Security Group, Royal Holloway, University of London, 2015.

[25] C.-Y. Li, X.-B. Chen, Y.-L. Chen, Y.-Y. Hou, and J. Li. *A new lattice-based signature scheme in post-quantum blockchain network.* IEEE Access, vol. 7, pp. 2026–2033, 2018.

[26] Maharage Nisansala Sevwandi Perera and Takeshi Koshiba. *Achieving Full Security for Lattice-based Group Signatures with Verifier-local Revocation.* Graduate School of Science and Engineering, Saitama University, Japan and Faculty of Education and Integrated Arts and Sciences, Waseda University, Tokyo, Japan, 2023.

[27] Mavroeidis, Vasileios, et al. "The Impact of Quantum Computing on Present Cryptography." *International Journal of Advanced Computer Science and Applications*, vol. 9, no. 3, 2018.

[28] Zhwan Mohammed Khalid and Shavan Askar. *Resistant Blockchain Cryptography to Quantum Computing Attacks.* International Journal of Science and Business, vol. 5, no. 3, pp. 116-125, 2021. doi:10.5281/zenodo.4497732. Available at: `https://www.researchgate.net/publication/349054132_Resistant_Blockchain_Cryptography_to_Quantum_Computing_Attacks`.

[29] Shi Bai and Steven D. Galbraith. *An improved compression technique for signatures based on learning with errors.* In Josh Benaloh, editor, Topics in Cryptology – CT-RSA 2014, pages 28–47, Cham, 2014. Springer International Publishing.

[30] Shi Bai, Léo Ducas, Eike Kiltz, Tancrede Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. *CRYSTALS-Dilithium: Algorithm specifications and supporting documentation*, October 2020.

[31] P. W. Shor. *Algorithms for quantum computation: discrete logarithms and factoring.* In Proceedings 35th annual symposium on foundations of computer science. IEEE, 1994, pp. 124–134.

[32] L. K. Grover. *A fast quantum mechanical algorithm for database search.* In Proceedings of the twenty-eighth annual ACM symposium on Theory of computing, 1996, pp. 212–219.

[33] Vadim Lyubashevsky. *Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures.* Department of Computer Science, Tel-Aviv University, ASIACRYPT 2009, pp. 598-616, Springer, 2009.

[34] Oded Regev. *On Lattices, Learning with Errors, Random Linear Codes, and Cryptography.* School of Computer Science, Tel-Aviv University, May 2, 2009.

[35] Erdem Alkim, Léo Ducas, Thomas Pöppelmann, and Peter Schwabe. *Post-quantum key exchange – a new hope.* Cryptology ePrint Archive, Report 2015/1092, 2015. Available at: `https://ia.cr/2015/1092`.

[36] Eiichiro Fujisaki and Tatsuaki Okamoto. *Secure Integration of Asymmetric and Symmetric Encryption Schemes.* Journal of Cryptology, 26(1):80–101, January 2013.

[37] Chris Peikert and Zachary Pepin. *Algebraically structured lwe, revisited.* Cryptology ePrint Archive, Report 2019/878, 2019. Available at: `https://ia.cr/2019/878`.

[38] Lily Chen, Stephen Jordan, Yi-Kai Liu, Dustin Moody, Rene Peralta, Ray Perlner, and Daniel Smith-Tone. *Report on Post-Quantum Cryptography.* Technical Report NIST Internal or Interagency Report (NISTIR) 8105, National Institute of Standards and Technology, April 2016.

[39] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. *A Modular Analysis of the Fujisaki-Okamoto Transformation.* In Yael Kalai and Leonid Reyzin, editors, Theory of Cryptography, volume 10677, pages 341–371. Springer International Publishing, Cham, 2017. Series Title: Lecture Notes in Computer Science.

[40] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. *On ideal lattices and learning with errors over rings.* In Henri Gilbert, editor, Advances in Cryptology – EUROCRYPT 2010, pages 1–23, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[41] Adeline Langlois and Damien Stehle. *Worst-case to average-case reductions for module lattices.* Cryptology ePrint Archive, Report 2012/090, 2012. Available at: `https://ia.cr/2012/090`.

[42] Vadim Lyubashevsky. *Lattice signatures without trapdoors.* In David Pointcheval and Patrick Schaumont, editors, EUROCRYPT 2012 – 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012., volume 7237 of Lecture Notes in Computer Science, pages 738–755, Cambridge, United Kingdom, April 2012. Springer.

[43] Jiaxin Pan and Benedikt Wagner. *Lattice-based Signatures with Tight Adaptive Corruptions and More.* PKC 2022, January 5, 2022. NTNU - Norwegian University of Science and Technology, Trondheim, Norway and CISPA Helmholtz Center for Information Security, Saarbrücken, Germany.

[44] Shi Bai and Steven D. Galbraith. *An improved compression technique for signatures based on learning with errors.* Department of Mathematics, University of Auckland, New Zealand, 2013.

[45] Ugwuishiwu, C. H., et al. *An overview of quantum cryptography and Shor's algorithm.* Int. J. Adv. Trends Comput. Sci. Eng, Vol. 9, No. 5, 2020.

[46] Vikas Srivastava, Anubhab Baksi, Sumit Kumar Debnath. *An Overview of Hash Based Signatures.* National Institute of Technology Jamshedpur, Nanyang Technological University, Singapore, 2020.

[47] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition.* Cambridge University Press, 2010.

[48] Nujud Alyami and Enas Aljuhani. *Quantum-Safe Blockchain Project.* GitHub repository, Available at: `https://github.com/nujudaly/Quantum-Safe-Blockchain-Project.git`.