UNIVERSITÉ DE GENÈVE

MASTER'S THESIS

Exploring Quantum-Resistant Signatures for Ethereum's Hybrid Blockchain

Author: Dany A. DARGHOUTH

E-mail: dany.al-moghrabi@etu.unige.ch

June 2025

UNIVERSITÉ DE GENÈVE

FACULTÉ DES SCIENCES
Département d'informatique

Acknowledgements

Writing this thesis has been a long and challenging journey, one that I could not have completed without the help, encouragement, and inspiration of many people. I am deeply grateful to all those who supported me along the way.

I would first like to thank my parents, my best friend and father Ahmad, who sat me in front of a computer before I could walk. That simple gesture lit the spark that would later become a lifelong passion. And my mother Patricia, a model of perseverance and resilience, whose unwavering love and strength have carried me through more moments than I can count. This achievement is as much theirs as it is mine. To the rest of my family, who have always believed in me and encouraged me to pursue my dreams, thank you for your constant support and for always being there.

I am deeply grateful to Professor E. Solana, whose guidance and insight shaped this thesis and left a lasting mark on my academic journey. It was through one of your courses that I first discovered my passion for this field. Your thoughtful teaching and mentorship not only opened the door but gave me the confidence to walk through it. Thank you for your support, patience, and for always encouraging me to think deeper.

To my grandfathers Jose and Salah, both very different, both deeply inspirational. One, a stonemason, whose legacy of craftsmanship and hard work continues to inspire me. The other, an Arabic teacher, lover of literature and philosophy, who nurtured my love of learning and taught me the value of precision and clarity. I wish I could have spent more time with them both, but I carry their lessons with me every day.

There are so many others whose presence made a difference, even in small ways. I wish I could name every single one of you here. Please know that your support has not gone unnoticed, and I am sincerely thankful.

Abstract

The development of quantum computing poses a serious threat to classical cryptographic primitives, particularly those used in securing blockchain systems such as Ethereum. As these systems rely on digital signatures for transaction authentication and consensus, transitioning to post-quantum secure algorithms is essential to maintaining long-term integrity and trust.

This thesis investigates the feasibility, complexity, and efficiency of integrating post-quantum digital signature schemes into the Ethereum blockchain. Focusing on three NIST-standardized candidates, SLH-DSA (SPHINCS+), ML-DSA, and FALCON, it analyzes their cryptographic properties, performance characteristics, and compatibility with Ethereum's hybrid Proof-of-Stake architecture. ML-DSA was identified as the most suitable candidate, offering a favorable balance between security guarantees and computational efficiency.

A multi-phase transition strategy was proposed and implemented in a simulation-based environment to assess the impact of these schemes on gas cost, signature size, and consensus performance. The results indicate that Ethereum can incorporate post-quantum signatures without prohibitive performance penalties, especially when compared to post-quantum Proof-of-Work alternatives. The thesis also explores how lattice-based signatures may support privacy enhancements, particularly in combination with zero-knowledge proof systems.

These findings hope to contribute to the broader effort to prepare decentralized platforms for a post-quantum future, offering a path toward secure, scalable, and privacy-preserving blockchain ecosystems.

Summary

The grammar, spelling and formatting of this report have been reviewed using AI tools such as Quillbot and Grammarly, explaining a possible high detection score when using AI detection tools.

1. Introduction

Recent and upcoming breakthroughs in the field of quantum computing present a considerable challenge to the security of modern cryptographic schemes, namely in blockchain technology. Ethereum[1] as one of the leading platforms relies on primitives such as the Elliptic Curve Digital Signature Algorithm (ECDSA) for transaction verification and network security. However such primitive are vulnerable to quantum attacks such as Shor's algorithm [1] possibly rendering them obsolete in the near future. As Ethtereum transitions from proof-of-work to proof-of-stake, exploring post-quantum resistant solutions becomes a necessity to ensure the long-term security of the platform.This thesis investigates the feasibility, complexity, and efficiency of integrating quantum-resistant signature schemes into Ethereum's blockchain, addressing both theoretical and practical considerations.

A proof-of-concept implementation is proposed along with this study, and available at *https://gitlab.unige.ch/Dany.Al-Moghrabi/exploring-ethereum-post-quantum-signatures*

1.1 Context

Blockchain technology has established itself as a foundational innovation in decentralized systems, with applications spanning finance, governance, and data security. Among the various blockchain platforms, Ethereum has distinguished itself by enabling smart contracts, self-executing programs that operate without intermediaries, powering decentralized finance (DeFi), non-fungible tokens (NFTs), and various solutions as such, and especially as Ethereum transitions to a hybrid Proof of Work (PoW) and Proof of Stake (PoS) model, the long-term security of its cryptographic infrastructure is a critical issue, as of today, Ethereum relies on cryptographic primitives that are vulnerable to quantum attacks, such as the Elliptic Curve Digital Signature Algorithm (ECDSA) used for transaction signing. However, as the field of quantum computing develops, the cryptographic basis of these algorithms face the risk of being broken.

Quantum computers use principles of superposition and entanglement to perform computations that can not be done by classical machines. Namely, Shor's algorithm[1], a quantum algorithm for integer factorization and discrete logarithm computation, puts the security of RSA and elliptic curve cryptography (ECC), including Ethereum's ECDSA-based signatures at high risk.
Though quantum computers are still at the early stages of development, industry leaders such as Google [2] and IBM [3] have demonstrated significant progress, leading experts to believe that the threat of quantum computing to current cryptographic standards is going to be a real and practical issue in the foreseeable future.

In order to address this threat, the National Institute of standards and technology (NIST) have started a selection process within post-quantum cryptography algorithms to find secure and efficient alternatives to current schemes. In addition, the Ethereum foundation acknowledged the importance of post quantum cryptography and has made research and work on developing quantum-resistant schemes.

While post-quantum cryptography has been an active research field, most studies focus on general transitions rather than the unique challenges posed by Ethereum's hybrid model. This thesis aims to bridge that gap by exploring the feasibility, complexity, and efficiency of integrating quantum-resistant signatures into Ethereum.

1.2 Literature Review

1.2.1 Post-Quantum Cryptography

The first concerns about the impact of quantum computing on cryptography were raised by Peter Shor in 1994 [1], when he introduced his quantum algorithm for factoring large integers and computing discrete logarithms in polynomial time . This algorithm, if implemented on a large enough quantum computer, would break the security of widely used cryptographic schemes such as RSA and ECDSA, which are based on the hardness of these problems, thus rendering them and the systems that rely on them insecure.

As mentioned in the previous section, quantum computing, although still in its early stages of development, is rapidly evolving, with breakthroughs emerging every year. As such, the need for cryptographic schemes that are secure against quantum attacks is becoming more and more pressing.

[1] Link to the Ethereum website https://ethereum.org/fr/

1.2.2 Impact of Quantum Computing on Cryptography and Blockchain

Quantum computing is expected to have a significant impact on cryptography and blockchain systems. Cryptographic primitives considered secure as of now become vulnerable to quantum algorithms, such as Shor's[1] and Grover's[4] algorithms. This means that the security of widely used cryptographic schemes, such as RSA, ECDSA, and SHA-256, is compromised in the presence of a sufficiently powerful quantum computer.

Namely blockchains will suffer from the vulnerabilities as they rely on such primitives for their security, Bitcoin being an example [5], especially as public key exposure (an inherent property of blockchains), can be used to reconstruct private keys and forge signatures as the schemes used can be broken using quantum algorithms. One can reasonably consider that Ethereum will suffer from the same vulnerabilities, as it relies on similar cryptographic primitives for its security.

An other non-negligible threat to these systems is the possibility of having encrypted sensitive data encrypted with quantum-vulnerable primitives being harvested and stored until a quantum computer is available to decrypt it. This period of times is known as the ¨risk-window¨, putting the immutable nature of blockchains at risk[6].

1.2.3 Standardization Process

Facing the new challenges posed by quantum computing, the National Institute of Standards and Technology (NIST) has initiated a process to develop cryptographic schemes able to resist quantum attacks[7]. The goal being to find replacement to widely used schemes such as RSA, ECDSA... that are namely vulnerable to Shor's Algorithm.

The process started in 2016, and is organized in several rounds, the first round was dedicated to collecting and analyzing the different proposals, the second round was dedicated to selecting the most promising candidates, and the third round was dedicated to further analyzing the selected candidates [8]. These candidates were then standardized and renamed.[9][10] (Falcon was also renamed and is expected to be standardized as FIPS 206, however its complete standardization was not complete as of June 2025[11]) As of today, they are the standards as they remain the most promising candidates for post-quantum cryptography.

Later in the process, NIST called for the submission of additional proposals, looking for new candidate possibly outperforming the existing ones, this new selection process, focused on signature schemes and separate from the "main" process, has reached the conclusion of its first round in 2024 [12], selecting 14 to advance to the second round, namely some code-based[13] candidates such as CROSS[14] and LESS[15], both relying on the hardness of decoding problems in structured codes and employing zero-knowledge techniques to achieve compact and secure signatures.

The selected candidates, although promising are still at an early stage of development, as such they are not considered as standards yet. And therefore, the focus of this work will be on the schemes that have been standardized by NIST.

Additionally to the proposed signature, NIST proposals also cover key encapsulation mechanisms (KEMs) and public key encryption schemes, namely Hamming Quasi-Cyclic [16] based on the difficulty of decoding a scrambled message mixed with random errors, a problem believed to be resistant to quantum attacks. Its quasi-cyclic structure allows for smaller key sizes compared to earlier code-based systems, enhancing practicality for real-world use, however as this work focuses on signature schemes, these KEMs although worth mentioning as the may give insight into possible bases for future candidates; will not be covered in this work.

1.2.4 Post-Quantum Signature Candidates

This section aims at presenting the different post-quantum signature schemes that are considered as standards and could possibly be implemented for Ethereum, a detailed explanation of each scheme will be provided in Section 3 : PqC Signatures Schemes for Ethereum.

Hash-Based Signatures (HBS) : Hash Based Signatures schemes provide an alternative to the traditional public key cryptography. The first works were done by Lamport in 1979 [17] [18]. The security of the scheme is based on the security of the underlying hash function.

HBS scheme can be relevant candidates for post-quantum cryptography, as they are considered secure due to their reliance on the security of cryptographic hash functions, which have been extensively studied and are believed to be resistant to quantum attacks. [19]. Namely, **SPHINCS+** [20] developed by Bernstein et al., is a stateless signature scheme that allows for high throughput and strong security guarantees, as its security is based on the security of the underlying hash function [21], and has been retained as one of the three finalists in the NIST post-quantum cryptography signature standardization process.[8] Other HBS candidates have been considered by NIST, such as XMSS [18] but have not been retained as finalists. After its selection, this scheme was renamed to **SLH-DSA (Stateless Hash-Based Digital Signature Algorithm)**[10]

Lattice-Based Signatures (LBS) : Lattice-based cryptography is a type of public-key cryptography based on the hardness of learning problems on lattices. The fundamental work on lattice-based signature schemes was done by Ajtai in 1996 [22]. The security of the scheme is based on the hardness of the underlying lattice problem, the first cryptographic scheme whose security was proven is the scheme proposed by Regev in 2005 [23].

NIST has retained two lattice-based signature schemes as finalists in the NIST post-quantum cryptography signature standardization process, **CRYSTALS-Dilithium** and **FALCON** [8]. These schemes are considered secure and efficient, and have been extensively studied in the literature.

CRYSTALS-Dilithium [24], developed by Ducas et al., is one of the Lattice based schemes retained, it is based on the hardness of the Ring-LWE problem, and is considered secure and efficient.This secheme then became **ML-DSA (Module-Lattice-Based Digital Signature Algorithm)**[9]after standardization.

The second Latttice based candidate is FALCON (Fast Fourier Lattice-based Compact Signatures over NTRU) [25](later known as **FFT over NTRU-Lattice-Based Digital Signature Algorithm**), developed by Fouque et al., based on NTRU lattices and uses Fast Fourier Transform (FFT) for key generation and signing. This scheme is based on the shortest vector problem (SVP) in NTRU lattices, a problem believed to be resistant to both classical and quantum attacks.

Overall these schemes provide a solid base to consider possible post-quantum cryptographic solutions for Ethereum, as they are considered secure and efficient, and have been extensively studied in the literature. However, other schemes not considered by NIST can also be considered as candidates for post-quantum cryptography in the context of blockchains, such as isogeny based schemes like SQISign [26] having the advantage of outputting small keys and signatures; however these schemes seem to be outperformed by the NIST finalists in terms of efficiency.[27] and will therefore not be considered in this work due the demands of the blockchain context.

1.3 Methodology

1.3.1 Research Approach

Both a theoretical approach to the topic and an experimental implementation and performance evaluation are proposed, in order to understand and evaluate the candidate algorithms and a possible implementation strategy. The theoretical part consists of a study of the proposed signature algorithms and their mathematical foundations, as well as their efficiency, and suitability for blockchain environments.

The experimental aspect consists of a proof-of-concept (PoC) implementation of how Ethereum would look like if selected post-quantum signature schemes were put in place. While the PoC will not be a full-scale Ethereum implementation, it will serve as a simplified model for evaluating the impact of such an integration integration.

1.3.2 Algorithm Selection

In order to select the most suitable candidate, the leading candidate algorithms proposed by the latest NIST status report [8] will be evaluated based on the following criteria:

- Security: The security of the algorithm against known attacks.
- Resistance to Quantum Attacks: The algorithm's resistance to quantum attacks.
- Signature size: The size of the signature generated by the algorithm.
- Key size: The size of the public and private keys generated by the algorithm.
- Computational efficiency; The time taken to generate a signature and verify it.

The proposed algorithms will also be evaluated based on its compatibility with the Ethereum blockchain and its ability to be integrated into the existing Ethereum infrastructure, although this consideration is also dependent on the implementation strategy. A single scheme meeting the above criteria, will be selected as the basis for the PoC implementation.

1.3.3 Implementation Strategy

The implementation strategy will be based on the selected post-quantum signature scheme, and will focus on the integration of the scheme into the Ethereum 2.0 blockchain. The implementation will be designed to be backward compatible with the existing model, and will also have to take the security of past transactions into consideration. In addition to these concerns, the implementation will also measure the integration's impact on the Ethereum network's performance, and security.

1.3.4 Evaluation Methodology

Evaluation of the system will consider security, performance, and scalability. For security evaluations, the system will be evaluated based on the security properties of the selected post-quantum signature scheme, as well as the system's resistance to quantum attacks, and compared to the current Ethereum blockchain as well as classic proof-of-work one for reference. For performance, time taken to generate a signature and verify it will be measured, as well as the overall throughput of the system.

1.3.5 Research Limitations

Although this study aims at providing a thorough evaluation of the impact of post-quantum signature schemes on the Ethereum blockchain, some limitations are to be considered. The proof-of-concept (PoC) implementation will be conducted on a simplified Ethereum blockchain model, meaning that certain real-world factors such as network congestion, miner incentives, and full Ethereum client integration, will not be fully taken into account. In addition, the current state of the Ethereum blockchain is not optimized for some of the schemes, namely lattice-based schemes, thus impact on the exact impact of the integration on gas costs in a production environment may require further testing on Ethereum testnets. Another key limitation is the assumption that a gradual transition away from ECDSA verification is feasible, whereas in reality, the process may face adoption resistance form the community as well as unforeseen challenges. Despite these limitations the study aims to provide a comprehensive evaluation of the impact of post-quantum signature schemes on the Ethereum blockchain, and to provide insights into the feasibility of such an integration.

2. Cryptographic Background

2.1 Private and Public Key Cryptography

Exchanging information while preserving confidentiality is a fundamental problem in cryptography, and it is the principal motivation for the development of public and private key cryptography.

2.1.1 Private Key Cryptography

Definition and Concept:

- A private key cryptography system is a system in which the same key (K) is used for both encryption and decryption. The key is kept secret and is shared only between the sender and the recipient.
- When encrypting a message M using a private key K, the sender uses an encryption function E to produce a ciphertext c:
$$c = E_K(M)$$
  The recipient then uses a decryption function D to decrypt the message using the same key K:
$$M = D_K(c)$$
- **Main advantages of private key cryptography :**
  - **Confidentiality:** The message is encrypted using a secret key, and only the users having the secret can decrypt it using the same key.

- **Speed:** Private key cryptography is generally faster than public key cryptography, as it requires less computational power.

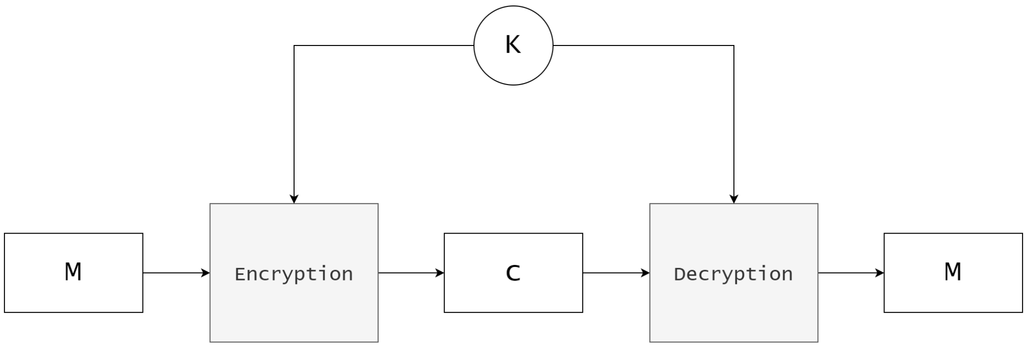

Figure 1: Private Key Cryptography

2.1.2 Public Key Cryptography

Definition and Concept:

- A public key cryptography system is a system wherein each user has two mathematically related keys, a public key (K_p) (for encryption) and a private key (K_s) (for decryption). The public key, being shared with every other user, is used to encrypt messages addressed to the user, who then uses their own private key (kept secret) to decrypt the message.
- For instance, considering two users, A and B, respectively having public keys K_{pA} and K_{pB}, and private keys K_{sA} and K_{sB} and using an encryption function E and a decryption function D, in the case where A wants to send an encrypted message M to B, A would encrypt M using K_{pB}, and B would decrypt the message using K_{sB}:

$$c = E_{K_{pB}}(M)$$

$$M = D_{K_{sB}}(c)$$

- **Main advantages of public key cryptography:**
  - **Confidentiality:** The message is encrypted using the public key of the recipient, and only the recipient can decrypt it using their private key.
  - **Authentication:** The sender can sign the message using their private key (as explained later in Section 2.3: Digital Signatures), and the recipient can verify the signature using the sender's public key.
  - **Non-repudiation:** The sender cannot deny having sent the message, as the recipient can verify the signature using the sender's public key.

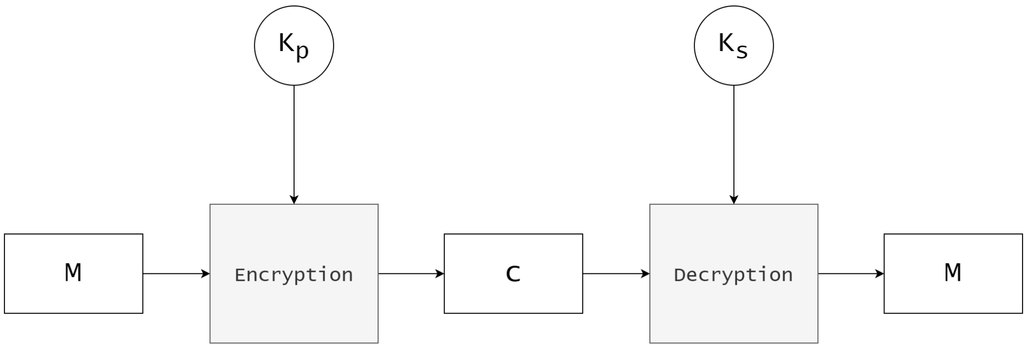

Figure 2: Public Key Cryptography

The main examples of public key cryptography systems are the RSA and Elliptic Curve cryptography systems, however such schemes require the use of longer keys in order to provide satisfactory security, so in practice they are used to provide a shared secret key for symmetric key cryptography systems.

2.2 Hash Functions

Definition and Concept:

A hash function $H(m), H : \{0,1\}^* \rightarrow 0,1^n$ is a function that maps an input (message) of arbitrary length, to a fixed size output (digest or hash). Hash functions have properties making them a fundamental building block in cryptography, and especially in blockchain technology:

- **Pre-image resistance:** Given a hash value h, it is computationally impossible to find any message m such that $H(m) = h$.
- **Second pre-image resistance:** Given a message m, it is computationally impossible to find another message m' such that $H(m) = H(m')$.
- **Collision resistance:** It is computationally impossible to find two different messages m and m' such that $H(m) = H(m')$.
- **Deterministic:** For a given input, the output is always the same.
- **Efficient:** The computation of the hash value is fast.

2.2.1 SHA-256

SHA-256 is a member of the SHA-2 family of hash functions [28] developed by NIST, that produces a 256-bit digest, and is widely used in proof-of-work blockchains and digital signatures as it shows a strong collision resistance, and is computationally efficient.

Algorithm:

1. **Padding:** The message is padded to a length that is a multiple of 512 bits.
2. **Initial Hash Values:** 8 Initial hash values are chosen based on the square roots of prime numbers.
3. **Processing:** The message is processed in 512-bit blocks, and the hash value is updated after each block.
4. **Output:** The final hash value is the concatenation of the 8 32-bit hash values as a 256-bit digest.

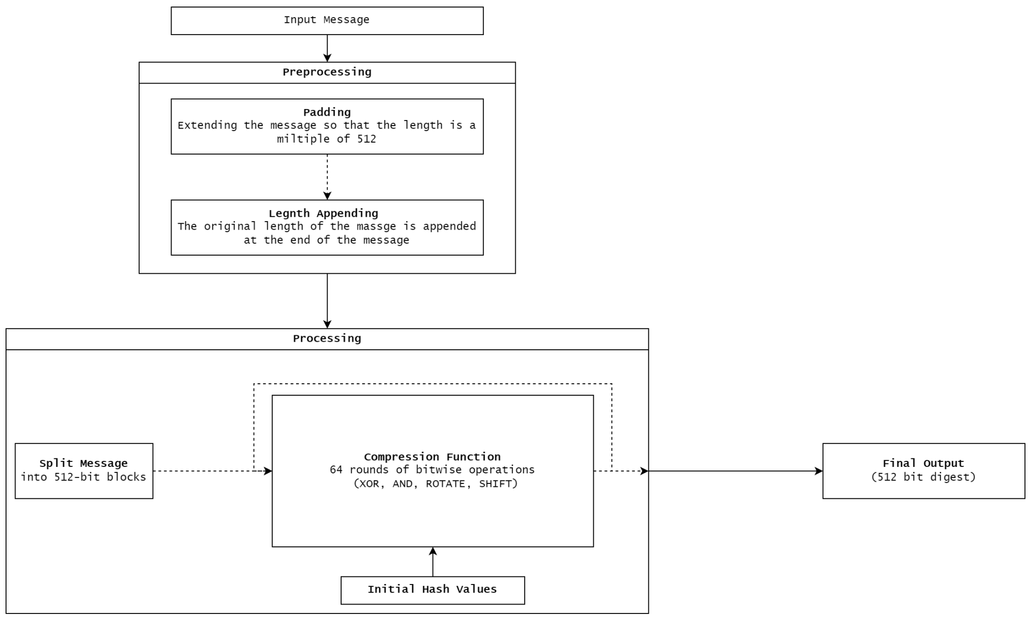
Input Message
Preprocessing
Padding
Extending the message so that the length is a miltiple of 512
Legnth Appending
The original length of the massge is appended at the end of the message
Processing
Split Message into 512-bit blocks
Compression Function
64 rounds of bitwise operations (XOR, AND, ROTATE, SHIFT)
Initial Hash Values
Final Output (512 bit digest)

Figure 3: SHA-256 Algorithm

Although not used in Ethereum for block hashing, SHA-256 is used in its proof-of-work algorithm, Ethash[29], and in digital signatures.

2.2.2 KECCAK

KEECCAK[30] is the family of hash functions that won the NIST SHA-3 competition, and is based on the sponge construction. It is used in Ethereum for block hashing, and in digital signatures and produces a 256-bit digest.

Property	SHA-256	KECCAK
Rounds	64	24
Performance	Faster on general hardware	More efficient on hardware-accelerated implementations.
Usage in Ethereum	Used in PoW mining	Used for block hashing, address generation and state hashing.

Figure 4: Comparison of SHA-256 and KECCAK hash functions

The particular construction of the KECCAK hash function allows for variable output sizes through its multi-phase approach, in the absorbing phase, the message is XORed with a representation of the current state, and in the squeezing phase, the output is generated by reading the state.

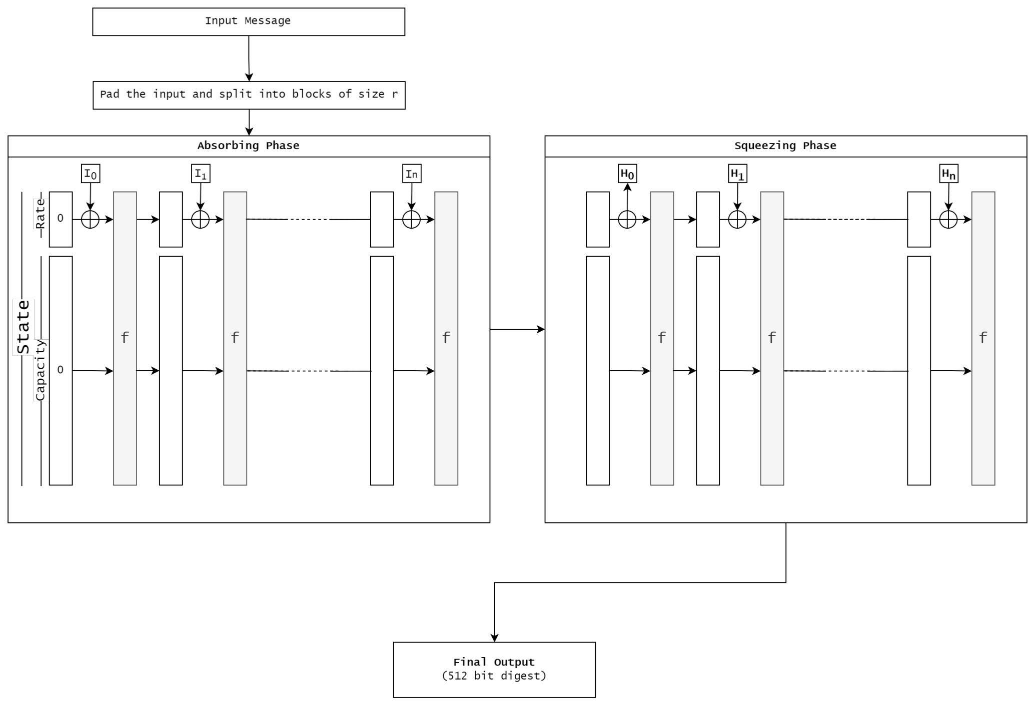

Figure 5: KECCAK Algorithm

2.3 Digital Signatures

In order to provide a from of authentication and ensure integrity in communications, digital signatures are used. These mechanisms relying on asymmetric cryptography also allow for non-repudiation, meaning that the sender cannot deny having sent the message. Digital signatures are based on public key cryptography, where a pair of keys (public and private) is generated. The private key is used to sign the message, while the public key is used to verify the signature. The security of digital signatures relies on the difficulty of certain mathematical problems, such as factoring large numbers or solving discrete logarithms.

Some of the most widely used algorithms include RSA, based on the difficulty of factoring large integers, and DSA, which relies on the discrete logarithm problem. And their applications reach from secure emails to blockchain technology, where they are used to verify transactions, ensure the integrity of the data stored on the blockchain and authenticate users.

2.3.1 Digital Signature Algorithm (DSA)

DSA [31] is a widely used digital signature algorithm that was proposed by the National Institute of Standards and Technology (NIST) in 1991. It is based on the discrete logarithm problem and is designed to provide a secure method for generating and verifying digital signatures. DSA is commonly used in various applications, including secure email, code signing, and digital certificates. The signature generation process involves the following steps:

1. **Key Generation:** A pair of keys (private and public) is generated. The private key is a randomly chosen integer, while the public key is derived from the private key using modular exponentiation.
2. **Hashing:** The message to be signed is hashed using a cryptographic hash function, such as SHA-256. The hash value is a fixed-size representation of the message.
3. **Random Number Generation:** A random number (k) is generated for each signature. This number must be unique and unpredictable to ensure security.
4. **Signature Generation:** The signature is created using the private key, the hash of the message, and the random number.
5. **Signature Verification:** The recipient uses the public key to verify the signature by checking if it matches the hash of the message.

This algorithm is however not without limitations, as it requires large key sizes to ensure security, and the random number (k) must be unique for each signature to prevent attacks. Additionally, DSA is not suitable for signing small messages, as it requires a larger hash size than other algorithms like RSA.

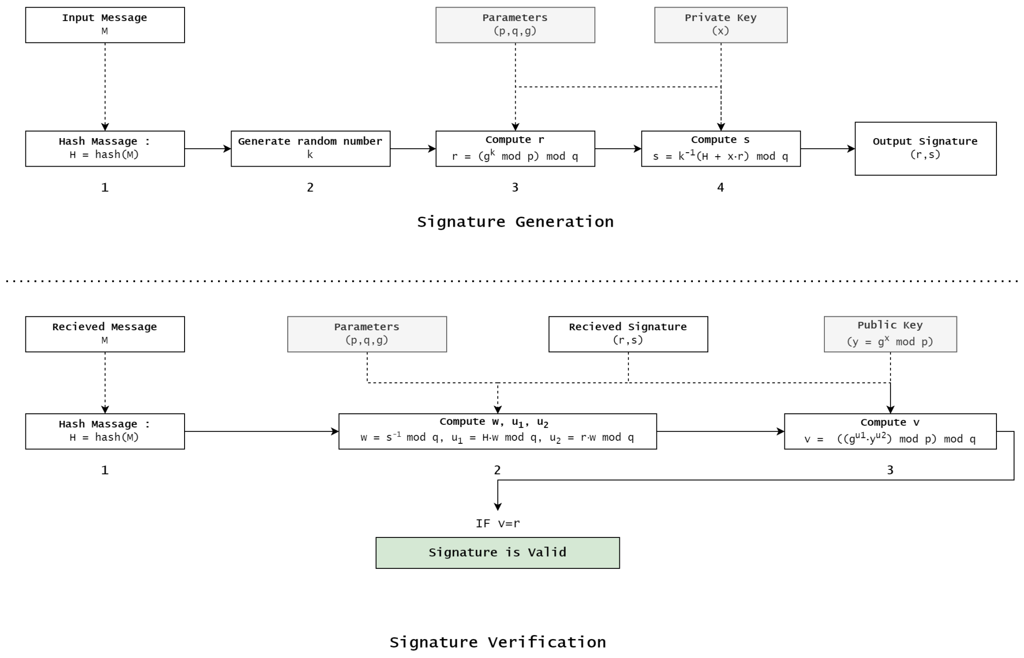

Figure 6: Digital Signature Algorithm (DSA) process

2.3.2 ECDSA

To mitigate the limitations posed by DSA, ECDSA[32] was introduced as an alternative. ECDSA is based on elliptic curve cryptography (ECC)[33], [34], which provides a level of security that is at least equivalent with smaller key sizes compared to traditional DSA, by using the mathematical properties of elliptic curves. The general steps for signature generation and verification are similar to DSA, but with the following differences:

1. **Key Generation:** The private key is a randomly chosen integer, and the public key is derived from the private key using elliptic curve point multiplication.
2. **Signature Generation:** The message is hashed, and a random number (k) is generated. The signature is created using the private key, the hash of the message, and the random number, but with elliptic curve operations.
3. **Signature Verification:** The recipient uses the public key to verify the signature by checking if it matches the hash of the message using elliptic curve operations.

Property	**DSA**	**ECDSA**
Underlying problem	Discrete Logarithm Problem	Elliptic Curve Discrete Logarithm Problem
Key Size	2048 bits	512 bits
Signature Size	320 bits	512 bits
Performance	Slower than ECDSA, especially in verification	Faster in Key generation, signing and verification
Security	Vulnerable to Shor's algorithm	

Figure 7: Comparison of DSA and ECDSA

Thanks to its improved security and efficiency, ECDSA has become the standard for digital signatures in many applications, including Ethereum. However, ECDSA is also vulnerable to quantum attacks, as Shor's algorithm can efficiently solve the underlying mathematical problems, making it necessary to explore post-quantum alternatives.

2.4 Blockchains

Blockchains are a type of distributed ledger technology (DLT), which is a decentralized database that allows multiple parties to maintain a shared record of transactions without the need for a central authority. These systems are immutable, providing tamper-proof records. These systems are based on the Merkle tree structure[35], a tree-like structure that allows for efficient verification of data integrity. Each leaf node in the Merkle tree represents a transaction, while the non-leaf nodes represent the hash of their child nodes.
A block is a structure that in its simplest form contains a list of transactions, a timestamp, and a reference to the previous block, thus forming a cryptographically linked chain, ensuring integrity znd historical transparency.
Blockchains are by design, relying on nodes (i.e. actors connected to the network) to function, each addressing a specific role, full nodes that store the entire blockchain, light nodes that store only a subset of the blockchain (the block headers), and miners/validators that validate transactions and create new blocks.

2.4.1 Consensus Mechanisms

A challenge in blockchain systems is to determine the validity and order of transactions. Consensus mechanisms are implemented to address this challenge, ensuring that all nodes in the network agree on the state of the blockchain.

Proof of Work (PoW)

Traditional blockchains, such as Bitcoin, use a consensus mechanism called Proof of Work (PoW)[36], in which in order to be able to add a block to the chain a miner must be the first to solve a complex mathematical problem, which requires significant computational power. Usually, the problem involves finding a nonce (a random number) that, when combined with the block's data and hashed, produces a hash that meets certain criteria (e.g., starts with a certain number of leading zeros). The first miner to find a valid nonce broadcasts the new block to the network, and other nodes verify its validity before adding it to their copy of the blockchain.
This consensus mechanism is energy-intensive and can lead to centralization, as miners with more computational power have a higher chance of solving the problem and adding new blocks. Additionally, PoW is vulnerable to 51% attacks, where a malicious actor gains control of more than half of the network's computational power, allowing them to manipulate the blockchain. These limitations have led to the development of alternative consensus mechanisms, such as Proof of Stake (PoS).

Proof of Stake (PoS)

Proof-of-Stake (PoS) [37] is a consensus mechanism, addressing some of the problems posed by PoW. Instead of having to be the first to complete a computationally intensive challenge, validators are are chosen through a probabilistic process influenced by the amount of cryptocurrency they hold and are willing to "stake" as collateral. The more coins a validator stakes, the higher the chance of being selected to validate transactions and create new blocks. This process is less energy-intensive than PoW, as it does not require massive computational power, when PoW's energy consumption stems from the need to solve complex mathematical problems, PoS relies on the validators' stake as a form of collateral, thus reducing the energy consumption, the only computation required is the selection of the validator and the validation of the transactions.

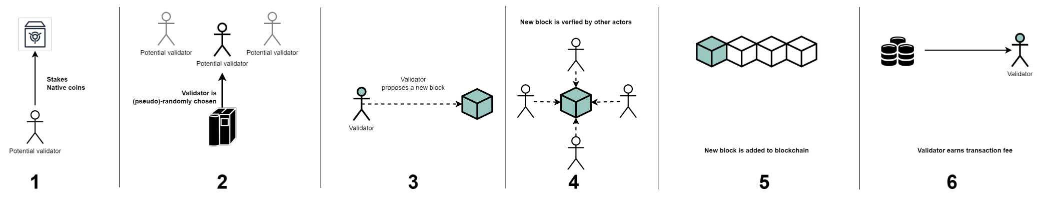

Figure 8: Proof of Stake (PoS) process

However, Pos is not without its limitations, the easiness of creating blocks lead to issues such long-range and nothing-at-stake attacks, in addition, such a system, is heavily reliant on human behavior and trust in the validators, as they can choose to act maliciously (e.g. bribe attacks)[2].

2.4.2 ETH 2.0. Hybrid Blockchain Model

Through its transition to Ethereum 2.0, Ethereum is moving from a classical PoW model to a hybrid model combining PoW and PoS mechanisms.

To address the scalability limitations inherent to the original Ethereum, Ethereum 2.0 introduces sharding—a form of horizontal database partitioning that splits the state and transaction processing across multiple chains (shards), all coordinated by the beacon chain. This allows transactions to be processed in parallel, highly increasing throughput. This transition, divided into phases will lead to a blockchain that is an entanglement of different chains (or layers) :

- **The Main chain** (formerly Ethereum 1.0 also called anchor or execution Layer) where smart contracts are deployed and transactions are executed, also providing staking and finality for the Beacon chain and the Shard chains.
  - Responsible for account balances and transaction history.
  - Maintains backward compatibility with Ethereum 1.0. for NFTs, dApps, and other smart contracts.
- **The Beacon chain** (or coordination layer) responsible for managing the PoS consensus mechanism and the network's validators.
  - Tracks validators and their stakes.
  - Probabilistically selects validators to propose and attest to new blocks.
  - Organizes validators into committees for finality and security.
  - Stores validator rewards and penalties (based on performance and honesty).
- **The Shard chains** (or data layer),responsible for processing transactions and storing data, each shard being able to process transactions in parallel, increasing the network's scalability. Shard chains periodically submit "crosslinks" (compressed summaries of their states) to the Beacon Chain, enabling synchronization and data availability checks.

[2]A more detailed explanation of the PoS consensus mechanism and its limitations can be found here :https://github.com/Dany-Drgh/pos-eth2.0

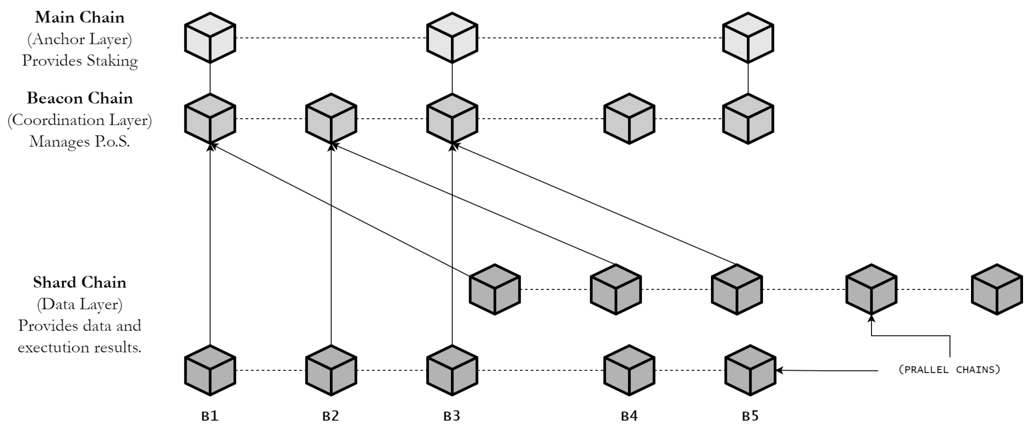

Figure 9: ETH 2.0 Hybrid Blockchain Model

Ethereum continues to rely on the Keccak-256 hash function and the ECDSA signature algorithm. The integration of post-quantum cryptographic (PQC) algorithms, particularly in the signature scheme layer, has therefore become a subject of growing importance. Ethereum 2.0's modular design potentially facilitates such cryptographic upgrades with less disruption compared to monolithic blockchains.

2.5 Lattice-Based Cryptography

2.5.1 Lattices

This section rather than giving a full explanation of lattices and their properties, aims only at providing a general overview of the mathematical background of lattices and their properties as some of the signature schemes presented in this thesis are based on lattice problems.

Lattices [38] are a mathematical structure used by some cryptographic algorithms, for their particular properties. A lattice in $\mathbb{R}^n$ is a discrete set of points formed by linear combinations of a finite set of basis vectors. Formally, a lattice $\mathscr{L}$ in $\mathbb{R}^n$ can be defined as:

$$\mathscr{L} = \left\{ \sum_{i=1}^{n} z_i \cdot b_i \mid z_i \in \mathbb{Z} \right\} \tag{1}$$

where b_i are the basis vectors of the lattice and z_i are integers. The dimension of a lattice is defined as the number of basis vectors, and the volume of a lattice is defined as the volume of the parallelepiped spanned by its basis vectors.

Some particular bases are considered ideal, a "short" and "orthogonal" basis is one where the basis vectors are as short as possible and as orthogonal to each other as possible. Paradoxically, "bad" bases (long and non-orthogonal) are preferred in cryptography, as they can be used to create hard problems that are difficult to solve. The hardness of these problems is what makes lattice-based cryptography secure.

Hard Lattice Problems

Lattice-based cryptography relies on the hardness of certain mathematical problems related to lattices. The most well-known hard lattice problems include:

- **Shortest Vector Problem (SVP)**: Given a lattice, find the shortest non-zero vector in that lattice.
- **Closest Vector Problem (CVP)**: Given a lattice and a target point, find the closest lattice point to the target.
- **Learning with Errors (LWE)**: Given a linear function with some noise, recover the secret vector used to generate it.
- **Short Integer Solution (SIS)**: Given a matrix A and a target vector t, find a short solution x to the equation $Ax = t$.

2.5.2 Lattice Based Cryptography

Based on the problems described above, lattice-based cryptography aims at creating secure cryptographic primitives, such as encryption schemes, digital signatures, and key exchange protocols. The security of these schemes is based on the assumption that solving the underlying hard lattice problems is computationally infeasible both for classical and quantum computers [23], making them highly attractive candidates for post-quantum cryptography and Namely Ethereum.

2.6 Hash-Based Cryptography

Hash-Based Cryptography in the context of digital signature [35] is the approach that relies on the security of hash functions, provided by their mathematical properties such as preimage resistance, second-preimage resistance and collision resistance (explained in section 2.2), as opposed to relying on number theory (like DSA / ECDSA).

Hash-based signatures are considered to be resistant to quantum attacks [39], and can further mitigate any threats by expanding output sizes from 256 to 512 bits, or even 1024 bits. Making them a suitable candidate for post-quantum cryptography, however this increase in output size might cause problems in terms of performance in the context of blockchain technology and Ethereum.

2.7 Quantum Computing and Cryptography

2.7.1 Quantum Computing

Quantum computing introduces a totally new approach to computing, when so far computers fundamentally relied on the binary system through bits, quantum computers use qubits, having two fundamentally different properties as opposed to bits:

- **Superposition**: A qubit can be in a state of 0, 1 or both at the same time, allowing for parallel processing of information.
- **Entanglement**: Qubits can be entangled, meaning the state of one qubit can depend on the state of another, no matter how far apart they are.

Such properties allow quantum computers to perform tasks in a way classical computers cannot, creating threats to current cryptographic systems.

2.7.2 Main threats to Cryptography

The particularities of qubtis allow quantum computers to solve some hard problems on which current cryptographic systems are based, exponentially faster than classical computers. The most notable examples are:

- **Integer factorization**: The problem of finding the prime factors of a composite number, which is the basis of RSA encryption.
- **Discrete logarithm problem**: The problem of finding the exponent in a modular arithmetic equation, which is the basis of DSA and ECDSA signatures.
- **Hash functions**: The problem of finding a pre-image or collision in a hash function, which is the basis of many cryptographic systems.

Shor's Algorithm

Shor's Algortihm [1] use the power of quantum computing to solve hard problems in polynomial times, $(O((log(N))^3))$, by reducing them to a periodicity problem.

Shor's algortihm for integer factorizations is provided as an example[3]*:*

[3]The QFT is only mentioned as it is involved in the algorithm, further details about this operation are available at : https://en.wikipedia.org/wiki/Quantum_Fourier_transform

Algorithm 1 Example of Shor's Algorithm for Integer Factorization

```
Require: $N$ ▷ The number to be factored
  1. Choose a random integer $a$ such that $gcd(a, N) = 1$
  2. Find the order $r$, i.e., the smallest integer such that $a^r \equiv 1 \mod N$
  ▷ $r$ is the period of the function $f(x) = a^x \mod N$
  This step is done using quantum parallelism
  A quantum computer prepares a superposition of states and evaluates $f(x)$ in parallel.
  The Quantum Fourier Transform to extract the period.
  3. If $r$ is even, compute $x = a^{r/2} \mod N$, $x$ is then a non-trivial factor of $N$.
  4. If $r$ is odd, pick a different $a$ and repeat the process.
```

Instead of the classical time complexity of $\exp((\log N)^{\frac{1}{3}}(\log\log N)^{\frac{2}{3}})$, Shor's algorithm can factor integers in polynomial time, making it a powerful tool for breaking RSA, applying a similar approach to the discrete logarithm problem, Shor's algorithm can also break DSA and ECDSA signatures.

Grover's Algorithm

Other aspects of cryptography are also threatened by quantum computing, such as hashing functions. Grover's algorithm [4] can search an unsorted database of N items in $O(\sqrt{N})$ time, thus providing a quadratic speedup for brute-force attacks namely on hashing functions. This means that a hash function with a security level of n bits would only provide $n/2$ bits of security against quantum attacks. For example, SHA-256, which is currently considered secure, would only provide 128 bits of security against quantum attacks. [4]

And important note is that PoW blockchain are expected to have a better resilience to quantum attacks than PoS blockchains, being reliant on Hash functions, the threat may be less severe than for PoS blockchains. This idea will be further discussed in a later section.

3. PqC Signatures Schemes for Ethereum

3.1 SLH-DSA (SPHINCS+) (Stateless Hash-Based Digital Signature Algorithm)

3.1.1 General Concept

SLH-DSA (SPHINCS+) [20] is a hash-based signature scheme designed to be quantum resistant, under minimal assumptions. It relies on, hash functions, one-time signatures and Merkle trees to provide efficient and secure signatures. The scheme was designed to be stateless, making it suitable for blockchain environnement.

3.1.2 Mathematical Foundations

1. **One time Signature Scheme**
   One time signature schemes (in this context mainly WOTS+[40][41]) is a key component of SLH-DSA (SPHINCS+), it allows for the generation of a unique signature for each message, ensuring that even if the private key is compromised, only one signature can be forged. SLH-DSA (SPHINCS+) uses WOTS+ and Merkle trees to construct a stateless signature scheme. At a high level, the WOTS+ scheme works as follows:
   (a) The signer generates a set of random secret values (private key), and repeatedly hashes (public key).
   (b) To sign a message, the signer encodes the hashed message into a sequence of indices, which are used to select the corresponding secret values from the private key, that are then hashed to produce the signature.
   (c) The signature can then be verified by hashing the signature the remaining number of times and checking if the result matches the public key.
2. **Merkle Trees**[35]
   Merkle trees are a fundamental component of SLH-DSA (SPHINCS+), their structure is similar to that of a binary

[4]The algorithm is not detailed here as it is not directly related to the topic of this report, but it is worth noting that Grover's algorithm does not break hashing functions, but rather reduces their security level.

tree, where each leaf node is a hash of a data block, and each non-leaf node is a hash of its concatenated child nodes. This allows for efficient verification of the integrity of the data stored in the tree.
The SLH-DSA (SPHINCS+) constructs a Merkle tree of WOTS+ public keys, derived from a single private key at its root. The scheme then relies on this structure to provide stateless signing and verification.

3.1.3 Scheme Description

The schemes is designed with different tunable parameters allowing to adjust the security level and performance. The main parameters are:

- n: the number of bits in the hash function output.
- w: (Winternitz parameter) controls the base used in WOTS+
- h: the height of the Merkle tree.

The key generation process in SLH-DSA (SPHINCS+), consists of generating a large set of WOTS+ key pairs from a seed (part of the private key), and organizing them in a Merkle tree using the hashes of the WOTS+ public keys only, leaving the root of the tree as the public key of the whole scheme.

Algorithm 2 SLH-DSA (SPHINCS+) Key Generation

```
Require: $\mathbf{sk_{seed}}$ ▷ The random seed, $\mathbf{n}$ ▷ The number of bits in the hash function output, $\mathbf{w}$ ▷ The Winternitz parameter, $\mathbf{h}$ ▷ The height of the Merkle tree, $\mathbf{H}$ ▷ a hash function.
  1. Using a pseudo-random function with $sk_{seed}$ generate $2^h$ WOTS+ key pairs. (Each pair meant to be used once.)
  2. Compute the Merkle tree leaf nodes by hashing the WOTS+ public keys.
  3. Construct the Merkle tree from the leaf nodes, by hashing the concatenation of each pair of child nodes to form the parent node, until reaching the root ($p_k$).

                    Public Key: $(\mathbf{p_k})$, Secret Key:$(\mathbf{sk_{seed}}, \mathbf{n}, \mathbf{w}, \mathbf{h})$
```

Signing a message with the SLH-DSA (SPHINCS+) scheme is done by hashing the message, determining which WOTS+ key pair to use, signing the message with the corresponding private key, computing the authentication path in the Merkle tree, i.e. the path from the leaf node to the root, including all the "sibling nodes needed to compute the hashes at the higher level, and finally returning the signature as a tuple of the WOTS+ signature and the authentication path.

Algorithm 3 SLH-DSA (SPHINCS+) Signing

```
Require: $\mathbf{m}$ ▷ The message to sign, $(\mathbf{sk_{seed}}, \mathbf{n}, \mathbf{w}, \mathbf{h})$ ▷ The secret key, $\mathbf{H}$ ▷ A hash function
  1. Compute the hash of the message $H(m)$
  2. Using a pseudo-random function with $sk_{seed}$, find the index $i$ of the WOTS+ key pair to use, $i = PRF(sk_{seed}) \bmod 2^h$. This method ensures that the same WOTS+ key pair is used for the same message, allowing for deterministic signing and ensuring the stateless property of the scheme.
  3. Compute the WOTS+ signature $\sigma_{WOTS+} = WOTS + (H(m))$ using the $i_{th}$ WOTS+ key pair.
  4. Compute the authentication path $auth_i$ from the leaf node to the root of the Merkle tree.

                    Signature: $\sigma = (\mathbf{i}, \sigma_{\mathbf{WOTS+}}, \mathbf{auth_i})$
```

Finally, the verification process consists of checking the WOTS+ signature, and then recomputing the Merkle tree root from the leaf node and the authentication path, to compare it the the sender's SLH-DSA (SPHINCS+) public key.

3.1.4 Security and Integrability with Ethereum

The scheme's resistance to post-quantum attacks is based on the security of the underlying hash function and the one-time signature scheme, Shor's algorithm has no impact on it's security and Grover's only provides a quadratic speedup for brute-force attacks, a speedup that is "easily" countered by increasing the key size. The stateless nature of the scheme may also interesting as it alleviates the need for state management in a blockchain context, making it easier to integrate into Ethereum.

Algorithm 4 SLH-DSA (SPHINCS+) Signature Verification

```
Require: m ▷ The message, $\sigma$ ▷ The signature, ($p_k$) ▷ The public key, H ▷ A hash function
  1. Compute the hash of the message $H(m)$
  2. Using the WOTS+ signature $\sigma_{WOTS+}$ and the hashed, recompute the WOTS+ public key $pk_{WOTS+}$.
  3. From $pk_{WOTS+}$ and the authentication path $auth_i$, recompute the Merkle tree root $p'_k$, iteratively hashing the
  concatenation of the leaf node and the sibling nodes in the authentication path.

                    If $p'_k = p_k$ the signature is valid.
```

However, the signature size (~5 KB), pose serious concern in regards to its practical integrability to Ethereum. The scheme also requires a large number of hash function calls, which may be a concern in terms of performance and gas costs. In addition, the scheme's processes are particularly computationally intensive both compared to the existing ECDSA scheme, and to other post-quantum schemes, making it less suitable for Ethereum's current architecture.

3.2 FALCON (FFT over NTRU-Lattice-Based Digital Signature Algorithm)

3.2.1 General concept

FALCON[25] is a signature scheme based on NTRU lattices[42] and using fast fourier transforms (FFT) [43], the former being a lattice based structure that allows for efficient computation of polynomial multiplications, and the latter being a fast algorithm for computing the discrete Fourier transform (DFT) and its inverse. Its resistance stems from the hardness of the structured SIS problem, which is a variant of the Short Integer Solution (SIS) problem, a well-known hard problem in lattice-based cryptography.

3.2.2 Mathematical Foundations

1. **NTRU Lattices** [42]
   NTRU lattices are a family of lattice derived from the NTRU system, defined over polynomial rings of the form:
   $$R_q = \mathbb{Z}_q[x]/(x^n + 1)$$
   with q being a prime number and n being a power of 2. An NTRU lattice is formed by taking the set of all pairs $(u, v) \in \mathbb{R}_q^2$ such that $u \cdot f + v \cdot g = 0 \bmod q$ for some polynomials $f, g \in R_q$, and each element is then a vector of coefficients of polynomials in R_q.

2. **Fast Fourier Transform (FFT)** [43]
   FFT is an efficient algorithm for computing the discrete Fourier transform (DFT) and its inverse. It reduces the computational complexity of DFT from $O(N^2)$ to $O(N \log N)$, fourier transforms are useful in the context of FALCON as they allow for efficient polynomial multiplication by changing the domain of the polynomials from the time domain to the frequency domain, where operations are simpler and faster and then transforming back to the time domain, allowing for efficient key and signature generation.

3. **Structured SIS Problem** [25]
   The structured SIS problem is a variant of the SIS problem, posing it over the R_q ring, the problem is then defined as :
   Given a polynomial $h \in R_q$, and a norm bound β, find a short vector $z \in R_q$ such that $h \cdot z \equiv t \bmod q$ for a given target $t \in R_q$ and $\|z\| \leq \beta$.

3.2.3 Scheme Description

The FALCON scheme provides tunable parameters allowing to adjust the security level and performance.[5] The parameters are as follows:

- n: Degree of the polynomial ring R_q, impacts the size of the keys and signatures, and chosen depending on the desired security level.

[5]It is worth noting that the parameters of the proposed version of FALCON were chosen to resist known attacks against structured lattices, strengthening its security assurances.[25]

- q: A prime number, which defines the modulus for the polynomial operations, impacts ring structure.
- σ: The standard deviation of the Gaussian distribution used for sampling.
- β: The norm[6] bound for the structured SIS problem, also influences key/signature sizes.

Algorithm 5 FALCON Key Generation

```
Require: n, l, σ
  1. Sample two short polynomials $f, g \in \mathbb{Z}[x]/(x^n+1)$
  (their coefficients are sampled from a Gaussian distribution with standard deviation $\sigma$ over $\mathbb{Z}$)
  2. Ensure $f$ is invertible and compute $f^{-1} \bmod q$.
  3. Compute $h = \frac{g}{f} \bmod q \in R_q$
                    Public Key: (h), Secret Key: (f, g)
```

Algorithm 6 FALCON Signing

```
Require: m ▷ The message to sign, (f, g) ▷ The secret key, (h) ▷ The public key, H ▷ A hash function, σ ▷ The standard
  deviation of the Gaussian distribution used for sampling, β ▷ The norm bound for the structured SIS problem
  1. Sample a short vector of polynomials $y \in \mathbb{Z}[x]/(x^n+1)$
  2. Compute $v = h \cdot y \bmod q$
  3. Compute the challenge $c = H(v||m)$
  4. Compute $z = y + f \cdot c$
  If $||z|| \leq \beta$ and $||v - g \cdot c|| \leq \beta$ then the signature is accepted, otherwise $y$ is resampled.
                    Signature: (z, c)
```

Verifying an FALCON signature consists of ensuring that z is a "valid" short vector, and then recomputing the challenge c to compare it with the one provided in the signature.

Algorithm 7 FALCON Signature Verification

```
Require: m ▷ The message, (z, c) ▷ The signature, β ▷ The norm bound for the structured SIS problem
  If $||z|| > \beta$ The signature is invalid
  1. Compute $v' = h \cdot z \bmod q$ (An approximation of the $v$ the signer computed)
  2. Compute the challenge $c' = H(v'||m)$
                    If $c' = c$ the signature is valid.
```

3.2.4 Security and Integrability with Ethereum

The security of FALCON based on the hardness of the structured SIS problem, and the small signature size, make this scheme an interesting candidate for Ethereum, namely such compact signatures would save space on the blockchain. However this scheme is vulnerable to side-channel attacks [25], and its implementation would require some work to prevent such attacks, making it less straightforward to integrate than other schemes.

Additionally, EVMs (Ethereum Virtual Machines) are not designed to handle the polynomial operations required by FALCON, or other Lattice based schemes[7], and would require significant changes to the EVM architecture to support such operations, which would one of the main challenges of integrating FALCON or other Lattice based schemes into Ethereum.

[6](The norm is usually either $||\cdot||_\infty$ or $||\cdot||_2$)

[7]As explained here : `https://ethereum.org/en/developers/docs/evm/`, EVMs are designed for 256-bit integer arithmetics

3.3 ML-DSA (Module-Lattice-Based Digital Signature Algorithm)

3.3.1 General Concept

ML-DSA[24] is a lattice-based digital signature scheme, a member of the CRYSTALS-DILITHIUM family, which was selected as a finalist in the NIST post-quantum cryptography standardization process. It is designed to be secure against quantum attacks and is based on the hardness of the Module Learning with Errors (MLWE) and Module Short integer solution (MSIS) problems, both of which are believed to be hard even for quantum computers.

3.3.2 Mathematical Foundations

1. **Module Lattices** [38]
   A module lattice is a generalization of lattices over integer to modules over rings, increasing representation and computation efficiency.
   Let a ring
   $$R_q = \mathbb{Z}_q[x]/(x^n + 1)$$
   (a ring of degree-n (usually, $n = 256$) polynomials with coefficients modulo q) Where q is a prime number and n is a power of 2. The module lattice is defined as R^k where k is the dimension of the lattice. The elements of the module lattice are polynomials with coefficients in $\mathbb{Z}_q$, and the operations are performed modulo q.[8]

2. **Hardness Assumptions** [38]
   ML-DSA relies on the hardnes of Lattice problems generalized to module lattices. The two main problems are:

   - **Module Learning with Errors (MLWE)**
     Given $A \in R_k^{l\times k}$ and $t = A\cdot s + e$ it is computationally hard to differentiate t from random when s and e are small-norm secrets and errors respectively.

   - **Module Short Integer Solution (MSIS)**
     Given a random matrix $A \in R_q^{l\times k}$ (a matrix of $k \times l$ elements, each a polynomial in R_q) it is hard to find a small z such that $A\cdot z = 0 \bmod q$

3.3.3 Scheme Description

The Scheme involves some paramters that are chosen based on the desired security level.[44] The parameters are:

- k: *impacts public key length*
- l: *impacts signature length*
- q: *a prime number, chosen based on mathematical constraints*

Algorithm 8 ML-DSA Key Generation

```
Require: k, l, q ▷ Security-dependent parameters, chosen based on the desired security level
  1. Sample $A \in R_q^{l\times k}$ (pseudo-random sampling using a seed, to only store the seed rather than the martix)
  2. Sample two short vectors of polynomials $s_1 \in R_q^l$ and $s_2 \in R_q^k$
  3. Compute $t = A\cdot s_1 + s_2 \in R_q^k$
                    Public Key: $(\mathbf{A}, \mathbf{t})$, Secret Key: $(\mathbf{s_1}, \mathbf{s_2})$
```

The signature consists of three parts: μ, z, and c. The message is hashed with the public key and the high part of w to produce a challenge c. The secret key is used to compute z by adding the challenge multiplied by the secret key to the short vector y. The verifier will then use the public key z and μ to compute a response c' and compare it with the original challenge c. If they match, the signature is valid.

[8]NTRU and modular lattices rely the same underlying structure, i.e. the polynomial ring R_q. The difference lying in the way the lattices are constructed over the ring.

Algorithm 9 ML-DSA Signing

```
Require: $\mathbf{m}$ ▷ The message to sign, $(\mathbf{s_1}, \mathbf{s_2})$ ▷ The secret key, $(\mathbf{A}, \mathbf{t})$ ▷ The public key, $\mathbf{H}$ ▷ A hash function, usually SHAKE-256
    1. Sample a short vector of polynomials $y \in R_q^l$
    2. Compute $w = A \cdot y$ and split $w$ into $w_1$ (high part) and $w_2$ (low part) such that $w = w_1 + w_2$
    2.5 Compute $\mu = H(pk||m)$ with $pk$ being the compressed public key.
    3. Hash the message with $w_1 : c = H(w_1, \mu, m)$
    4. Compute $z = y + c \cdot s_1$
    If $z$ or $c \cdot s_2$ are not short enough (i.e. their norm is too large), the signature is rejected and $y$ is resampled.
                                   Signature: $(\mu, \mathbf{z}, \mathbf{c})$
```

Algorithm 10 ML-DSA Signature Verification

```
Require: $(\mu, \mathbf{z}, \mathbf{c})$ ▷ The signature to verify, $(\mathbf{A}, \mathbf{t})$ ▷ The public key, $\mathbf{H}$ ▷ A hash function, usually SHAKE-256, naturally the same as the one used in signing
    1. Compute $w' = A \cdot z - c \cdot t$ (An approximation of $w = A \cdot y$)
    2. Compute $w'_1$ and $w'_2$ from $w'$
    3. Hash $c'$ with $w'_1 : c' = H(w'_1, \mu, m)$
                     If $\mathbf{c} = \mathbf{c}'$ and norms are within bounds Signature is verified.
```

The verifier, needing w to recompute c' a compare it with the received c, but not having access to y, uses the public key (A, t) to approximate w by computing $w' = A \cdot z - c \cdot t$. The signature is valid if the recomputed c' matches the original c and the norms of z and $c \cdot s_2$ are within bounds.

3.3.4 Security and integrability with Ethereum

This scheme offers a strong post-quantum security, relying on the hardness of the MLWE and MSIS problems[24] [23]. The security level can be adjusted by changing the parameters k, l, and q, also affecting the size of the public key and signature[44]. The scheme is designed to be efficient, with fast signing and verification times, making it suitable for integration into Ethereum.Its practical integrability into the Ethereum 2.0 blockchain, is promising despite its relatively large key and signature sizes, for instance, in Dilithium II, public keys are ~ 1.3 KB and signatures ~2.7 KB, which significantly exceeds the current ECDSA footprint (~64 bytes each), however ML-DSA remains a viable candidate for secure transaction validation in a post-quantum Ethereum environnement.

The possible challenges of integrating ML-DSA in Ethereum, are similar to those of FALCON, as both schemes are based on lattice structures, and the main challenges being side-channel attacks, as well as the need to adapt EVMs to support Lattice-based operations, which may require significant changes to the Ethereum protocol and smart contract execution environment.

4. Implementation on the Ethereum Blockchain

4.1 Selected Post-Quantum Signature Scheme

The selection of a signature scheme for the Ethereum blockchain will rely on different criteria ensuring the selecting scheme will address all the requirements of the blockchain context, the following criteria will be used:

- **Security:** The main concern of any cryptographic scheme is its security. All three candidates relying on hard mathematical problems and being quantum resistant, to the point of having become NIST standards [12], this criteria is met, leading to the choice being based on the other criteria.
- **Key / Signature Size:** SLH-DSA (SPHINCS+) provides large signature and key sizes, negatively impacting its scalability and performance in the context of Ethereum thus making the scheme seem unfit for the blockchain context, the other two candidates have a smaller key and signature size, while preserving satisfactory security levels.
- **Signing and Verification performance:** ML-DSA is expected to be faster than FALCON, especially in the verification process, a particularly important aspect as signatures verifications are a very frequent operation in the Ethereum blockchain. ML-DSA however provides larger signatures than FALCON, which may impact the performance of the scheme in the context of Ethereum, but is expected to be compensated by the faster verification process.
- **Integrability:** SLH-DSA (SPHINCS+) introduces the challenge of managing the large signature and key sizes, leading to significant intergration challenges with the Ethereum blockchain. FALCON being a Lattice-based scheme, it will require an adaptation of EVMs to accommodate FFT-based operations, inducing a possibly heavy overhead in deployment. ML-DSA although needing some adaptation of EVMs, does not seem to require as much as FALCON, and is expected to be easier to integrate into the Ethereum blockchain.

Although all candidates have been standardized by NIST, and therefore being relevant in the consideration, ML-DSA stands out as the most suitable scheme for the Ethereum blockchain based on its theoretical and expected performance, inducing an expected lower impact on gas fees and offering a better integrability than the other candidates.

4.2 Implementation Strategy

Integrating the new signature scheme into the existing Ethereum environnement requires taking into account several critical issues, namely, compatibility with the current protocol, performance, security, user adaptation and trust. In order to ensure a smooth transition, a phased approach seems to be the most appropriate :

- Phase 0: Research and Scheme selection
- **Phase 1: Dual Signature Verification Support:** This first phase will focus on introducing the verification process of the the new signature scheme, preparing the blockchain to accept new signatures while maintaining backwards compatibility. Users should not feel the impact of this phase, neither should the gas fees or overall performance of the blockchain. This phase is also the point at which EVM should be adapted to support ML-DSA's lattice-based operations.
- **Phase 2: Hybrid Signing:** At this stage, the new signature scheme will be entirely integrated into the blockchain and users will have the option to chose between the legacy scheme or the new one for signing their transactions. This phase will allow users to gradually adapt to the new scheme, while still being able to use the legacy scheme and will give useful insights on the new scheme's performance and security in a real-world context.
- **Phase 3: Network-Wide Migration:** During this phase, ML-DSA will become the default signature scheme for the Blockchain, all critical operations will be performed using the new scheme, while the legacy scheme will still be supported for a period of time, progressively eliminating ECDSA from the network's operations.
- **Phase 4: ECDSA Deprecation:** The final phase will consist of the complete removal of ECDSA from the transaction validation process. This process might require a hard fork to ensure all nodes are updated and the legacy scheme is no longer used. Smart contract and wallets will be required to migrate to the new scheme to keep their functionalities. This phase is expected to be the most impactful as after this irreversible step, all users will be required to use the new scheme, and the legacy scheme will no longer be supported. Previous transaction and smart contracts using ECDSA will still be available on the blockchain in a read-only mode.

4.3 Implementation Details

The proof of concept implementation was done in python using packages offering the necessary cryptographic primitives, the `ecdsa` [9] package for the and the `pqcrypto` [10] package for the post-quantum cryptography primitives. The implementation is available on GitLab[11]

For comparison, two blockchain models were implemented: the main one being the simplified Etherreum blockchain model that will represent the current state of the Ethereum blockchain and its expected states throughout and after the transition,and a second Proof-of-work based one, that will help gain insight on how the performance of a proof-of-work blockchain adapted to maintain its security against quantum threats would compare both to the post-quantum proof-of-stake blockchain and to a classical non-post-quantum proof-of-work blockchain.

4.3.1 Block Structure and Chain Logic

In this model, the block structure is defined as a simplified version of the Ethereum block structure, which includes the following fields:

- `index`: The block number in the chain.
- `previous_hash`: The hash of the previous block.
- `transactions`: A list of transactions objects, containing the sender, recipient, amount, and signature.
- `timestamp`: The time when the block was created.
- `nonce`: A number used only for the proof of work blockchain.

```python
class Block:
    def __init__(self, index, previous_hash, transactions, validator, signature,
    ↪   signature_scheme, timestamp=None, hash_function=hashlib.sha256):
        self.index = index
        self.previous_hash = previous_hash
        self.transactions = transactions
        self.validator = validator
        self.signature = signature
        self.signature_scheme = signature_scheme
        self.timestamp = timestamp or time.time()
        self.hash = self.calculate_hash()

    def calculate_hash(self):
        tx_str = ''.join([tx.hash() for tx in self.transactions])
        block_str = f'''{self.index}{self.previous_hash}
        {self.timestamp}{tx_str}{self.validator}
        {self.signature_scheme}'''
        return self.hash_function(block_str.encode()).hexdigest()

```

Listing 1: Block class definition.

The chain logic is implemented to handle the addition of new blocks, verification of transactions, and block validation. And is designed to allow for performance measurements manageable in environments with computational limited resources, while still being representative of the Ethereum blockchain. It also includes a simple way to manage the proportion of blocks signed with the ECDSA and ML-DSA signature schemes, allowing for easy simulation of the transition phases of the Ethereum blockchain.

[9]Link to the ecdsa package: https://pypi.org/project/ecdsa/
[10]Link to the ecdsa package: https://pypi.org/project/pqcrypto/
[11]Link to the GitLab repository: https://gitlab.unige.ch/Dany.Al-Moghrabi/exploring-ethereum-post-quantum-signatures/-/tree/main/src?ref_type=heads

4.3.2 Transaction Model

The transaction class was designed to represent a simple transaction in the blockchain, containing the sender, recipient, amount, and signature, additionally the possibility to hash the transaction was added to allow for easy verification of the transaction's integrity. The transaction class also includes methods to serialize and hash the transaction, which are used for block validation and chain logic. The transaction class is designed to be simple and efficient, allowing for easy integration into the blockchain model.

```python
class Transaction:
    def __init__(self, sender, recipient, amount, signature=None,
     ↪  hash_function=hashlib.sha256):
        self.sender = sender
        self.recipient = recipient
        self.amount = amount
        self.signature = signature
        self.hash_function = hash_function

    def serialize(self):
        return f'{self.sender}->{self.recipient}:{self.amount}'

    def hash(self):
        return self.hash_function(self.serialize().encode()).hexdigest()
```

Listing 2: Transaction class definition.

4.3.3 Signature schemes and Key Sizes

In order for the schemes to be easily swappable, an abstract class `SignatureScheme` was created, which defines the methods that all signature schemes must implement. Three signature schemes are implemented using this interface and packages that provide the necessary cryptographic primitives. The schemes are as follows:

- **ECDSA (SECP256k1)**: The current signature scheme used in Ethereum, based on the SECP256k1 curve.
  - Key size: Private key: 32 bytes; Public key: 65 bytes.
  - Signature size: From 64 to 72 bytes.
- **ML-DSA (ML-DSA-65)**: The ML-DSA signature scheme with equivalent security to the current ECDSA scheme.
  - Key size: Private key: 4032 bytes; Public key: 1952 bytes.
  - Signature size: 3293 bytes.
  - *This scheme is also used to sign the transactions in the post-quantum proof-of-stake blockchain.*

The key and signature sizes are taken from the respective packages used for the implementation, and are subject to small variations, but aim at being representative of the actual sizes used in the Ethereum blockchain or in the NIST standards.

In order to facilitate the integration of the schemes, an abstract class `SignatureScheme` was created, which defines the methods that all signature schemes must implement. This allows for easy swapping of the signature schemes in the blockchain model.

```python
# signature_scheme.py
from abc import ABC, abstractmethod

class SignatureScheme(ABC):
    @abstractmethod
    def generate_keys(self):
        pass

    @abstractmethod
    def sign(self, private_key, message):
        pass

    @abstractmethod
    def verify(self, public_key, message, signature):
        pass
```

Listing 3: SignatureScheme abstract class definition.

4.3.4 Validator Model and Dual Singing

This model uses a simplified version of the Ethereum proof-of-stake model, and focusing on the signature, block production and validation processes, it does not implement the staking and slashing mechanisms, instead validators are fully randomly selected from a basic pool of validators rather rather than based on stake-weighted probability.

For the simulated transition phases of the Ethereum blockchain, in order to support the dual signing mechanisms Each validator is assigned keypairs for both the ECDSA and ML-DSA signature schemes. Block validation is done via digital signatures produced by the selected validator, which mimics the Ethereum mechanism where validators attest to blocks through signed messages.

```python
class Validator:
    def __init__(self, address, ecdsa_scheme, mldsa_scheme):
        self.address = address
        self.ecdsa_scheme = ecdsa_scheme
        self.mldsa_scheme = mldsa_scheme
        self.ecdsa_sk, self.ecdsa_pk = ecdsa_scheme.generate_keys()
        self.mldsa_sk, self.mldsa_pk = mldsa_scheme.generate_keys()

    def sign(self, message, scheme_name):
        if scheme_name == "ecdsa":
            return self.ecdsa_scheme.sign(self.ecdsa_sk, message)
        elif scheme_name == "mldsa":
            return self.mldsa_scheme.sign(self.mldsa_sk, message)
        else:
            raise ValueError(f"Unknown scheme: {scheme_name}")

    def get_scheme(self, scheme_name):
        if scheme_name == "ecdsa":
            return self.ecdsa_scheme
        elif scheme_name == "mldsa":
            return self.mldsa_scheme
        else:
            raise ValueError(f"Unknown scheme: {scheme_name}")
```

Listing 4: Validator class definition.

4.3.5 Proof of Work Blockchain

In order to gain insight on how a proof-of-work blockchain adapted to maintain its security against quantum threats would compare to the post-quantum proof-of-stake blockchain, another blockchain model was implemented. A first model was implemented to represent a standard proof-of-work blockchain, similar to Bitcoin's, using the ECDSA signature scheme for internal transactions and SHA-256. A second model was implemented to represent a post-quantum proof-of-work blockchain, using the ML-DSA signature scheme for internal transactions and SHA-512 as the hash function, this model uses a proof-of-work mechanism with hashes double the length of the standard and transactions signed (in proof of work block chains such as Bitcoin transactions are still signed although blocks are not [45]) simulating the management of both the speedup offered by Grover's algorithm[4] and Shor's[1] algorithm breaking the usually used signature scheme. The block structure and chain logic are similar to the post-quantum proof-of-stake blockchain, but with the addition of a nonce field for proof-of-work.

The block structure for the proof-of-work blockchain is as follows:

- `index`: The block number in the chain.
- `previous_hash`: The hash of the previous block.
- `transactions`: A list of transactions objects, containing the sender, recipient, amount, and signature.
- `timestamp`: The time when the block was created.
- `difficulty`: The difficulty of the proof-of-work.
- `nonce`: A number used for proof-of-work.

```python
class PoWBlock:
    def __init__(self, index, previous_hash, transactions, difficulty, timestamp=None,
    ↪ hash_function=hashlib.sha512):
        self.index = index
        self.previous_hash = previous_hash
        self.transactions = transactions
        self.difficulty = difficulty
        self.timestamp = timestamp or time.time()
        self.nonce = 0
        self.hash = None
        self.hash_function = hash_function

class PoWBlockchain:
    def __init__(self, difficulty=4, block_hash_function=hashlib.sha512, tx_scheme =
    ↪ MLDSASignatureScheme()):
        self.chain = []
        self.difficulty = difficulty
        self.tx_scheme = tx_scheme # Transaction scheme for signing transactions
        self.tx_priv_key, self.tx_pub_key = self.tx_scheme.generate_keys() # Transaction
        ↪ keys
        self.block_hash_function = block_hash_function
        self.create_genesis_block()
```

Listing 5: Proof-of-Work Block and Blockchain class definition.

Both the block and blockchain class are designed to work similarly to the proof-of-stake blockchain, naturally with the exception being their consensus mechanism, allowing for easy integration and comparison between the two models.

4.3.6 Phase Configuration

In order to simulate the transition phases of the Blockchain, 4 blockchains were set up, each representing a different phase of the transition by implementing a different proportion of blocks signed with the ECDSA and ML-DSA signature schemes. The phases are as follows:

Phase	Description	% of ECDSA Blocks	% of ML-DSA Blocks
Phase 1	Initial phase with only ECDSA blocks	100%	0%
Phase 2	Hybrid Phase with both ECDSA and ML-DSA blocks with a majority of ECDSA blocks	75%	25%
Phase 3	Hybrid Phase with both ECDSA and ML-DSA blocks with a majority of ML-DSA blocks	25%	75%
Phase 4	Final phase with only ML-DSA blocks	0%	100%

Figure 10: Simulated phases of the transition from ECDSA to ML-DSA

This configuration aims to represent the gradual transition from the current ECDSA-based Ethereum blockchain to a post-quantum ML-DSA-based blockchain, allowing for performance tracking through the process as well as a comparison between the current state ot the Ethereum blockchain (Phase 1) and the expected post-quantum one (Phase 4).

4.3.7 Simulation and Performance Tracking

The simulation is designed to run for 1000 blocks, with the ratio of usage of each scheme determined as explained earlier. The simulation tracks the following performance metrics:

- **Block Creation Time**: The time taken to create each block.(This measurement is also realized for the proof-of-work blockchain.)
- **Average Transaction Verification Time**: The time taken to verify each transaction in the block.
- **Average Signature Verification Time**: The time taken to verify the signature of each block.

The latter two metrics are weighted by the ratio of blocks signed with each scheme, allowing for a fair comparison between the two signature schemes.

```python
# Set up phase parameters
NUM_BLOCKS = 1000
ECDSA_RATIO_CASES = {
    "Phase 1 (100% ECDSA - 0% ML-DSA )": 1.0,
    "Phase 2 (75% ECDSA - 25% ML-DSA)": 0.75,
    "Phase 3 (25% ECDSA - 75% ML-DSA)": 0.25,
    "Phase 4 (0% ECDSA - 100% ML-DSA)": 0.0
}

for label, ratio in ECDSA_RATIO_CASES.items():
    chain = Blockchain(validators, ecdsa_ratio=ratio)
    times = []

    for i in tqdm(range(NUM_BLOCKS), desc=f"Running {label} block time measurements",
    ↪  bar_format='{desc}: {bar:30} {percentage:3.0f}%'):
        txs = [Transaction("A", "B", i)]
        start = time.perf_counter()
        chain.add_block(txs)
        times.append(time.perf_counter() - start)
```

Listing 6: Block time measurements for the proof-of-stake blockchain phases.(Measurements for the proof-of-work blockchain are done in a similar way.)

The singing and verification measurements are done in a similar way, at each phase, 1000 signatures are created respecting each phase's ratio of usage of each scheme, and the time taken to sign and verify each signature is measured. The average time is then calculated for each phase, allowing for a comparison between the two signature schemes.

5. Feasibility and Performance Analysis

5.1 Feasibility Analysis

This section explores these aspects in greater detail to evaluate the feasibility of adopting ML-DSA in a live Ethereum environment.The simulation confirms that, conceptually, transitioning to post-quantum signatures such as ML-DSA is achievable without fundamentally altering Ethereum's blockchain structure or consensus mechanism.

EVM Compatibility and Smart Contract Integration

A significant technical consideration is compatibility with the Ethereum Virtual Machine (EVM). The current EVM is optimized for elliptic curve arithmetic, particularly for ECDSA-based signatures. By contrast, ML-DSA relies on polynomial operations and modular arithmetic over lattice structures. While not fundamentally incompatible, this would necessitate the addition of new precompiled contracts (as seen in EIPs[12] like EIP-197[46] and EIP-2537[47] for cryptographic curves) and potentially even new opcodes or gas metering adjustments. Alternatively, integration through eWASM[48], (a proposed upgrade to the Ethereum Virtual Machine (EVM)) may allow for more flexible cryptographic primitives in future versions of Ethereum.

Transaction Format and Network Propagation

ML-DSA signatures are significantly larger than those produced by ECDSA. Larger signatures directly translate to increased bandwidth usage, slower transaction propagation, and higher gas fees. Moreover, block size and gas usage policies would need revision to prevent issues caused by inflated payloads. These risks could be mitigated with layered solutions such as compression or post-quantum aggregation, but those remain active areas of research.

Backward Compatibility and Governance Overhead

Even though a progressive rollout strategy was simulated (via Phases 1 to 4), a full transition to post-quantum signatures would eventually require a hard fork to enforce the use of ML-DSA at the consensus level. This would entail governance coordination similar to past major forks like Byzantium[49] or London[50]. During the transition period, dual-signature support must be managed securely, with consensus clients verifying both types of signatures. This requires canonical rules to avoid consensus failure between clients. Additionally, tooling and protocol-level auditability must be extended to accommodate post-quantum key formats and their serialization.

Key Management and Validator Operations

The simulation assumes validators possess both ECDSA and ML-DSA key pairs. In practice, managing dual key infrastructures at scale poses operational and security risks. Existing tools (e.g., MetaMask, Ledger, or staking services) are deeply integrated with elliptic curve formats. Introducing lattice-based formats would require new standards for seed generation, serialization, and HD key derivation. Furthermore, on-chain staking and slashing conditions would need to recognize and enforce rules for both key types, increasing protocol complexity and the likelihood of misconfiguration during the transition.These operational challenges could increase the risk of validator misbehavior or misconfiguration, especially during the transition period, and would require extensive tooling updates and stakeholder education.

Although not without integration challenges, ML-DSA offers a pragmatic and forward-compatible path toward post-quantum security for Ethereum—provided that the ecosystem evolves to accommodate larger keys, new cryptographic primitives, and updated operational standards.

[12]Ethereum Improvement Proposal, design documents providing information to the Ethereum community about new features, standards, or changes to the protocol, see `https://ethereum.org/en/eips/#what-are-eips`

5.2 Performance Analysis

This subsection aims at presenting the results of the different performance metrics collected during the simulations, comparing the different processes across the 4 different transition phases of the blockchain and an PoW based blockchains, one representing a system close to bitcoin's and one adapted to offer post-quantum resistance.

5.2.1 Block Time

Block creation time refers to the total time required to construct, sign (if applicable), and append a block to the blockchain. This metric includes transaction processing, block header generation, signature computation, and chain linkage.

In the simulation, 1000 blocks were generated for each phase under a fixed configuration, and the average block creation times were measured.

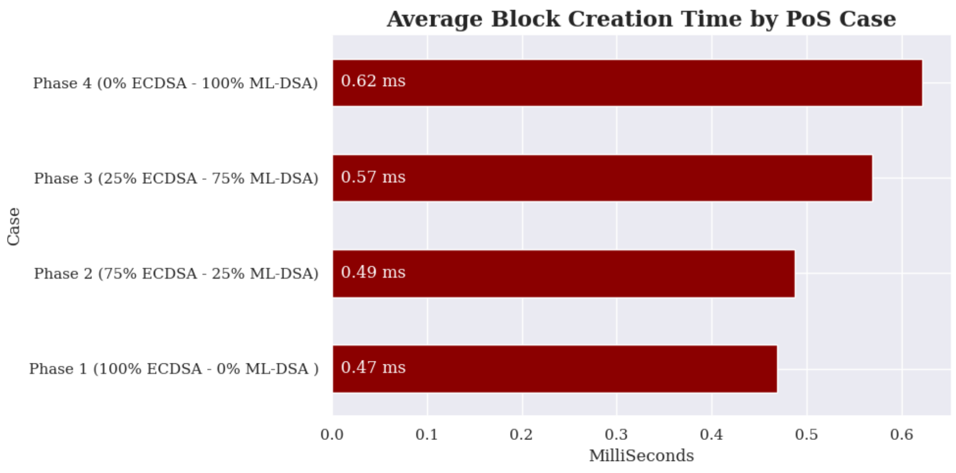

Figure 11: Average block creation time in the PoS Simulation.

As expected, block creation time increases gradually as more ML-DSA blocks are introduced. This reflects the relatively higher computational cost of ML-DSA signing operations compared to ECDSA. The difference between Phases 1 and 4 remaining relatively small ($0.11ms$). This increase, although measurable, represents only a ~ 24% rise over the baseline and remains within Ethereum's tolerance for transaction finality, which typically operates on block times of 12 seconds. Thus, it is unlikely to create any meaningful latency at scale.

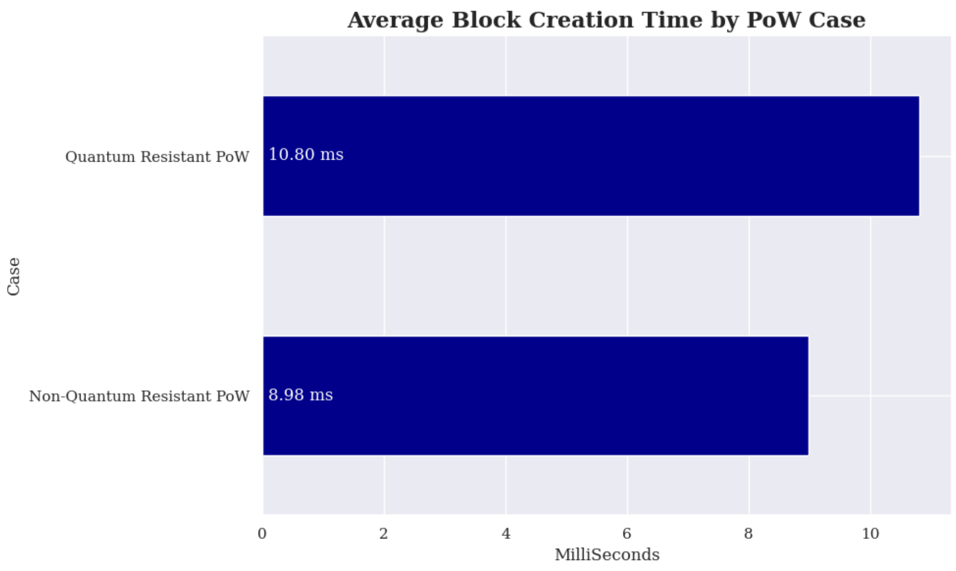

Figure 12: Average block creation time in the PoW Simulation.

The Proof-of-Work configurations show an average block time an order of magnitude higher than the Proof-of-Stake phases due to the inherent computational cost of mining, with the quantum resistance configuration being slightly slower than the standard PoW configuration. This is expected, as the hash function used (SHA-512) in introduces additional computational overhead compared to the traditional SHA-256 used in Bitcoin. The PoW results serve primarily as a baseline to show that, even with quantum-safe hashing, PoW remains significantly more resource-intensive than a signature-heavy PoS model. This confirms that the performance advantage of PoS holds even in the context of quantum-resistant signatures.

These results confirm that, while ML-DSA introduces measurable overhead in block creation, it seems to remain within acceptable limits for real-time block production.

5.2.2 Signing and Verification Time

This metrics respectively measure the time required to produce a signature and verify it, independently of other blockchain logic. It reflects the cryptographic cost associated with each signature scheme.

In the experiment, 1000 signatures were generated and verified per phase. In mixed phases, a weighted average was computed based on the proportion of ECDSA and ML-DSA used. The results are shown hereafter:

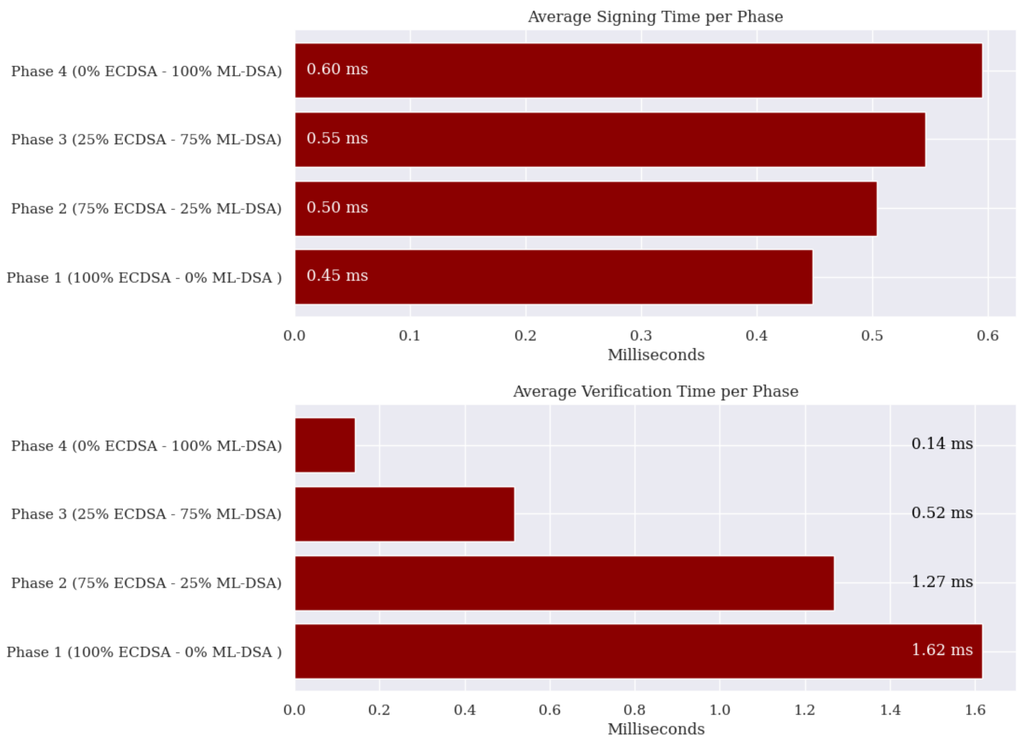

Figure 13: Average block signing and verification Time comparison across all ETH based blockchains.

The figure reveals two key trends:

- Signature time increases slightly with higher ML-DSA adoption. Phase 1 (100% ECDSA) averaged ~ 0.45 ms, while Phase 4 (100% ML-DSA) reached ~ 0.60 ms. This overhead remains modest and does not represent a major bottleneck in block production.

- Verification time decreases significantly as the portion of ML-DSA block increases. Verification in Phase 1 averaged ~ 1.60 ms, while ML-DSA in Phase 4 required only ~ 0.14 ms, this speedup is expected as ML-DSA signatures are known to be much faster to verify than ECDSA signatures, due to their structure and the underlying mathematical operations involved. In addition since verification occurs every time a transaction is processed or re-validated, while signing is performed only once by the sender, ML-DSA's speedup on the verification side is particularly advantageous for a high-throughput blockchain like Ethereum.

These results show that ML-DSA introduces a manageable overhead in signing while offering substantial gains in verification speed. This trade-off may not only support the feasibility of ML-DSA in Ethereum, but also suggest a performance benefit in scenarios where verification is the more frequent operation.

5.2.3 Gas Cost Estimation

An important factor in considering the integration of ML-DSA into Ethereum is the gas cost[13] associated with transmitting and verifying those signatures. In Ethereum, calldata incurs a cost of 16 gas per non-zero byte and 4 gas per zero byte. Since post-quantum signatures such as ML-DSA are significantly larger than ECDSA signatures, they result in substantially higher gas consumption during transaction propagation and processing.

Phase	Gas Cost (in gas units)	Average Signature Size (bytes)
Phase 1 (ECDSA)	1024.00	64.00
Phase 2 (Hybrid 75% ECDSA - 25% ML-DSA)	13917.00	875.25
Phase 3 (Hybrid 25% ECDSA - 75% ML-DSA)	39703.00	2497.75
Phase 4 (ML-DSA)	52596.00	3309.00

Figure 14: Gas cost and average signature size for each simulation phase.

As expected, gas costs scale nearly linearly with the size of the signature.In the final state, with average signatures of (~ 3309 bytes) post-quantum signatures would occupy a disproportionate share of a block's 15M gas limit. For example, a single ML-DSA transaction could consume over 50,000 gas solely for calldata. These results highlight an important challenge of adopting post-quantum signatures on Ethereum: the significant increase in transaction size, and therefore in gas cost. This overhead may be mitigated in practice through future protocol optimizations, or introducing post-quantum-aware gas pricing models.

Despite the increased signature size and moderate signing overhead, the simulation demonstrates that ML-DSA offers interesting performance characteristics for Ethereum. The reduced verification time and acceptable increase in block production latency support its potential as a scalable post-quantum signature candidate. However, addressing calldata gas inefficiency and optimizing transaction formats will be essential to ensure practical adoption.

6. Privacy and Security Enhancements

6.1 Privacy Enhancements through post-quantum signatures

Post-quantum signature schemes, namely the ones based on lattice structures, offer new opportunities for privacy improvements in decentralized networks like Ethereum. While digital signatures are not inherently designed for privacy, some post-quantum primitives possess structural features that enable privacy-preserving protocols.

A key privacy concern, in Blockchains and namely in Ethereum is linkability[51][52], the ability of observers to link multiple transactions to the same user based on recurring digital signature patterns. This can be exploited by adversaries performing transaction graph analysis, enabling the de-anonymization of users even in pseudonymous settings. In ECDSA, the signature structure reuse can facilitate these analysis.

Lattice-based schemes like ML-DSA offer a mitigation this threat by incorporating intrinsic randomness in the signing process. Each signature, even when generated from the same key and message, is statistically independent due to the noise sampling from high-entropy distributions. This property significantly increases unlinkability, reducing the risk of identifying recurring actors in the network.

Additionally, the large key and randomness spaces inherent to lattice-based schemes make pre-image and correlation attacks computationally infeasible even for powerful adversaries. When deployed within Ethereum's account-based model, ML-DSA could serve as the basis for enhanced privacy-preserving constructs, supporting untraceable transaction patterns without requiring fundamental changes to the blockchain architecture.

6.2 Use of zero-knowledge proofs alongside post-quantum signatures

Zero knowledge proofs[53], a cryptographic protocols that allow one party to prove to another that a statement is true without revealing any additional information beyond the validity of the statement itself; are a powerful tool already used in Ethereum to verify smart contract and transaction correctness through zk-SNARKs[54], zk-STARKS[55], and zk-Rollups[56], and considering post-quantum cryptography will be an important aspect to keep these constructs secure against quantum threats. While independent from the signature layer, the inclusion of zero-knowledge proofs in Ethereum's protocol stack is crucial for enhancing privacy, scalability, and security.

[13]See Ethereum gas documentation: `https://ethereum.org/en/developers/docs/gas/#what-is-gas`

Some currently used zero-knowledge proof systems like zk-SNARKs[54] rely on elliptic curves, making them vulnerable to quantum attacks. A challenge posed by quantum resistant ZKPs, such as zk-STARK and lattice-base zk-SNARKs[57] is that they tend to have large proof sizes and induce a heavy computational overhead. They however are being studied and developed for practical use:

- zk-STARKs: quantum-secure (since they only rely on hash functions), transparent (no trusted setup), scalable and support large computations but come with large proof sizes and significant gas and bandwidth costs in blockchain environments like Ethereum.
- Lattice-based zk-SNARKs[58]: designed to offer compact, post-quantum-secure, and universal zero-knowledge proofs. Unlike STARKs, lattice-based SNARKs can maintain relatively shorter proofs and better integration with Ethereum's infrastructure, especially if Ethereum adopts ML-DSA, which is also based on lattice assumptions.

By selecting ML-DSA as the primary signature scheme, Ethereum would already be operating in a lattice-based cryptographic context, easing the transition to lattice-based zk-SNARKs, opening the door to developing cryptographically unified proof systems, reducing the complexity of maintaining hybrid cryptographic stacks and potentially optimizing performance through shared primitives (e.g., ring/module sampling, Gaussian noise, or polynomial arithmetic).

However, lattice-based ZKP systems are still in an early phase. Active research continues to improve their efficiency, trusted setup requirements, and compatibility with practical constraint systems. Recent proposals such as qSNARK[59], LigeroLWE[60], and Module-LWE-based Groth-like constructions [61] show promise in achieving practical zero-knowledge proofs over lattices, but they have yet to reach the same maturity and performance optimization as classical alternatives.

These developments suggest that future-proofing Ethereum's privacy and scalability stack will require cohesive advances in both post-quantum signature schemes and zero-knowledge proof constructions.

7. Conclusion

As quantum computing continues to evolve from theoretical to practical threat, ensuring the long-term security of blockchain systems has become a critical concern. In this context, the Ethereum blockchain,central to a wide array of decentralized applications, must adapt to maintain its robustness in a post-quantum world.

This thesis explored the feasibility, efficiency, and integration challenges of post-quantum digital signature schemes within Ethereum's hybrid Proof-of-Stake framework. After a thorough evaluation of the leading candidates standardized by NIST, SLH-DSA (SPHINCS+), ML-DSA, and FALCON, the analysis focused on their structural properties, implementation overhead, and compatibility with Ethereum's account-based model. ML-DSA appeared to be the better choice, showing promise for deployment due to its balance of security and performance. A simulation-based implementation strategy was proposed and assessed across five blockchain configurations, allowing for empirical comparisons under realistic conditions.

While post-quantum schemes provide quantum resilience, they also introduce significant trade-offs particularly in terms of signature size, gas cost, and computational overhead, although keeping the expected performance better than that of a Post-Quantum adapted proof-of-work blockchain, namely in terms of block-time. PoW systems on the other hand, would require less effort to adapt to quantum threats.
These aspects must be carefully balanced against the operational constraints of Ethereum's network.
In addition, the thesis explored how post-quantum cryptography could augment privacy features, particularly in conjunction with zero-knowledge proofs. Although these areas are still under active research, integrating post-quantum ZKPs may play a pivotal role in the evolution of private, scalable, and quantum-resistant smart contracts.

Further work is required to refine transition strategies, reduce performance costs, and ensure compatibility with Ethereum's evolving consensus model. Beyond Ethereum, the broader blockchain ecosystem will benefit from standardized benchmarks and cross-platform integration efforts for post-quantum security.

Finally, this thesis hopes to contribute to a growing body of work that aims to future-proof decentralized infrastructure, ensuring that core technologies like Ethereum remain secure, functional, and trustworthy in a context where quantum adversaries may soon become a reality.

References

[1] P. W. Shor, "Algorithms for quantum computation: Discrete logarithms and factoring," *Proceedings of the 35th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 124–134, 1994. DOI: 10.1109/SFCS.1994.365700.

[2] F. Arute, K. Arya, R. Babbush, *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, pp. 505–510, 2019. DOI: 10.1038/s41586-019-1666-5. [Online]. Available: https://www.nature.com/articles/s41586-019-1666-5.

[3] I. Quantum, *Ibm quantum computing*, https://research.ibm.com/quantum-computing, Accessed: 2025-03-02, 2023.

[4] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, ACM, 1996, pp. 212–219. DOI: 10.1145/237814.237866.

[5] D. Aggarwal, G. K. Brennen, T. Lee, M. Santha, and M. Tomamichel, "Quantum attacks on bitcoin, and how to protect against them," *Ledger*, vol. 3, pp. 68–90, 2018. DOI: 10.5195/ledger.2018.127. [Online]. Available: https://doi.org/10.5195/ledger.2018.127.

[6] M. Mosca, "Cybersecurity in an era with quantum computers: Will we be ready?" *IEEE Security & Privacy*, vol. 16, no. 5, pp. 38–41, 2018. DOI: 10.1109/MSP.2018.3761723. [Online]. Available: https://doi.org/10.1109/MSP.2018.3761723.

[7] National Institute of Standards and Technology, *Nist asks the public to help future-proof cryptography*, Accessed: 2025-04-04, 2016. [Online]. Available: https://www.nist.gov/news-events/news/2016/02/nist-asks-public-help-future-cryptography.

[8] G. Alagic, D. Apon, D. Cooper, *et al.*, "Status Report on the Third Round of the NIST Post-Quantum Cryptography Standardization Process," National Institute of Standards and Technology, Tech. Rep. NIST IR 8413, 2022. DOI: 10.6028/NIST.IR.8413. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8413.pdf.

[9] "Fips 204: Module-lattice-based digital signature standard (ml-dsa)," National Institute of Standards and Technology, Tech. Rep., 2024. [Online]. Available: https://csrc.nist.gov/pubs/fips/204/final.

[10] "Fips 205: Stateless hash-based digital signature standard (slh-dsa)," National Institute of Standards and Technology, Tech. Rep., 2024. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.205.pdf.

[11] National Institute of Standards and Technology, *Post-quantum cryptography: Selected algorithms*, https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms, Accessed: 2025-06-12, 2024.

[12] G. Alagic, M. Bros, P. Ciadoux, *et al.*, "Status Report on the First Round of the Additional Digital Signature Schemes for the NIST Post-Quantum Cryptography Standardization Process," National Institute of Standards and Technology, NIST Interagency/Internal Report (NISTIR) 8528, Oct. 2024. DOI: 10.6028/NIST.IR.8528. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/ir/2024/NIST.IR.8528.pdf.

[13] C. Aguilar-Melchor, N. Aragon, P. Gaborit, and G. Zémor, "Efficient cryptographic signatures in the post-quantum code-based setting," in *Post-Quantum Cryptography – PQCrypto 2017*, T. Lange and T. Takagi, Eds., ser. Lecture Notes in Computer Science, vol. 10346, Springer, 2017, pp. 43–58. DOI: 10.1007/978-3-319-59879-6_3. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-319-59879-6_3.

[14] M. Baldi, P. Santini, D. Taufer, and F. Chiaraluce, "Cross: Code-based round signature scheme – specification," National Institute of Standards and Technology (NIST), Tech. Rep., 2023, Submitted to NIST for the Additional Post-Quantum Signature Schemes Call. [Online]. Available: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/CROSS-spec-web.pdf.

[15] C. Aguilar-Melchor, N. Aragon, P. Gaborit, and G. Zémor, "Less: Lattice- and error-correcting code-based signature scheme – specification," National Institute of Standards and Technology (NIST), Tech. Rep., 2023, Submitted to NIST for the Additional Post-Quantum Signature Schemes Call. [Online]. Available: https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/less-spec-web.pdf.

[16] P. Gaborit, C. A. Melchor, N. Aragon, *et al.*, *Hqc*, https://pqc-hqc.org/, Accessed: 2025-04-09.

[17] L. Lamport, "Constructing digital signatures from a one-way function," SRI International, Tech. Rep. CSL-98, Oct. 1979.

[18] D. Roh, S. Jung, and D. Kwon, "Winternitz signature scheme using nonadjacent forms," *Security and Communication Networks*, 2018. DOI: 10.1155/2018/1452457.

[19] V. Srivastava, S. Kumar, and A. Saxena, "An overview of hash-based signatures," *Cryptology ePrint Archive*, 2023, https://eprint.iacr.org/2023/411.

[20] D. J. Bernstein, A. Hülsing, S. Kölbl, R. Niederhagen, J. Rijneveld, and P. Schwabe, "The sphincs <sup>+</sup> signature framework," 2019. DOI: 10.1145/3319535.3363229.

[21] E. O. Kiktenko, "Sphincs$^+$ post-quantum digital signature scheme with streebog hash function," 2019. DOI: 10.48550/arxiv.1904.06525.

[22] M. Ajtai, "Generating hard instances of lattice problems," in *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing (STOC '96)*, ACM, 1996, pp. 99–108. DOI: 10.1145/237814.237838.

[23] O. Regev, "On lattices, learning with errors, random linear codes, and cryptography," in *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, ser. STOC '05, Baltimore, MD, USA: Association for Computing Machinery, 2005, pp. 84–93, ISBN: 1581139608. DOI: 10.1145/1060590.1060603. [Online]. Available: https://doi.org/10.1145/1060590.1060603.

[24] L. Ducas, V. Lyubashevsky, and T. Prest, "CRYSTALS-DILITHIUM: A Lattice-Based Digital Signature Scheme," in *Post-Quantum Cryptography - 9th International Conference, PQCrypto 2018*, ser. Lecture Notes in Computer Science, vol. 10786, Springer, 2018, pp. 238–268. DOI: 10.1007/978-3-319-79063-3_11. [Online]. Available: https://pq-crystals.org/dilithium/data/dilithium-specification-round3-20210330.pdf.

[25] P.-A. Fouque, J. Hoffstein, P. Kirchner, *et al.*, "Falcon: Fast-Fourier Lattice-based Compact Signatures over NTRU," Tech. Rep., 2018. [Online]. Available: https://falcon-sign.info/falcon.pdf.

[26] L. D. Feo, D. Kohel, A. Leroux, C. Petit, and B. Wesolowski, "SQISign: Compact post-quantum signatures from quaternions and isogenies," in *Advances in Cryptology – ASIACRYPT 2020*, ser. Lecture Notes in Computer Science, vol. 12491, Springer, 2020, pp. 64–93. DOI: 10.1007/978-3-030-64837-4_3. [Online]. Available: https://doi.org/10.1007/978-3-030-64837-4_3.

[27] D. Moody, *Nist standardization of additional signature schemes*, Presentation at the PQC Conference, Amsterdam, Accessed: March 2025, 2023. [Online]. Available: https://pkic.org/events/2023/pqc-conference-amsterdam-nl/pkic-pqcc_dustin-moody_nist_nist-standardization-of-additional-signature-schemes.pdf.

[28] N. I. of Standards and T. (NIST), *Fips pub 180-4: Secure hash standard (shs)*, 2015. [Online]. Available: https://csrc.nist.gov/publications/detail/fips/180/4/final.

[29] E. Foundation, *Ethash mining algorithm*, Accessed: 2025-03-16, 2025. [Online]. Available: https://ethereum.org/en/developers/docs/consensus-mechanisms/pow/mining/mining-algorithms/ethash/.

[30] G. Bertoni, J. Daemen, M. Peeters, and G. V. Assche, "The keccak secure hash algorithm - sha-3 submission," NIST SHA-3 Competition, Tech. Rep., 2011. [Online]. Available: https://keccak.team/files/Keccak-submission-3.pdf.

[31] N. I. of Standards and T. (NIST), *Fips pub 186-4: Digital signature standard (dss)*, 2013. [Online]. Available: https://csrc.nist.gov/publications/detail/fips/186/4/final.

[32] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ecdsa)," *International Journal of Information Security*, vol. 1, no. 1, pp. 36–63, 2001. DOI: 10.1007/s102070100002.

[33] V. S. Miller, "Use of elliptic curves in cryptography," *Advances in Cryptology—CRYPTO'85 Proceedings*, pp. 417–426, 1986.

[34] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of computation*, vol. 48, no. 177, pp. 203–209, 1987.

[35] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology - CRYPTO '87*, ser. Lecture Notes in Computer Science, vol. 293, Springer, 1988, pp. 369–378. DOI: 10.1007/3-540-48184-2_32.

[36] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, vol. 21260, 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[37] S. King and S. Nadal, *Ppcoin: Peer-to-peer crypto-currency with proof-of-stake*, White Paper, 2012. [Online]. Available: https://peercoin.net/assets/paper/peercoin-paper.pdf.

[38] C. Peikert, "A decade of lattice cryptography," *Foundations and Trends® in Theoretical Computer Science*, vol. 10, no. 4, pp. 283–424, 2016.

[39] D. J. Bernstein, A. Hülsing, J. Kelsey, *et al.*, "Hash-based signatures," in *Post-Quantum Cryptography*, Springer, 2015, pp. 109–128. DOI: 10.1007/978-3-319-22174-8_6. [Online]. Available: https://eprint.iacr.org/2015/705.

[40] J. Buchmann, E. Dahmen, and A. Hülsing, "XMSS - a practical forward secure signature scheme based on minimal security assumptions," in *Post-Quantum Cryptography (PQCrypto 2011)*, ser. Lecture Notes in Computer Science, vol. 7071, Springer, 2011, pp. 117–129. DOI: 10.1007/978-3-642-25405-5_8. [Online]. Available: https://doi.org/10.1007/978-3-642-25405-5_8.

[41] A. Hülsing, "Wots+ - shorter signatures for hash-based signature schemes," in *Progress in Cryptology – AFRICACRYPT 2013*, Springer, 2013, pp. 173–188. DOI: 10.1007/978-3-642-38553-7_10. [Online]. Available: https://eprint.iacr.org/2011/191.

[42] J. Hoffstein, J. Pipher, and J. H. Silverman, "Ntru: A ring-based public key cryptosystem," in *International Algorithmic Number Theory Symposium*, Springer, 1998, pp. 267–288.

[43] J. W. Cooley and J. W. Tukey, "An algorithm for the machine calculation of complex fourier series," *Mathematics of computation*, vol. 19, no. 90, pp. 297–301, 1965.

[44] L. Ducas, E. Kiltz, T. Lepoint, *et al.*, "Crystals-dilithium: Algorithm specifications and supporting documentation (round 3)," National Institute of Standards and Technology (NIST), Tech. Rep., 2021, Round 3 Submission to the NIST Post-Quantum Cryptography Standardization Project. [Online]. Available: https://pq-crystals.org/dilithium/data/dilithium-specification-round3.pdf.

[45] Bitcoin Developers, *Bitcoin developer guide: Block chain*, https://developer.bitcoin.org/devguide/block_chain.html, Accessed: 2024-06-12.

[46] V. Buterin, *Eip-197: Addition of modexp precompile*, https://eips.ethereum.org/EIPS/eip-197, Accessed: 2025-06-17, 2017.

[47] A. Vlasov, *Eip-2537: Precompile for bls12-381 curve operations*, https://eips.ethereum.org/EIPS/eip-2537, Accessed: 2025-06-17, 2020.

[48] eWASM Team, *Ewasm design overview*, https://github.com/ewasm/design, Accessed: 2025-06-17, 2019.

[49] N. Johnson, *Eip-609: Byzantium hard fork meta*, https://eips.ethereum.org/EIPS/eip-609, Accessed: 2025-06-17, 2017.

[50] T. V. Epps and T. Beiko, *Eip-3238: London hard fork meta*, https://eips.ethereum.org/EIPS/eip-3238, Accessed: 2025-06-17, 2021.

[51] S. Meiklejohn, M. Pomarole, G. Jordan, *et al.*, "A fistful of bitcoins: Characterizing payments among men with no names," *Communications of the ACM*, vol. 59, no. 4, pp. 86–93, 2016. DOI: 10.1145/2896384.

[52] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, and S. Capkun, "Evaluating user privacy in bitcoin," in *Financial Cryptography and Data Security (FC)*, Springer, 2013, pp. 34–51. DOI: 10.1007/978-3-642-39884-1_4.

[53] S. Goldwasser, S. Micali, and C. Rackoff, "The knowledge complexity of interactive proof systems," in *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing (STOC)*, 1985, pp. 291–304. DOI: 10.1145/22145.22178.

[54] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *2013 IEEE Symposium on Security and Privacy*, IEEE, 2013, pp. 238–252. DOI: 10.1109/SP.2013.47.

[55] E. Ben-Sasson, I. Bentov, Y. Horesh, and M. Riabzev, "Scalable, transparent, and post-quantum secure computational integrity," *Cryptology ePrint Archive*, no. 046, 2018. [Online]. Available: https://eprint.iacr.org/2018/046.

[56] Ethereum Foundation, *Zk-rollups*, https://ethereum.org/en/developers/docs/scaling/zk-rollups/, Accessed 2025-06-04, 2023.

[57] A. Chiesa, P. Mishra, N. Spooner, and T. Xie, "Post-quantum snarks from lattices," in *Advances in Cryptology – EUROCRYPT 2023*, Springer, 2023, pp. 113–143. DOI: 10.1007/978-3-031-30589-4_5.

[58] V. Lyubashevsky, A. Rondepierre, and G. Neven, *Lattice-based snarks and their application to verifiable machine learning*, Available at https://eprint.iacr.org/2023/309, 2023. IACR: 2023/309 (cs.CR).

[59] F. Benhamouda, L. K. G. Helms, R. Player, and M. Simkin, *Post-Quantum Zero-Knowledge and SNARKs from Lattices*, Available at https://eprint.iacr.org/2022/1055, 2022. IACR: 2022/1055 (cs.CR).

[60] M. R. Albrecht, B. Libert, and T. Peters, *LigeroLWE: Digital Signatures from Sublinear Lattice-Based Zero-Knowledge Proofs*, Available at https://eprint.iacr.org/2021/540, 2021. IACR: 2021/540 (cs.CR).

[61] M. Chase, E. Kiltz, and J. Loss, *Lattice-Based Succinct Non-Interactive Arguments via Delegation from Group to Ring*, Available at https://eprint.iacr.org/2022/310, 2022. IACR: 2022/310 (cs.CR).