BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF INTELLIGENT SYSTEMS
ÚSTAV INTELIGENTNÍCH SYSTÉMŮ

BLOCKCHAIN RESISTANT TO QUANTUM ATTACK

BLOCKCHAIN ODOLNÝ VŮČI KVANTOVÉMU ÚTOKU

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **MICHAL ĽAŠ**
AUTOR PRÁCE

SUPERVISOR **Mgr. KAMIL MALINKA, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2024

BRNO UNIVERSITY OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY

Bachelor's Thesis Assignment

155332

Institut:	Department of Intelligent Systems (DITS)
Student:	**Ľaš Michal**
Programme:	Information Technology
Title:	**Blockchain Resistant to Quantum Attack**
Category:	Security
Academic year:	2023/24

Assignment:

1. Learn about blockchain technology and the vulnerabilities a quantum attack can bring. Next, focus on existing blockchain implementations that claim to be resistant to quantum attack (e.g., Cardano, Ripple).
2. Learn about post-quantum cryptographic algorithms and discuss their suitability for use in blockchain.
3. Design your own quantum attack-resistant blockchain solution (with emphasis on ensuring integrity and security of transactions).
4. Implement the design. Existing libraries of post-quantum algorithms can be used. If too complex, some layers not related to security can be abstracted.
5. Test the functionality of the implementation, evaluate its performance and security parameters.
6. Compare the results with existing implementations and evaluate the effectiveness of the resulting solution.

Literature:

- NÚKIB: Útoky s využitím kvantového počítače mohou prolomit současné šifrování: řešením je včasná a efektivní implementace nových standardů. 5151/2023, E/630, Brno, strategická analýza, 2023
- NÚKIB: Kvantová hrozba a kvantově odolná kryptografie“, Příloha k dokumentu: Minimální požadavky na kryptografické algoritmy
- Brad Chase, & Ethan MacBrough. (2018). Analysis of the XRP Ledger Consensus Protocol. arXiv:1802.07242

Requirements for the semestral defence:
Items 1 to 3.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor:	**Malinka Kamil, Mgr., Ph.D.**
Head of Department:	Hanáček Petr, doc. Dr. Ing.
Beginning of work:	1.11.2023
Submission deadline:	9.5.2024
Approval date:	6.11.2023

Abstract

The development of quantum computers presents new opportunities but also introduces a threat by compromising currently used cryptographic algorithms. This threat significantly impacts blockchain technology, which relies on various cryptographic principles. The objective of this work is to design and implement a blockchain that incorporates new post-quantum cryptographic algorithms resistant to attacks performed by quantum computers. The main part of my solution involves analyzing blockchain components vulnerable to quantum attacks, selecting appropriate post-quantum algorithms, and subsequently implementing them within the blockchain. The result of this work is an overview of blockchain vulnerabilities and their solutions in the post-quantum era. Additionally, the implemented solution compares the performance of multiple post-quantum and currently used algorithms. The results show that post-quantum cryptography can have a significant impact on blockchain performance. However, post-quantum blockchains will still be usable, and they will withstand the era of quantum computers.

Abstrakt

Vývoj kvantových počítačov prináša nové možnosti s ktorými však prichádza aj hrozba v podobe prelomenia súčasne používaných kryptografických algoritmov. Táto hrozba v značnej miere ovplyvňuje aj technológiu blockchain, ktorá pre svoje fungovanie využíva množstvo kryptografických princípov. Cieľom tejto práce ja navrhnúť a implementovať blockchain, ktorý bude aplikovať nové post-kvantové kryptografické algoritmy, ktoré sú odolné aj voči útokom realizovaným kvantovými počítačmi. Podstatou môjho riešenia je analýza častí blockchainu, ktoré sú ohrozené kvantovým útokom, výber vhodných post-kavantových algoritmov a nakoniec ich implementácia v blockchain. Výsledkom tejto práce je hlavne prehľad možných zraniteľností a riešení pre blockchainy v post-kvantovej dobe. Navyše, implementované riešenie porovnáva výkonnosť viacerých post-kvantových aj súčasne používaných algoritmov. Výsledky ukazujú, že post-kvantová kryptografia môže mať významný vplyv na výkonnosť blockchainov. Avšak technológia blockchain bude naďalej použiteľná a určite prežije dobu kvantových počítačov.

Keywords

blockchain, post-quantum cryprography, quantum threat, post-quantum blockchain

Kľúčové slová

blockchain, kvantovo odolná kryptografia, kvantová hrozba, kvantovo odolný blockchain

Reference

ĽAŠ, Michal. *Blockchain Resistant to Quantum Attack*. Brno, 2024. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Mgr. Kamil Malinka, Ph.D.

Rozšírený abstrakt

V rýchlo sa rozvíjajúcom svete technológií ponúka vzostup kvantových počítačov nové riešenia pre komplexné problémy, ktoré sú náročne riešiteľné pre súčasné počítače. Tieto nové výpočetné možnosti však prinášajú aj nové hrozby. Vývoj kvantových počítačov môže predstavovať vážnu hrozbu pre súčasne používanú kryptografiu.

Blockchain je technológia pôvodne popularizovaná známymi kryptomenami ako Bitcoin alebo Ethereum. Technológia blockchain však našla využitie aj v mnohých iných aplikáciách. Blockchainy sú však veľmi závislé na mnohých kryptografických princípoch, ktoré ohrozuje vzostup kvantových počítačov.

Našťastie výskumníci a relevantné inštitúcie sú si vedomé hrozby, ktoré kvantové počítače predstavujú. Americký inštitút pre štandardy a technológie (National Institute of Standards and Technology – NIST) už niekoľko rokov aktívne pracuje na štandardizácii nových post-kvantových algoritmov, ktoré budú odolné voči útokom klasických, ale aj budúcich kvantových počítačov.

Táto práca sa venuje hlavne možnostiam integrovania novej post-kvantovej kryptografie do technológie blockchain. Hlavným cieľom je analyzovať kritické komponenty blockchainu ohrozené kvantovými útokmi, analyzovať vhodné post-kvantové algoritmy a prakticky ich implementovať do blockchainu. Implementované riešenie navyše porovnáva výkonnosť viacerých post-kvantových algoritmov spolu so súčasne používanými. Výsledkom je aj prehľad dopadov novej post-kvantovej kryptografie na blockchainy.

Druhá kapitola sa zaoberá základmi blockchainu, jeho fungovaním a vlastnosťami. Sú tu popísané štruktúry blokov, transakcií, účel konsenzus mechanizmu a peer-to-peer (P2P) siete.

Tretia kapitola pojednáva o kvantovej hrozbe a novej post-kvantovej kryptografii. Dôležité sú taktiež odporúčania od relevantných inštitúcií pre zaistenie post-kvantovej bezpečnosti.

Štvrtá kapitola spája predošlé dve dohromady. Venuje sa špecifickým komponentám blockchainu, ktoré sú ohrozené kvantovými útokmi a opisuje riešenia existujúcich implementácií, ktoré o sebe tvrdia, že sú odolné voči kvantovým útokom. Na záver tejto kapitoly sú diskutované vhodné post-kvantové algoritmy pre použitie v blockchainoch.

Piata kapitola opisuje môj návrh post-kvantového blockchainu s použitím post-kvantovej kryptografie a so zabezpečením všetkých komponent analyzovaných v predošlej kapitole.

Na záver šiesta a siedma kapitola obsahuje popis implementácie a testovanie navrhnutého riešenia.

Moje riešenie sa primárne zameriava na zaistenie bezpečnosti a integrity transakcií v blockchaine. Tento cieľ je dosiahnutý využitím novej post-kvantovej kryptografie, dodržania bezpečnostných odporúčaní od relevantných inštitúcií a zabezpečením všetkých kritických komponent blockchainu, ktoré sú ohrozené kvantovými útokmi. V implementácii sú zahrnuté post-kvantové algoritmy pre digitálne podpisy – Dilithium a Falcon vo všetkých ich bezpečnostných úrovniach. Pre porovnanie sú tu implementované aj súčasne používané algoritmy Ed25519 a ECDSA.

Počas testovania bola overená funkčnosť a výkonnosť implementovaného riešenia so všetkými spomenutými algoritmami. Výsledky ukazujú, že post-kvantová kryptografia má na rozdiel od súčasne používanej horšiu výkonnosť. Stále je však prakticky použiteľná. Hlavným problémom je veľkosť post-kvantových kľúčov a digitálnych podpisov. Táto vlastnosť post-kvantovej kryptografie spôsobuje, že v blockchainovej sieti sa musia prenášať väčšie objemy dát a celková veľkosť blockchainu taktiež výrazne rastie. Ďalšie zistenia a výsledky sú uvedené v kapitole testovanie a zhrnuté v samotnom závere práce.

Blockchain Resistant to Quantum Attack

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Mgr. Kamil Malinka, Ph.D. I have listed all the literary sources, publications, and other sources which were used during the preparation of this thesis.

.....
Michal Ľaš
May 7, 2024

Acknowledgements

I would like to sincerely thank my supervisor Mgr. Kamil Malinka, Ph.D. for all the advice and insightful comments. The same thanks go to consultant Ing. Martin Perešíni. Additionally, I am deeply thankful to my family, particularly my parents, for their unwavering support during my studies.

Contents

1 Introduction **4**

2 Blockchain technology **5**
2.1 Blockchain overview 5
2.2 Blockchain classification 7
2.3 Blocks 8
2.4 Transactions 9
2.5 Peer-to-peer network 11
2.6 Consensus mechanism 12

3 Quantum threat and post-quantum cryptography **15**
3.1 Quantum threat 15
3.2 Official recommendations 17
3.3 Key-encapsulation mechanisms 18
3.4 Digital signatures 20
3.5 Post-quantum cryptography comparison 21

4 Post-quantum blockchain **24**
4.1 Blockchain components affected by quantum threat 24
4.2 Existing post-quantum blockchains 25
4.3 Post-quantum cryptography for blockchain 26

5 Design of post-quantum blockchain **28**
5.1 Requirments 28
5.2 Performance testing design 28
5.3 Usage 30
5.4 Achieving quantum resistence 31
5.5 Blockchain design 32
5.6 Node design 32

6 Implementation **39**
6.1 Signer 39
6.2 Storage 40
6.3 Peer-to-peer network 40
6.4 Message processing 42
6.5 Consensus Mechanism 43
6.6 Wallet 44
6.7 Console Interface 44

7 Testing **46**
7.1 Functionality testing 46
7.2 Evaluating security parameters 47
7.3 Performance testing 47
7.4 Performance testing discussion 50

8 Conclusion **52**

Bibliography **53**

List of Figures

2.1 Linked chain of blocks – blockchain 5
2.2 Evolution from traditional ledger to blockchain [7] 6
2.3 Classification of blockchains [43] 8
2.4 Generic Chain of Blocks [54] 9
2.5 Merkle hash tree with transactions T_0-T_9. 10
2.6 The Transaction Journey [7] 10
2.7 Example of signing a transaction 11
2.8 Comparison of centralized and P2P network 12
2.9 Example of a blockchain valid nodes 13

5.1 High-level scheme of designed blockchain 29
5.2 High-level design of blockchain protocol 32
5.3 Design of blockchain node components 33
5.4 Account structure 35
5.5 Transaction structure 35
5.6 Block structure 36
5.7 Proposals structure 37
5.8 Designed message scenarios 38

6.1 Signer class diagram 39
6.2 Structure of data storage 40
6.3 Connection Manager tasks 41
6.4 Communication of Connection Manager and Message Processor 42
6.5 Steps of implemented consensus mechanism 45

7.1 Measured complexity of the validator application with different digital signature algorithms. 48
7.2 Measured memory allocation of the validator application with different digital signature algorithms. 49
7.3 Measured amounts of transferred data by the validator application with different digital signature algorithms. 50

Chapter 1

Introduction

In the rapidly evolving landscape of technology, the rise of quantum computing offers solutions to complex problems that are challenging or even impossible for classical computers. However, this great computational power also introduces a significant vulnerability. As quantum computing advances, it poses a significant threat to the cryptographic systems driving our digital world.

Blockchain is a technology initially popularized by cryptocurrencies like Bitcoin or Ethereum, but it has found usage in broader applications. However, blockchains strongly rely on various cryptographic principles, which makes them vulnerable to potential threats from quantum computers.

Fortunately, researchers are aware of the quantum threat, and institutions such as the National Institute of Standards and Technology (NIST) are actively working on establishing new standards for secure cryptographic algorithms against both quantum and classical computers.

This work explores the potential of integrating new post-quantum (PQ) algorithms into blockchain technology. The objective is to analyze blockchain components vulnerable to quantum attacks, identify appropriate PQ algorithms, and practically implement them within the blockchain. Additionally, the implemented solution compares the performance of multiple PQ and currently used cryptographic algorithms.

The second chapter provides an overview of key concepts and components of blockchain technology. It covers the structure of blocks, transactions, the purpose of consensus mechanisms, and peer-to-peer networks.

The third chapter discusses the quantum threat to existing cryptographic systems and the newly developed PQ cryptography algorithms. This chapter also includes recommendations from relevant institutions such as NSA or NIST. Furthermore, it thoroughly examines PQ algorithms selected as finalists of the NIST competition.

The fourth chapter delves into the specific components of a blockchain that are vulnerable to quantum threats. It explores solutions implemented by existing blockchains that claim to be quantum-resistant and analyzes appropriate PQ cryptography for inclusion in blockchains.

The fifth chapter outlines my design of a PQ blockchain, using PQ cryptography and securing all vulnerable components analyzed in the previous chapter.

Finally, the sixth and seventh chapter provides a detailed overview of the implementation and testing of the designed solution.

Chapter 2

Blockchain technology

This chapter serves as a brief introduction to the blockchain technology. It covers the basis of blockchain, how it works, what structures it uses, and what principles it employs.

2.1 Blockchain overview

Blockchain is a distributed ledger of cryptographically signed transactions grouped in blocks. These blocks are then linked together so that each block contains a cryptographic hash value of the previous block, as shown in Figure 2.1 [54]. The purpose of this design is to ensure the integrity of the blockchain. Even a minor change in any transaction will change the block's cryptographic hash. Since blocks are linked, changes cascade through the subsequent blocks, which makes it easy to detect any alternation. While it is technically possible to recalculate hashes for all blocks, blockchains employ one additional feature preventing such changes. This feature is called a consensus mechanism, and it ensures that changes made to the blockchain require majority agreement among its participants. The consensus mechanism will be discussed later in Section 2.6.

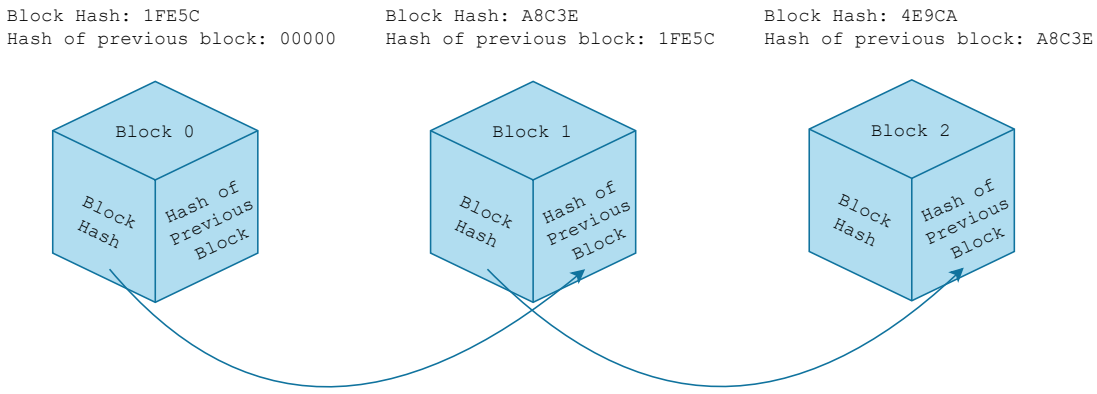

Figure 2.1: Linked chain of blocks – blockchain

From an evolutionary standpoint, a blockchain is characterized as a digital, distributed, decentralized ledger wherein individual participants share and update the ledger within a peer-to-peer (P2P) network according to specific protocols [7]. Figure 2.2 illustrates

the progression from traditional ledgers to blockchain technology. The peer-to-peer (P2P) networks topic will be covered later in Section 2.5.

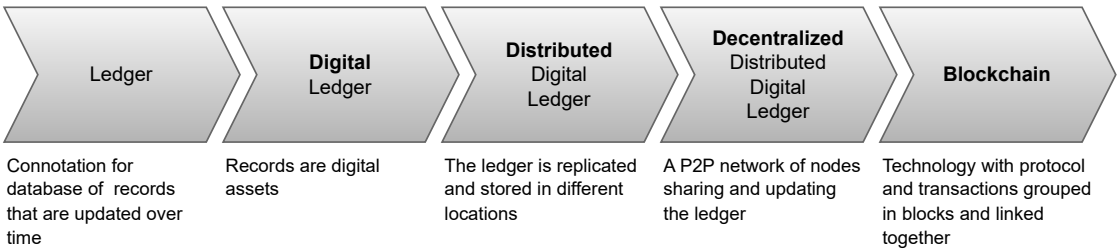

Figure 2.2: Evolution from traditional ledger to blockchain [7]

Blockchain technology contains its own key features such as [43]:

- **Security** – transactions must be cryptographically signed to prevent falsification and blockchain changes must be validated by the majority of participants or validators.
- **Transparency** – transaction history is available to all blockchain participants.
- **Decentralization** – there is no central authority controlling the validation of the transactions.
- **Immutability** – once the transaction has been validated and inserted into a blockchain, it is impossible to change or remove it.
- **Programmability** – ability to execute smart contracts.

Furthermore, blockchains do not depend on centralized trusted authorities to process transactions. Also, there is no need for third-party verification and validation of the transactions [43]. Verification and validation in blockchain are accomplished through mutual confirmation of transactions and consensus among participants in the network. Not all of these features are always applied to every blockchain. As will be discussed further in Section 2.2, different types of blockchain may have other purposes so that they may use just some of the mentioned features.

Blockchain technology became popular mainly because of the popular cryptocurrency Bitcoin, which idea was described by Satoshi Nakamoto in the original paper [35] from 2008. Nakamoto's paper presents an important idea and possible solutions for a decentralized payment system. Bitcoin as a cryptocurrency was created later in 2009 based on this idea. It is important to note that blockchain is not synonymous with cryptocurrency. While blockchain is the underlying technology for many cryptocurrencies, its applications extend beyond payments or currencies. For example, blockchains can be used for storing electronic medical records, digital certificates, tracking supply chain management, voting systems, and many other applications.

However, despite its unique features and popularity, blockchain technology is not suitable in all cases. Belotti et al. in [7] delve into the considerations for determining the suitability of blockchain for specific purposes. It includes a straightforward decision map that shows whether it is appropriate to use blockchain or traditional ledger technology.

2.2 Blockchain classification

There are two main classifications of blockchains based on access control: permission-less and permissioned blockchains [43].

2.2.1 Permission-less blockchain

A Permission-less blockchain network is accessible to anyone who wants to join the blockchain network. Anyone can publish a new block or create/read the transactions on the blockchain. Furthermore, there are no specialized nodes with any privileged rights. Because anyone can publish a new block, there has to be a mechanism that prevents malicious users from publishing blocks in a way that subverts the system. For this reason, an essential part of the permission-less blockchain is a consensus mechanism. An example of such a blockchain is a **public blockchain**. Well-known public blockchains are, for example, Bitcoin or Ethereum [54].

2.2.2 Permissioned blockchain

A permissioned blockchain network is accessible only with permission from some authority. Since only authorized nodes maintain the blockchain, it is possible to restrict read access, who can issue transactions, and who can publish a block. Additionally, it increases the level of trust in a blockchain. Permissioned blockchain also uses a consensus algorithm, but these methods often do not require as significant resources as in the case of permission-less blockchains [54]. Permissioned blockchain can be further divided into three categories: **private blockchain**, **consortium blockchain**, and **hybrid blockchain**.

Private blockchain

A private blockchain is a type of permissioned blockchain suitable, for example, for a smaller company. It can bring scalability because it does not influence speed and efficiency even if the network grows. Moreover, in case of any mishap, it is easy to identify the nodes involved in the transactions. The main disadvantage is that the whole blockchain is controlled by the organization as a centralized unit [43]. In addition, data on the blockchain has great redundancy. Each node on the blockchain can own a copy of the data which can serve as a backup. Well-known private blockchains are, for example, Hyperledger Fabric or Ethereum Enterprise.

Consortium blockchain

A consortium blockchain is similar to a private blockchain, but it is managed by multiple organizations. It incorporates features from both private and public blockchains. Some important aspects of the organizations could be private, while others can remain public [43]. Well-known consortium blockchains are, for example, Quorum or R3 Corda.

Hybrid blockchain

A hybrid blockchain takes the best from private and public blockchains. The consensus mechanism can be optimized since only a few nodes are authorized to validate the transaction [43]. As an example, it can serve the Dragonchain blockchain.

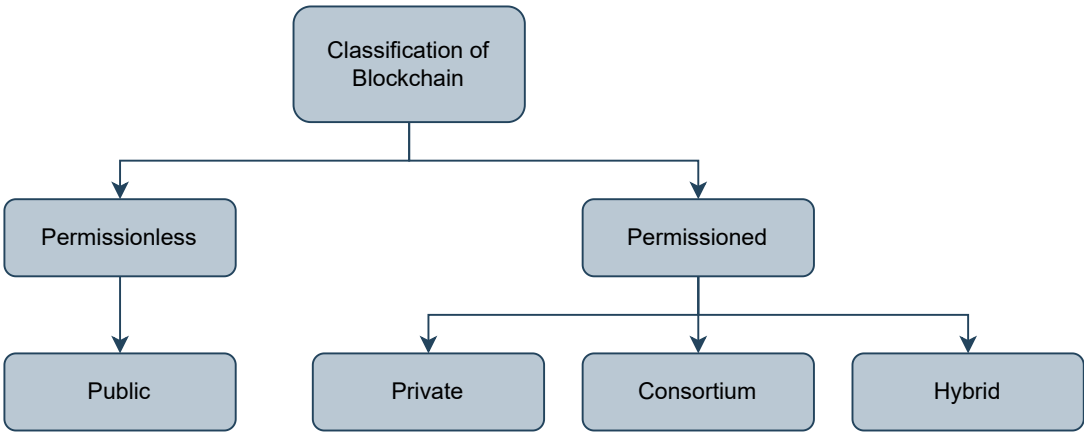

Figure 2.3: Classification of blockchains [43]

2.3 Blocks

A blockchain block is a data structure composed of transactions (data) and a header. The header contains metadata designed to identify the block and enable the verification of its content.

Each implementation of a blockchain can define its own block structure. However, some data fields are frequently utilized by the majority of them [54]:

- **Block Header**
  - The block number, also known as block height or sequence number. It is a unique identifier of the block.
  - Hash value of the previous block
  - A hash representation of the block data. It can be a hash of all the combined block data, or a common practice is to generate a Merkle tree and store the root hash there (more about Merkle trees is discussed further in this section).
  - A timestamp
  - The size of the block

  Blockchains utilizing the Proof-of-Work (PoW) consensus mechanism can also include a nonce value field. It is a number manipulated by the publishing node to solve the PoW puzzle.

- **Block Data**
  - A list of validated transactions and ledger events included within the block
  - Other data may be present (account balances etc.)

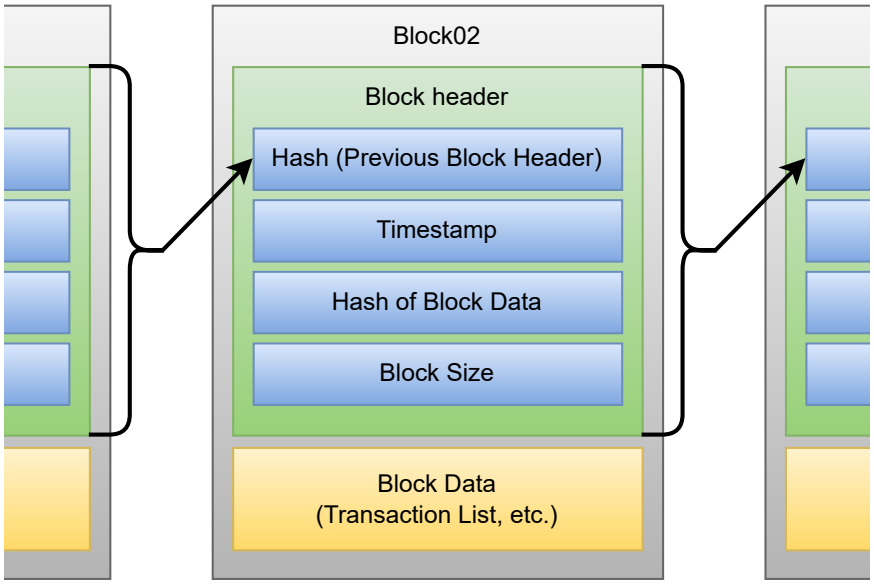

Figure 2.4: Generic Chain of Blocks [54]

Merkle trees

Merkle tree (patented in 1989 [34]) is a hash tree procedure that consists of data (transactions) that are iteratively hashed in pairs [7]. The transactions, represented as leaf nodes, are hashed, and then every two hashes are put in a pair and hashed again until everything is hashed in one root hash. If the number of transactions is odd, the last transaction is duplicated and hashed with itself. The example of a Merkle tree is shown in Figure 2.5.

The primary purpose of the Merkle tree root hash in a block header is to easily verify the integrity of the transaction in the block.

2.4 Transactions

Transactions come into place whenever users want to interact with one another in the blockchain network. Transactions do not have to be strictly financial and do not just carry and store transaction data. Different blockchain implementations can implement their own structure of transactions and the way transactions are performed. The usage of blockchain transactions is not limited, for example, a transaction can be a piece of code that will be executed after certain circumstances are met, this is also called a smart contract [7].

The journey of a transaction (also illustrated in Figure 2.6) [7]:

- *Creation:* The transaction's sender defines, according to blockchain transaction structure, the digital asset's source and destination.
- *Propagation:* The transaction is propagated through leading nodes[1] to validating nodes in a Peer-to-Peer (P2P) network.

[1]Nodes that are distributing transactions, blocks, and other blockchain messages

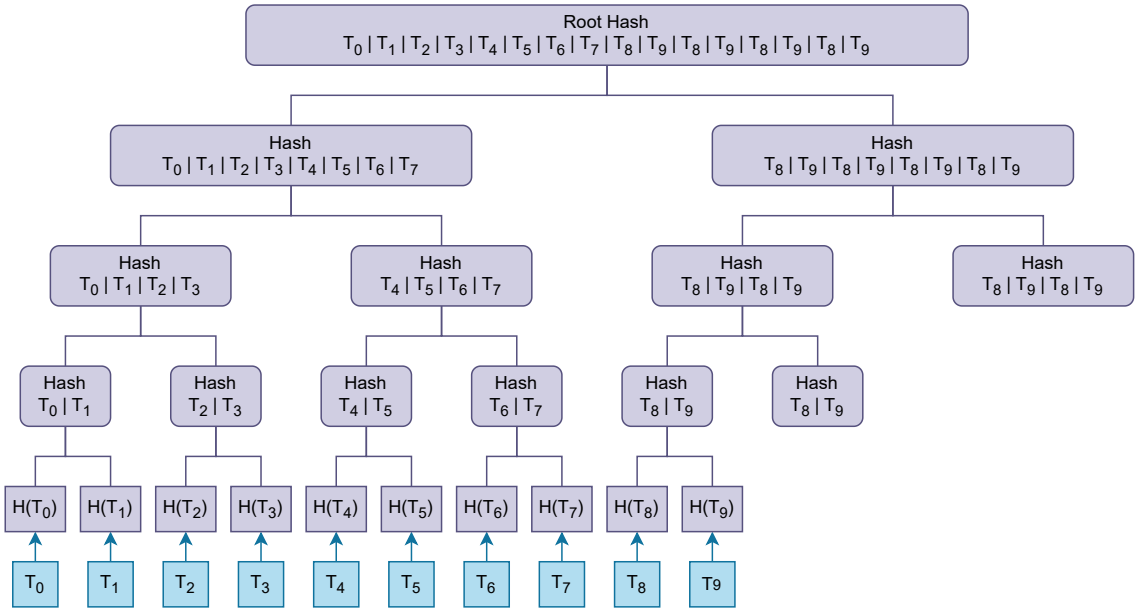

Figure 2.5: Merkle hash tree with transactions T_0-T_9.

- *Validation:* During this stage, transactions, organized into blocks, must successfully navigate the various phases of the envisioned consensus mechanism to be considered valid and, consequently, executable. Afterward, the block with transactions can be attached as the next block to the blockchain.
- *Propagation:* The block with the transaction is propagated through the blockchain network to let all nodes update their copy of the blockchain.
- *Confirmation:* The transaction is executed only if the block with transactions is validated and eventually published on the blockchain. If there are two valid chains, nodes must agree on a single chain of blocks (this is also the function of the consensus mechanism). Once confirmed, this block will become an immutable part of the final ledger version and may no longer be discarded.

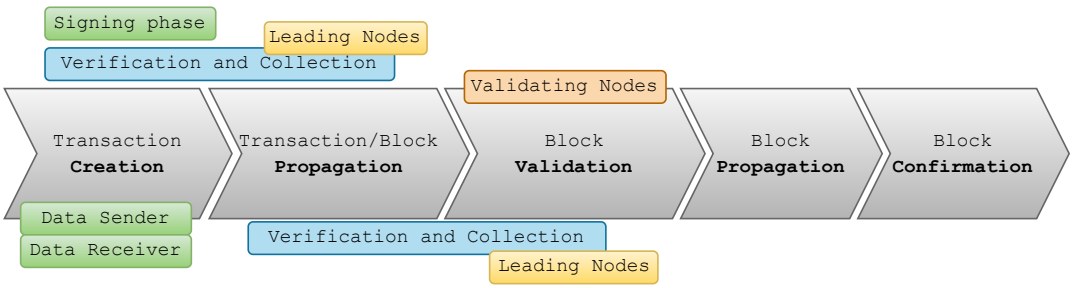

Figure 2.6: The Transaction Journey [7]

It is crucial to note transactions are critically dependent on cryptography. Each transaction requires a unique sender signature to prevent falsification and ensure integrity. If a transaction is altered or includes an invalid signature, it is rejected. Figure 2.7 shows how digital signature is performed. It is essential to securely store the private (secret) key to prevent potential attackers from gaining access. If the private key is stolen, an attacker could exploit the opportunity to sign transactions.

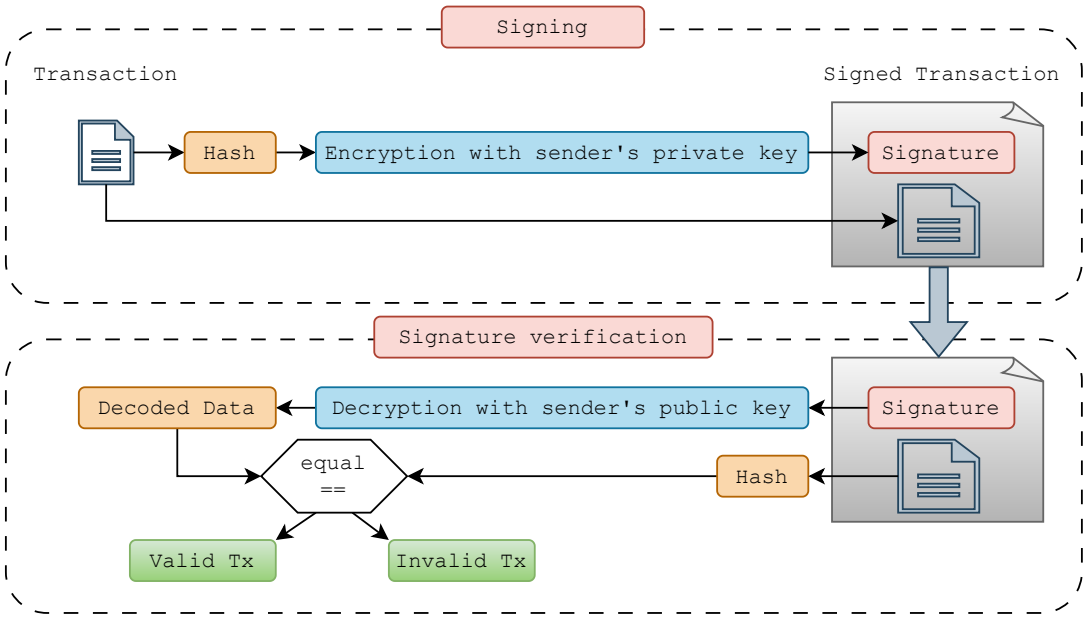

Figure 2.7: Example of signing a transaction

Another challenge that blockchain addresses is double-spending. In transactions involving electronic assets, the blockchain must verify that the sender is the legitimate owner of the asset. To guarantee ownership, the blockchain maintains a record of asset ownership. This is commonly addressed through data models like UTXO, Account, UTXO$^+$, or Key-value.

2.5 Peer-to-peer network

Peer-to-peer (P2P) networks are interconnected collections of nodes, which communicate together. Participants in P2P are known as peers. The main benefit of using the P2P network is file/data sharing. In centralized systems, the download mainly depends on the strength of the server, while in the decentralized P2P network, if one of the peers fails to operate, the other peers in the network can finish the required action [43]. The difference between centralized and decentralized (P2P) network is illustrated in Figure 2.8.

Blockchains utilize a P2P network to maintain decentralization. There is no central server functioning as an intermediary to process transactions; instead, transactions are distributed among nodes within the blockchain.

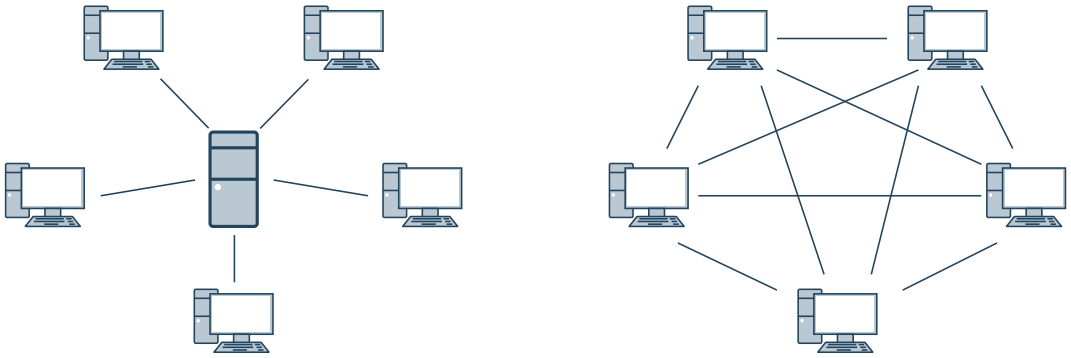

Figure 2.8: Comparison of centralized and P2P network

2.6 Consensus mechanism

Since there is no trusted third party or central authority a consensus mechanism is an algorithm based on predefined rules that determines the node that publishes the next block or ensures that nodes cooperate to create the next block. To add a new block to the blockchain, all nodes must reach a common agreement over time. In permission-less blockchains publishing nodes usually compete to publish the next block. They do this to get revenue through cryptocurrency or transaction fees. In permissioned blockchains, a consensus can be reached more easily because of the trust between nodes [54]. Nodes responsible for reaching consensus are called validators; in the case of the Proof-of-Work (PoW) consensus mechanism, they can be called miners.

The consensus mechanism also solves situations when different nodes publish a block at approximately the same time. This event is also called a fork. The example of a blockchain fork is shown in Figure 2.9. In most of the blockchain networks is „longer“ chain viewed as the correct one. The other chains will be considered invalid and discarded [54]. This mechanism relies on the assumption that the majority of nodes in the blockchain are honest so they always create longer chains than malicious ones.

A scenario in which a fork may occur is when a malicious node publishes a block containing invalidated transactions that were not propagated. If the malicious node eventually wins in the competition and publishes this block on the blockchain, other nodes with a different block will persist in trying to publish their own block, leading to the creation of two branches. One of these branches includes the block with invalid transactions. While the malicious node continues to compute new blocks on its branch, other nodes on the blockchain persist in working on the correct branch. Over time, the correct branch will naturally become longer and be chosen as the correct chain.

There are many types of consensus mechanisms. Each blockchain can implement its own to exactly meet its needs and purpose. The widely used consensus mechanisms are Proof-of-Work (PoW) and Proof-of-Stake (PoS), which are described more in detail below, but there are many others like Proof of Burn (PoB), Proof of Capacity (PoC), Proof of Space (PoSpace), and others. The final consensus mechanism among the three outlined

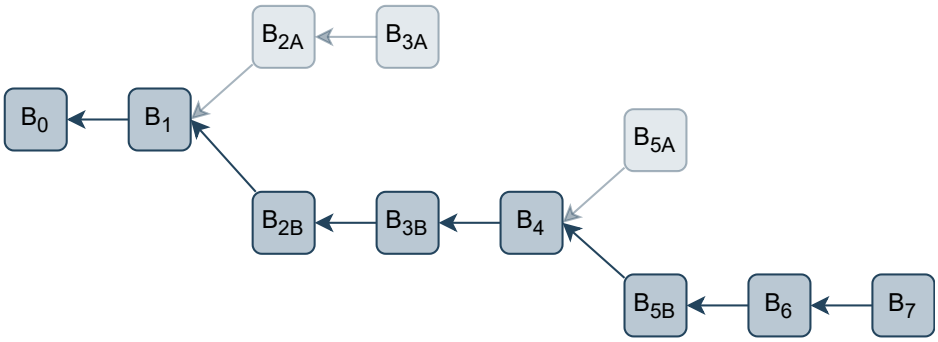

Figure 2.9: Example of a blockchain valid nodes

below is the XRP consensus mechanism. It claims more detailed exploration, as it is chosen for utilization in subsequent design and implementation.

Proof-of-Work (PoW)

The proof-of-work consensus mechanism operates on the principle of solving a computationally intensive puzzle. Nodes engaging in the solution of this puzzle are commonly referred to as miners or miner nodes. The successful solution of the puzzle serves as „proof“ that they have expended computational effort. It is based on the concept that computing the puzzle is hard, but verifying is easy [54].

As an illustration, a common method involves determining a nonce value for the block. The nonce is a value that, when hashed with the block's hash, produces a hash containing a specific number of leading zeros. The computational challenge lies in finding this nonce, which requires extensive trial and error. Conversely, the verification process is simple, it requires to calculate just one hash [54].

A notable drawback of the PoW consensus mechanism is its demand for computational power, which can lead to significant electricity consumption. This characteristic raises concerns about its environmental sustainability, as the energy-intensive character of the process is not aligned with eco-friendly practices.

Proof-of-Stake (PoS)

The proof-of-stake consensus mechanism is based on an election of the validator that validates the next block. Participation in the validator election requires the possession of a stake, denoting a certain amount of cryptocurrency. A higher stake means a higher probability of being elected as the next validator. This is far less computationally intensive than PoW, but it concentrates the validation among the wealthiest nodes. On the other hand, the wealthier nodes may have a vested interest in accurate validation [7].

XRP Ledger Consensus Protocol

XRP Ledger Consensus Protocol is a type of Federated Byzantine Agreement consensus mechanism. It consists of a special type of nodes called *validators* that are trusted by other nodes. Each participant on the network creates a unique node list (UNL) of chosen validator

nodes. This list is in full competence of each node so each node can choose other nodes it trusts. The philosophy behind this is that nodes gravitate toward reliable validators, while nonreliable validators have a smaller impact.

Validators receive candidate transactions and proposals. Candidate transactions are transactions seeking inclusion in the next version of the ledger[2]. Proposals are a special kind of message between validators. It contains a set of transactions that a validator proposes to other validators. Validators communicate and update proposals until a supermajority of chosen validators agree on the same set of transactions. Consensus is an iterative process. At the start of the first round, at least 50% of validators must agree. In subsequent rounds, this threshold is increasing up to 80%. Candidate transactions not included in the agreed-upon proposal remain candidate transactions and may be considered again in the next version of the ledger. This phase is called the **deliberation** phase [11, 12].

The next phase is called **validation**. In this phase, nodes decide whether to fully validate a ledger based on the validations issued by trusted nodes. It can be broken up into two parts [11, 12]:

- Calculating the resulting ledger version from an agreed-upon transaction set.
- Comparing results and declaring the ledger version validated if enough trusted validators agree.

If the network fails to achieve supermajority agreement on validations, it stops making further progress, and the consensus mechanism is repeated. Over time, there is an increasing likelihood that a majority of the validators within the network have received an identical set of candidate transactions. This ensures that each consensus round decreases disagreement among the validators [11, 12].

Notice that XRP consensus is built on a different principle than PoW or PoS. While in PoW and PoS, the next block publisher competes to become the next validator, in XRP, consensus nodes rather cooperate to achieve the consensus.

Blockchain technology summary

To summarize, a blockchain is a distributed, decentralized, immutable ledger of transactions organized into blocks and connected through cryptographic hashes. Fundamental elements of blockchain include asymmetric cryptography, digital signatures, and cryptographic hashes. Communication within the blockchain occurs through a peer-to-peer (P2P) network. The verification and validation of transactions on the blockchain are accomplished through a specific algorithm known as the consensus mechanism.

[2]In the case of XRP a ledger is something like a block. It represents a new blockchain state.

Chapter 3

Quantum threat and post-quantum cryptography

This chapter discusses the risks posed by quantum computers and the corresponding solutions. A significant concern is their potential to undermine current cryptography. However, there is promising progress in developing new post-quantum (PQ) cryptographic algorithms designed to withstand attacks from quantum computers.

The first Section 3.1 outlines challenges, threats, and existing solutions in the PQ world. It also provides an overview of the PQ algorithms that will be analyzed in this work. Section 3.2 delves into recommendations for PQ cryptography from relevant institutions. Sections 3.3 and 3.4 offer a more detailed examination of Key Encapsulation Mechanism (KEM) algorithms and digital signature algorithms, as they are particularly vulnerable to quantum threats. Lastly, Section 3.5 presents a comprehensive comparison, assessing the sizes of public and private keys, ciphertexts, signatures, and performance of the mentioned PQ algorithms. It concludes with a summary of this chapter and highlights the disparities between currently used cryptography and new PQ cryptography.

3.1 Quantum threat

Quantum computers are a new type of computers that utilize the principles of quantum mechanics. In the future, there are assumptions that quantum computers at some tasks will outperform the current microtransistor-based systems, which are already reaching their limits. Although new technologies bring new opportunities and solutions, they also bring new risks and vulnerabilities if misused. In the case of quantum computers, the risks are mainly associated with threats to the currently used cryptography [42].

A specific concern arises from the capability of quantum computers to solve certain mathematical problems much more efficiently than classical computers. Many existing encryption algorithms rely on these mathematical problems' complexity, which makes them vulnerable to quantum threats [14].

There are already algorithms designed specifically for quantum computers that pose a direct threat. Notably, Shor's algorithms [49] address the efficient factorization of large numbers and the Discrete Logarithm problem. These problems exhibit exponential complexity on classical computers but only linear complexity on quantum computers. This poses a direct risk to current asymmetric cryptography. Specifically for algorithms RSA and DSA, as well as elliptic curves algorithms Ed25519 or ECDSA. All currently used

asymmetric algorithms based on the problem of factoring large numbers and the problem of solving a discrete logarithm can be broken in the future precisely with the use of a quantum computer and Shor's algorithm.

Additionally, Grover's algorithm [21], a quantum search algorithm, can search an unsorted database with $O(n^{1/2})$ complexity [6]. This algorithm can be used to efficiently search for collisions in currently used hash functions, which pose a threat to applications that require non-collision hash functions.

From this point of view, the current situation is somewhat unusual. We know how to break current cryptographic algorithms such as RSA – we need to efficiently factor large numbers. We know an algorithm that can do this – Shor's algorithm. The only thing we still lack is a sufficiently powerful quantum computer. In the last year, IBM launched the Osprey quantum computer which has 433 qubits. However, Microsoft Research has calculated that around 2 500 qubits are needed to compute elliptic curve discrete logarithms to crack a standard 256-bit key. For 2048-bit RSA it is around 4000 qubits. Nevertheless, expectations are high. IBM wants to build a 100 000 qubit machine within 10 years and Google is targeting a million qubits by the end of the decade [32, 50].

The response to the quantum threat lies in quantum-resistant cryptography, often referred to as PQ cryptography. This form of cryptography is designed to provide robust encryption against the capabilities of both quantum and classical computers. Notably, the National Institute of Standards and Technology (NIST) has taken a significant initiative in this domain, launching a competition to standardize one or more quantum-resistant algorithms. From this competition, four algorithms have already been selected for standardization. The CRYSTALS-Kyber algorithm was chosen for the Key-Encapsulation Mechanism (KEM) category, and three algorithms – CRYSTALS-Dilithium, Falcon, and SPHINCS$^+$ – were selected for the digital signature category. While these are the currently chosen algorithms, other candidates from the fourth round of the competition are also noteworthy. While these algorithms have not yet been selected for NIST standardization, some of them are expected to be considered in the future.

These algorithms are slowly but surely making their way into existing cryptographic libraries. A notable project is the `PQClean` library [30, 29] which contains a collection of original implementations of algorithms from the NIST competition. PQ algorithms from this library will also be used in the subsequent implementation part of this work.

It is crucial to note that PQ cryptography currently represents the most practical approach to the quantum threat. However, the long-term resistance of PQ cryptography against quantum attacks remains uncertain [55].

For this thesis, the analysis will focus on algorithms from the NIST competition that advanced to the finals and the fourth round. The remaining algorithms from the third round will not be included in the analysis due to their similarity to the competition finalists and the shortcomings noted by NIST in the report from the third round of the competition [1]. Furthermore, despite the SIKE algorithm advancing to the fourth round of the NIST competition, shortly afterward, it faced a security breach driven by a classical computer attack. Consequently, SIKE is currently considered insecure and is not recommended for use. Security concerns, described in more detail by Nosouhi et al. in [39], also applies to the BIKE algorithm. Although both algorithms reached the fourth round of the NIST competition, they will not be part of the analysis because of their security concerns.

Two PQ digital signature algorithms, Extended Merkle Signature Scheme (XMSS) and Leighton–Micali Signatures (LMS), were already standardized by NIST in 2020. However,

these are categorized as stateful hash-based signatures, which makes them less suitable for general use due to their security reliance on careful state management [36].

3.2 Official recommendations

Although no standard for PQ algorithms has been released (the first NIST plans for 2024), relevant institutions have already provided recommendations for the transition to quantum-resistant encryption. This section will summarize the recommendations from the Czech *National Cyber and Information Security Agency(NCISA)* [40, 41], the National Authority for Cybernetic and Information Security, and the American National Security Agency (NSA) [37, 38].

3.2.1 Symetric encryption and hash functions

For symmetric encryption and hashing functions, the solution is relatively straightforward. Achieving quantum security in symmetric encryption involves using a sufficiently long encryption key. Similarly, for hash functions, maintaining quantum resistance is achieved by utilizing a sufficiently long output length.

Symetric encryption

The quantum resistance/vulnerability of block and stream ciphers, according to NCISA, can be summarized as follows: Ciphers with a key length of 256 bits are quantum-resistant, while those with key lengths of 128 bits and 192 bits are quantum-vulnerable. NSA in the Commercial National Security Algorithm Suite 2.0 (CNSA 2.0) only accepts AES-256 [41].

Hash functions

The quantum resistance/vulnerability of hash functions according to NCISA can be summarized as follows: Hash functions with output lengths of 384 bits or more are quantum-resistant, while those with key lengths of 256 bits or less are quantum-vulnerable. NSA in the CNSA 2.0 suit only accepts SHA-384 and SHA-512 hash functions [41].

3.2.2 Asymmetric cryptography

For asymmetric cryptography, the solution is no longer so simple. These cryptographic processes often rely on mathematically challenging problems that quantum computers can effectively solve. Consequently, ongoing efforts involve the development of new algorithms designed to withstand even the computational power of quantum computers. The earlier mentioned NIST competition is specifically dedicated to advancing solutions in this area.

PQ cryptography algorithms can be divided by their theoretical base to *Lattice-Based Cryptography*, *Hash-Based Cryptography*, *Code-Based Cryptography*, *Multivariate Cryptography*, and *Isogeny of Elliptic Curves*. For a more in-depth discussion on these bases, refer to [14, 55].

Asymmetric cryptography primarily finds application in the secure exchange of a symmetric key, referred to as the Key-Encapsulation Mechanism (KEM), and in the generation of digital signatures. Newly developed PQ asymmetric algorithms are specifically tailored for these fundamental use cases. More detailed descriptions of specific algorithms are provided in Sections 3.3 and 3.4.

3.2.3 Hybrid usage of classic and post-quantum cryptography

In the scientific community, there is a general opinion, particularly in the early stages, that quantum-resistant cryptography should be employed in conjunction with classical cryptography. This opinion is shared by the majority of European authorities, such as the German Federal Office for Information Security (BSI) or the French Agence nationale de la sécurité des systèmes d'information (ANSSI). The main reason for this opinion is the relative novelty of some PQ algorithms, which lack sufficient guarantees about the unsolvable mathematical problems upon which they are built. Notably, the occurrence of successful attacks using only classical computers highlights the importance of carefulness when adopting PQ cryptographic solutions [41].

The NSA holds a slightly different perspective on this matter, as it does not require the use of hybrid cryptography for the algorithms approved in the CNSA 2.0 document [37]. The agency justifies this standpoint by expressing confidence in the approved algorithms. Simultaneously, the NSA believes that overly complex protocols implementing hybrid cryptography could potentially reduce the performance and security of these approved algorithms [38].

3.3 Key-encapsulation mechanisms

This section provides a detailed exploration and key aspects of the selected PQ KEM algorithms, namely Kyber, McEliece, and HQC. The focus is placed on security, the sizes of public keys, the resulting ciphertext size, and the overall performance of each algorithm. The size of private keys is considered less crucial, as they are typically securely stored on the user's side, and their disk storage requirements are negligible in modern computing environments. For a more in-depth understanding of each algorithm, more details can be found on their official websites (Kyber [46], McEliece [2], HQC [33]) or in the report from the third round of the NIST competition [1].

For these algorithms (KEMs and digital signatures) NIST has defined 5 security levels [41]:

1. corresponds to the difficulty of a brute force attack on AES-128.
2. corresponds to the difficulty of generic collision search on SHA-256.
3. corresponds to the difficulty of a brute force attack on AES-192.
4. corresponds to the difficulty of generic collision search on SHA-384.
5. corresponds to the difficulty of a brute force attack on AES-256.

CRYSTALS-Kyber

Kyber is an IND-CCA2-secure[1] KEM, whose security is based on the hardness of solving the learning-with-errors[2] (LWE) problem over module lattices [46]. This algorithm comes with 3 security levels – level 1, 3, 5 which differ in the lengths of keys and cyphertexts.
The key characteristics of Kyber are [1]:

[1] Indistinguishability under Chosen-Ciphertext Attack – it is a level of security property that cryptographic encryption schemes aim to achieve.

[2] LWE is a mathematical problem used in quantum cryptography. More about this method is covered in Regev et al. [44].

- It is a lattice-based KEM chosen by NIST for standardization (draft of the upcoming standard [24]).
- It has a strong theoretical security foundation supported by decades of lattice cryptography literature.
- Public key and ciphertext sizes are on the order of a thousand bytes (see Table 3.1) which is comparable to other algorithms in the same category.
- Fast key generation, encapsulation, and decapsulation (see Table 3.3).
- Excellent performance overall in software, hardware, and many hybrid settings. Furthermore, researches indicates that the Module-LWE model, upon which KYBER is constructed, has a very good performance, which, however, does not dramatically affect the algorithm's security.

It is noteworthy to mention that the NSA approved the utilization of the Kyber level 5 algorithm in non-hybrid usage. The NSA's motivation primarily rests on the reliability of the Kyber algorithm, its security, and its efficacy [41].

Classic McEliece

Classic McEliece is a code-based KEM with an IND-CAA2 security level. This algorithm is already more than 45 years old, and there have been no cases of breaking it during this time. During the NIST competition, McEliece has been upgraded for greater efficiency and better security [14, 2]. Just like KYBER McEliece comes in 3 different security levels, but there are more variants. Each variant has a number next to the name, for example *mceliece348864*. The number *348864* refers to the key parameters. A higher number means better security and larger key sizes, which is the major problem with the McEliece algorithm.
The key characteristics of MecEliece are [1]:

- It is a code-based KEM that advanced to the fourth round of the NIST competition (draft of the possible standard [3]).
- Old and reliable algorithm, its security is supported by a long history of successfully withstanding cryptanalysis.
- Very large public key size and fairly slow key generation, but smallest ciphertext sizes from all mentioned algorithms (see Tables 3.1 and 3.3).

Hamming Quasi-Cyclic

HQC is a code-based KEM and follows an LWE-like encryption protocol. Just like KYBER HQC comes in 3 different security levels – level 1, 2, and 3. Level 1 was broken during the second round of the NIST competition, but levels 3 and 5 remain secure [14, 33].
The key characteristics of HQC are [1]:

- It is a code-based KEM that advanced to the fourth round of the NIST competition.
- Levels 3 and 5 appear secure and reliable, with no known security breaches.

- In comparison with KYBER, HQC features slightly larger public keys and significantly larger ciphertexts. However, these key sizes remain relatively small compared to the McEliece keys.
- It has good performance comparable with KYBER.

3.4 Digital signatures

This section thoroughly explores key aspects of the selected PQ digital signature algorithms: Dilithium, FALCON, and SPHINCS$^+$. Like the KEM algorithms, the primary focus is evaluating security, public key sizes, signature size, and overall algorithmic performance. The size of private keys is not that important, and it is for the same reason as with KEM algorithms. For a more in-depth understanding of each algorithm, more details can be found on their official websites (Dilithium [47], FALCON [18], SPHINCS$^+$ [48]) or in the report from the third round of the NIST competition [1].

CRYSTALS-Dilithium

Dilithium is a lattice-based digital signature algorithm based on the Fiat-Shamir with aborts technique of Lyubashevsky (see [31]), which uses rejection sampling to make lattice-based Fiat-Shamir schemes compact and secure. It comes in 3 security levels – level 2, 3, and 5 [1, 47].
The key characteristics of Dilithium are [1]:

- It is a lattice-based signature algorithm chosen by NIST for standardization (draft of the upcoming standard [23]).
- It has a relatively simple implementation and strong theoretical security.
- The public key and signature are larger than those in the algorithm, FALCON. Nevertheless, they still maintain an acceptable size (see Table 3.2).
- Among the mentioned algorithms, Dilithium has a very good performance and reliability (see Table 3.4).

Notably, the NSA approved using the Dilithium level 5 algorithm in non-hybrid usage. The NSA's motivation primarily rests on the reliability of the Dilithium algorithm, its security, and its efficacy [41].

Falcon

FALCON (Fast Fourier Lattice-based Compact Signatures over NTRU[3]) is a lattice-based signature scheme utilizing the "hash-and-sign" paradigm. It comes in 2 security levels – level 1, and 5 [1].
The key characteristics of FALCON are [1]:

- It is a Lattice-Based signature algorithm chosen by NIST for standardization.
- It has strong theoretical security, but more complex implementation.

[3]Number Theory Research Unit – it is an asymmetric cryptosystem that uses lattice-based cryptography.

- Keys and signature sizes are the smallest among all compared algorithms.
- Signature verifying is fast, but signing and key generation are slower than in the case of Dilithium (see Table 3.4).

SPHINCS$^+$

SPHINCS$^+$ combines the use of one-time signatures, few-times signatures, Merkle trees, and hypertrees. In contrast to the XMSS and LMS algorithms discussed earlier in this chapter, SPHINCS$^+$ is stateless. SPHINCS$^+$ comes in 3 security levels – level 1, 3, and 5, but there are also different variants based on the underlying hash functions, which are SHAKE256, SHA-256, or Haraka [48, 1].
The key characteristics of Falcon are [1]:

- It is a hash-based signature algorithm chosen by NIST for standardization (draft of the upcoming standard [25]).
- It has a complex implementation with numerous parameters for each security category. The complexity of the implementation also poses concerns about the overall security of the algorithm because it is hard to evaluate the whole scheme.
- Among the compared algorithms, SPHINCS$^+$ has the smallest public keys, measuring only in tens of bytes. However, the signatures are very large, ranging from 7 to almost 50 kilobytes.
- SPHINCS$^+$ also has fast key generation and signature verification, but the signing is much slower.

It is worth mentioning that the security of SPHINCS$^+$ also relies on the security of the underlying hash functions.

3.5 Post-quantum cryptography comparison

This section contains an overview of key sizes, ciphertext or signature sizes, and the performance of all previously mentioned PQ algorithms. The performance information for individual algorithms was sourced from the [52] website, and performance testing was performed on an Intel(R) Xeon(R) Platinum 8259CL CPU @ 2.50GHz. The comparison involved distributable values obtained from C-code implementations compiled with optimization for AVX2 vector instructions.

To summarize this chapter. The detailed PQ algorithms in this chapter offer the most practical approach to ensuring PQ resistance. Since these are relatively new algorithms, their use requires a certain degree of caution and it is also advisable to follow the recommendations of the relevant institutions. The information about the mentioned PQ algorithms will be used later to select suitable cryptographic algorithms for blockchain. Additionally, here is a short comparison, which highlights the differences between currently used cryptography and new PQ cryptography.

Currently, widely used algorithms for asymmetric cryptography include RSA (Rivest, Shamir, Adleman), DSA (Digital Signature Algorithm), and elliptic curve-based algorithms ECC (Elliptic Curve Cryptography). RSA is employed for asymmetric encryption and digital signatures, while DSA is only used for digital signatures. Elliptic curves can be

Algorithm	Claimed Security	Public key	Private key	Ciphertext
Kyber512	Level 1	800	1632	768
Kyber768	Level 3	1 184	2 400	1 088
Kyber1024	Level 5	1 568	3 168	1 568
Classic McEliece348864	Level 1	261 120	6 492	128
Classic McEliece460896	Level 3	524 160	13 608	188
Classic McEliece6688128	Level 5	104 992	13 932	240
Classic McEliece6960119	Level 5	1 047 319	13 948	226
Classic McEliece8192128	Level 5	1 357 824	14 120	240
HQC-128	Level 1	2 249	40	4 481
HQC-192	Level 3	4 522	40	9 026
HQC-256	Level 5	7 245	40	14 469

Table 3.1: Key and ciphertext sizes (in bytes) for the KEM algorithms [1]

Algorithm	Claimed Security	Public key	Private key	Signature
Dilithium	Level 2	1 312	2 528	2 420
	Level 3	1 952	4 000	3 293
	Level 5	2 592	4 864	4 595
Falcon-512	Level 1	897	7 553	666
Falcon-1024	Level 5	1 793	13 953	1 280
SPHINCS$^+$-128s	Level 1	32	64	7 856
SPHINCS$^+$-128f	Level 1	32	64	17 088
SPHINCS$^+$-192s	Level 3	48	96	16 224
SPHINCS$^+$-192f	Level 3	48	96	35 664
SPHINCS$^+$-256s	Level 5	64	128	29 792
SPHINCS$^+$-256f	Level 5	64	128	49 856

Table 3.2: Key and signature sizes (in bytes) for the digital signatures algorithms [1]

utilized alongside the Diffie-Hellman algorithm (ECDH) to exchange shared secrets or with the DSA algorithm (ECDSA) to make digital signatures. All these algorithms are based on the mentioned problems of factoring large numbers or the problem of solving a discrete logarithm.

Compared to currently used cryptography, PQ cryptography has lower performance and there are also large differences in the sizes of keys, ciphertext, and signatures. Typically, in current cryptography, the size of keys and digital signatures is in the order of hundreds to thousands of bits, significantly less than in PQ cryptography.

Particularly for blockchain technology, the size of keys and digital signatures can pose significant challenges. Since they are often stored within a blockchain, this can cause substantial growth in its overall size.

Algorithm	Keygen	Encapsulation	Decapsulation
Kyber512	29 172	36 768	26 943
Kyber768	45 407	54 332	42 098
Kyber1024	61 960	74 939	60 053
Classic McEliece348864	151 761 145	47 503	119 873
Classic McEliece460896	385 383 414	90 694	231 764
Classic McEliece6688128	591 004 800	191 851	273 034
Classic McEliece6960119	567 788 742	164 539	251 788
Classic McEliece8192128	625 667 532	203 624	268 867
HQC-128	104 115	197 030	360 575
HQC-192	244 636	459 309	766 797
HQC-256	447 179	845 083	1 425 978

Table 3.3: Performance of the KEM algorithms (in processor cycles) [52]

Algorithm	Keygen	Signing	Verifycation
Dilithium2	90 195	236 975	87 348
Dilithium3	153 215	380 755	144 980
Dilithium5	247 152	476 989	236 726
Falcon-512	21 234 790	888 844	143 976
Falcon-1024	63 158 867	1 800 943	292 065
SPHINCS$^+$-128s	5 9910 564	447 597 974	745 416
SPHINCS$^+$-128f	933 692	21 966 943	1 891 461
SPHINCS$^+$-192s	96 144 674	1 080 729 340	1 152 859
SPHINCS$^+$-192f	1 405 335	38 270 621	2 709 479
SPHINCS$^+$-256s	59 723 455	786 789 398	1 565 715
SPHINCS$^+$-256f	3 740 593	79 046 495	2 729 293

Table 3.4: Performance of the signature algorithms (in processor cycles) [52]

Chapter 4

Post-quantum blockchain

This chapter serves as a bridge between the preceding ones. Its primary objective is identifying parts of blockchain technology that may be vulnerable to quantum threats, as explained in Section 4.1. Subsequently, Section 4.2 explores blockchains that claim to be quantum-resistant. This section also discusses well-known blockchains that are considering adopting PQ cryptography, explaining their concerns and the cryptography they are currently using.

Since blockchain technology heavily depends on cryptography, Section 4.3 focuses on selecting appropriate PQ algorithms for inclusion in blockchains among those mentioned in the previous chapter.

4.1 Blockchain components affected by quantum threat

Block hashes and Merkle trees – Each block in a blockchain contains a hash representing data within that block, as well as the hash of the previous block, typically calculated using the Merkle tree procedure. As mentioned in the Section 3.1 Grover's algorithm can be employed to search for hash collisions, which may enable the replacement of a block in the blockchain while maintaining its apparent integrity [16]. To mitigate this threat, it is crucial to employ a hashing function with an adequate output length, as detailed in Section 3.2.1.
Transaction signatures – Each transaction in a blockchain must feature a digital signature. To ensure PQ resistance signatures have to be created by a quantum-resistant algorithm, these algorithms were discussed in Sections 3.4.
Transactions confidentiality – For blockchains prioritizing data confidentiality, attention must be given to communication encryption. The KEM algorithms discussed in Section 3.3 are employed to ensure the secure exchange of a symmetric key. Recommendations for symmetric cryptography were mentioned in Section 3.2.1.
Consensus mechanism – Regarding consensus mechanisms, in the case of the PoW consensus mechanism, it is advisable to use a PoW variant that does not grant quantum computers an advantage over classical ones.

For example, as previously discussed in the Section 3.1, Grover's algorithm can accelerate the generation of hashes, which poses a challenge for blockchains employing a PoW consensus mechanism or a similar approach. A user equipped with a quantum computer could gain a significant advantage, potentially leading to dominance over the blockchain network by outperforming classical computers [16] .

A notable alternative is the Lattice-based Consensus mechanism, or Lattice-based Proof-of-Work (LPoW), which is quantum-safe. LPoW presents a challenging puzzle for both clas-

sical and quantum computers; this ensures security while the verification process remains simple as in the case of traditional PoW consensus mechanisms [19].

In the case of PoS and other consensus mechanisms that use the concept of randomness, it is important to choose a reliable random generator. Theoretically, PQ computers will be able to find the deterministic nature of the pseudo-random generated values, as long as this process is based on the phenomenon of classical physics [28]. In PoS, this vulnerability could empower a malicious actor to manipulate stake height to increase the likelihood of being more frequently chosen as the final validator. This issue is not restricted to consensus mechanisms; many cryptographic algorithms rely on pseudo-random number generators. However, the solution lies in quantum random generators, which are also a hot topic in the PQ era. The key is to integrate them into existing solutions and use them instead of current pseudo-random generators as soon as they are available.

Some consensus mechanisms, like PoS, Ouroboros, or XRP consensus mechanism, also use digital signatures. As for transactions, it is crucial to use PQ digital signature algorithms.

Other options for PQ consensus mechanisms are discussed by Gomes et al. in [20], which also deals with quantum consensus mechanisms based on quantum computations.

4.2 Existing post-quantum blockchains

Many well-known blockchains realize the PQ threat and are actively preparing to secure against quantum attacks. However, the practical implementation of PQ algorithms in blockchains presents numerous challenges, including speed, key size, signature size, and ciphertext size concerns. Doubts persist about the overall security of existing PQ algorithms. Additionally, the development of quantum computers capable of executing substantial quantum attacks is anticipated to occur within a timeframe of 10 to 30 years[1] [16].

For these reasons, well-known blockchains such as Cardano, Ripple (XRP Ledger), and Hedera are cautious about implementing PQ algorithms. Cardano is willing to transition to PQ cryptography but is awaiting the standardization of new PQ algorithms from the NIST competition. Currently, Cardano employs the Edwards-curve Digital Signature Algorithm (EdDSA), specifically Ed25519. The consensus mechanism, Ouroboros, operates on a principle similar to PoS and is considered usable even in the era of quantum computers [26].

Similarly, Ripple (XRP Ledger) is considering implementing PQ algorithms but believes the current ones are inefficient, fearing a decline in blockchain performance. Ripple presently employs the ECDSA or EdDSA algorithms. Despite this, Ripple's consensus mechanism, which was described in Chapter 2.6, is believed to be secure even in the quantum computing era [53].

Hedera, so-called as a „third generation blockchain", expresses interest in adopting the Falcon algorithm due to its balance between security and key size. However, Hedera awaits the NIST standard for Falcon algorithm which is expected in 2024 before implementing it [55].

Some blockchains, like Quantum Resistant Ledger (QRL), have already implemented PQ-resistant algorithms. QRL uses the Extended Merkle Signature Scheme (XMSS), a standardized PQ stateful algorithm, for digital signatures. As a consensus mechanism, it uses PoS [55].

[1] The estimated timeline for the development of truly efficient quantum computers varies among different sources. Some experts claim that a truly efficient quantum computer may never be built.

IOTA, until 2021, utilized the Winternitz One Time Signatures (WOTS) algorithm, which is considered resistant to quantum attacks. However, in 2021, IOTA transitioned to EdDSA for better performance and reduced transaction size. IOTA currently uses a PoW mechanism but plans to adopt PoS in IOTA 2.0 [27].

Corda, in contrast, introduced a PQ algorithm based on SPHINCS$^+$, providing users with choices like RSA, ECDSA, EdDSA, and SPHINCS$^+$ for digital signatures. EdDSA is the default, with SPHINCS$^+$ as a PQ alternative [55].

In summary, current blockchain implementations often withhold from adopting new PQ algorithms due to significant drawbacks and pending standards. Some blockchains implement already standardized PQ algorithms like WOTS, WOTS+, or XMSS. However, these algorithms come with their own set of drawbacks. WOTS operates as a one-time signature scheme, necessitating the use of a different private key for each signature. XMSS is a more complex stateful algorithm derived from WOTS and also requires key management and periodic updating. From this point of view, this work is unique for its original focus on implementing new PQ algorithms in blockchain. The Table 4.1 offers a summary of the mentioned blockchain platforms. The green cells indicate the utilization of PQ resistant algorithms, while red cells signify algorithms that are considered insecure in the PQ era. Notice that Hedera and Corda have empty cells in the consensus mechanism column. This is because they employ custom consensus mechanisms which were not analyzed for their PQ resistance yet.

Blockchain	**Utilized cryptography**	**Consensus mechanism**
Cardano	Ed25519	Ouroboros
Ripple (XRP)	ECDSA, Ed25519	XRP Consensus Protocol
Hedera	Falcon	
QRL	XMSS	PoS
IOTA	Ed25519	PoW
IOTA 2.0	Ed25519	PoS
Corda	ECDSA, RSA, Ed25519, SPHINCS$^+$	

Table 4.1: Summary of existing post-quantum blockchains

4.3 Post-quantum cryptography for blockchain

To create an efficient PQ blockchain, several factors are important [16]:

- Small encryption key sizes. Devices interacting with the blockchain need to store private keys and often multiple public keys. Particularly for Internet of Things (IoT) devices, small key sizes are essential for space-saving and efficient work with these keys.
- Small sizes of digital signatures. Since every blockchain transaction requires a digital signature, the size of these signatures significantly impacts the size of a block and the entire blockchain.
- Fast execution. Post-quantum schemes must be optimized for speed to enable the blockchain to process a maximum number of transactions.

- Low computational complexity. This ties in with the previous point, but it is important to note that certain post-quantum schemes may work more efficiently on specific hardware without necessarily being computationally simple.
- Low energy consumption. Ensuring the long-term sustainability of the blockchain requires energy-efficient computing processes.

However, achieving these ideal properties is challenging in practice, especially concerning key size, digital signatures, and efficiency. It often involves a balanced trade-off between security and practicality, or key size and performance [55].

Key-Encapsulation Mechanism (KEM)

In the KEM category, the optimal choice among the algorithms is quite evident. While HQC shares similarities with KYBER, KYBER holds a slight advantage across all measurements. McEliece stands out for its compact ciphertexts and longstanding reliability, but its drawback lies in the size of public keys. So, KYBER emerges as the most favorable choice. It offers a balanced combination of reasonable public key and ciphertext sizes with very good performance. KYBER is also trusted by the NSA and other institutions as a safe and reliable choice.

Digital signatures

In the case of algorithms for digital signatures, the decision is no longer as straightforward. In the blockchain, where each transaction requires a digital signature, the SPHINCS$^+$ algorithm could be impractical due to excessively large signatures.

So, the primary argument arises between the FALCON and Dilithium algorithms. Both exhibit comparable properties, with FALCON having slightly smaller public keys and signatures, while Dilithium has slightly better performance. Both algorithms are trusted by relevant institutions, with Dilithium being particularly favored by the NSA. Ultimately, both FALCON and Dilithium are deemed as the best choices for deployment in the blockchains.

Chapter 5

Design of post-quantum blockchain

This chapter presents my design for a PQ blockchain. It consists of four sections. The initial Section 5.1 describes the requirements and goals of this design. Since one of the main goals is to test the performance Section 5.2 describes the strategy and chosen evaluation metrics. The subsequent Section 5.3 outlines the possible usage of this design. Section 5.4 details my approach to achieving PQ security, with the description of employed algorithms, cryptography, etc. The last two Sections 5.5 and 5.6 defines my design of the entire blockchain with all its components and data structures.

5.1 Requirments

The main requirement is to achieve PQ resistance of the blockchain with an emphasis on ensuring the integrity and security of transactions. This goal will be achieved by integrating new PQ cryptography and adhering to practices outlined in Section 4.1. Figure 5.1 illustrates the high-level blockchain scheme with highlighted components on which the main focus will be placed. This blockchain stack is inspired by the stacked model presented in the Article by Homoliak et al. [22].

Additionally, particular emphasis will also be placed on evaluating the performance of this blockchain with the usage of various PQ digital signatures as well as with digital signatures currently in use.

Recognizing the complexity of developing an entire blockchain system, certain aspects of the final implementation will be abstracted. Additionally, these abstractions will simplify the subsequent testing. All used abstractions will be clearly marked and supplemented with proposals for full functionality.

5.2 Performance testing design

Testing of a whole blockchain can be a complex task. There are various metrics and properties that can be considered. Also, there are multiple testing methods like empirical analysis, live monitoring, experimental analysis, simulation, mathematical modeling, etc. Often monitored properties are transaction execution time, latency, transaction throughput, and scalability. However, these properties mostly depend on the utilized consensus mechanism [15]. For this work, the important metric is the performance of different digital signature algorithms within the blockchain.

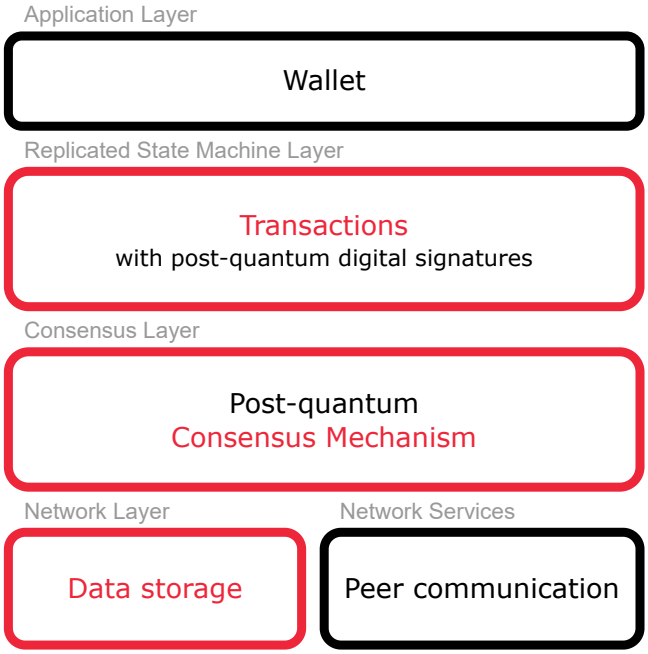

Figure 5.1: High-level scheme of designed blockchain

The monitored attributes will be similar to those in the analysis made by Chandel et al. [10], who compared the performance of the Rivest-Shamir-Adelman (RSA) algorithm and the Elliptic-Curve Cryptography (ECC). Their comprehensive analysis results are based on the key sizes and performance of *key generation*, *sign*, and *digital signature verification* operations [15]. The same attributes were observed by the Open Quantum Safe project [52] which has created benchmarks and has compared new PQ algorithms from the NIST competition.

It would make sense to compare even cryptography hash functions. However, this comparison was already done by Ferreira et al. [17], who conducted a study on blockchain-based IoT to explore the performance of hash functions in blockchains. Particularly, the authors developed a blockchain in an IoT scenario to evaluate the performance of different cryptographic hash functions such as MD5, SHA-1, SHA-224, SHA-384, and SHA-512. The results show that SHA-224 and SHA-384 are the best hash functions for blockchains due to their lack of collision attacks [15]. However, from these algorithms, only SHA-386 and SHA-512 are recommended for use in the PQ era.

Direct comparison with different blockchain platforms is also difficult. The main problem is large differences in consensus mechanisms. Additionally, each blockchain may serve distinct purposes [15]. Specifically for this design and implementation, direct comparisons with other blockchains might be misleading due to the used abstraction, which may lead to inaccurate results. Therefore, the comparison focus will be only on different digital sig-

nature algorithms, as this represents the most significant difference from current solutions. Especially compared will be new PQ algorithms with currently widely used ECC.

5.3 Usage

Blockchain can have various uses. I decided to implement proposed blockchain as an open cryptocurrency for exchanging digital funds using an Account-based model. To simplify the implementation, there will be several related abstractions:

1. Blockchain will be started with a fixed number of pre-initialized accounts. Each node's account database will be then pre-initialized with these accounts. This ensures easier testing, as it will be possible to set the attributes of individual accounts. Furthermore, nodes will not have to discover and synchronize with each other, because everything will be manually set. Adding new accounts to a running blockchain can be achieved by treating the account creation process as a transaction. Such a transaction would also go through the consensus process, wherein validators verify if the same account does not already exist. Upon successful validation and creating a new block, all validators update their account databases to include the newly created account.

2. Since individual nodes will be pre-initialized, there will not be a mechanism for node discovery within the blockchain network. However, a possible solution can involve new nodes attempting to connect to specific, publicly known, and trusted nodes. These nodes can then provide a list of other nodes available for connection.

3. This design also lacks a mechanism that would ensure the synchronization of a new node that joins the blockchain network after the blockchain has made some progress. Synchronization involves a node contacting its peers to obtain valid blocks and accounts created during its absence or inactivity. This functionality is not implemented under the assumption that during testing there will be a fixed number of nodes initialized simultaneously. However, this mechanism can be integrated into the current implementation. The key is to ensure that nodes synchronize only with valid blocks and accounts. This can be achieved by having the node contact trusted peers (peers included in its Unique Node List[1]), and if 80% of these peers report the same blocks and accounts, they can be considered valid.

With the mentioned abstraction this will be the primary scenario for using the proposed design:

1. All nodes of the blockchain will be launched simultaneously. Before the actual launch, the account databases of each node will be pre-initialized with existing accounts.

2. After the pre-initialization nodes will create connections with each other, they will start communicating and creating transactions.

3. Subsequently, it will be possible to stop the nodes, check the results, and examine the final blockchain state.

4. The process will be repeatable, for example with a different algorithm for digital signatures.

[1]UNL is a list of peers trusted by some node. Each node can create its own UNL. This concept was described in Section 2.6 as a part of XRP ledger consensus protocol.

Theoretically, until the nodes are stopped, the entire network can continue functioning. This scenario is designed specifically for testing.

5.4 Achieving quantum resistence

This section will detail the approach I have taken to secure the essential components of the blockchain against quantum attacks, as previously outlined in Section 4.1. It will also discuss the PQ algorithms that I decided to select and provide an explanation for my choices.

Hash function

As a hash function, I decided to utilize the SHA-512, primarily because it aligns with the recommendations from the NCISA and the NSA also recommends it in their CNSA 2.0 suite.

Post-quantum cryptography

I decided to utilize several algorithms for PQ digital signatures, the performance of which will be compared in the testing chapter. These are Falcon and Dilithium algorithms in all their levels, namely Falcon-512, Falcon-1024, Dilithium2, Dilithium3, and Dilithium5. Additionally, I have decided to compare these PQ algorithms with the currently prevalent ECDSA and EdDSA algorithms, specifically Ed25519, commonly used in existing blockchains. Despite recommendations, I have decided against implementing a hybrid use of PQ and current cryptography due to potential implementation complexities and also to preserve the meaning of the subsequent comparison of PQ and current algorithms.

Consensus mechanism

As for the consensus mechanism, I have decided for the algorithm employed by Ripple (XRP Ledger), which was detailed in Section 2.6. Even though this mechanism is a little more complex, it is very effective. In terms of PQ resistance, it does not use the principle of PoW or random selection of validators, but validators cooperate to achieve the consensus. However, this consensus uses a special type of messages called proposals that require digital signatures, so even there has to be employed mentioned PQ cryptography.

Transactions confidentiality

Ensuring confidentiality of transactions will not be a part of this implementation, as it primarily pertains to permissioned blockchains, and this aspect is already addressed by new PQ protocols such as KEMTLS, which ensure encrypted communication. Especially KEMTLS (see [51]) is interesting mainly because for authentication it uses PQ KEM algorithms instead of digital signatures since cyphertexts of KEM algorithms are usually almost 2 times smaller than PQ digital signatures.

Quantum entropy

Section 4.1 also mentioned doubts about the security of pseudo-random generators in the age of PQ computers. The consensus mechanism I chose does not use the concept of randomness.

However, entropy is necessary for the used cryptographic algorithms. My design still uses a pseudo-random entropy generator provided by the operating system, because quantum entropy generators are not widely available. Nevertheless, Allende et al. in [4], deals with a similar topic to this thesis, and presents a solution with the implementation of a server that serves as a source of PQ entropy.

5.5 Blockchain design

This Section serves as an introduction to the designed blockchain, with its high-level illustration in Figure 5.2. A node has a copy of the blockchain (chained blocks with transactions) and the account data of all nodes in the blockchain network. The node communicates with other nodes in the P2P blockchain network. The important messages are transactions and proposals. When a node obtains a transaction it verifies it and broadcasts to other nodes. Proposals are messages specific to the chosen consensus mechanism, which is the XRP Consensus Protocol described in Section 2.6. Nodes exchange proposals to reach a consensus with each other. Finally, to ensure that a node works properly it is necessary to keep it in synchronization with other nodes in the network. All details about the functioning of a node will be described in the subsequent Section 5.6.

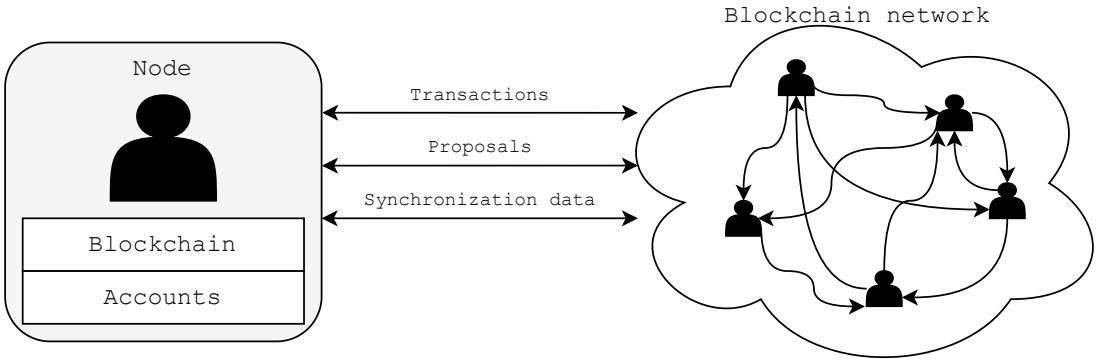

Figure 5.2: High-level design of blockchain protocol

5.6 Node design

The design is segmented into several distinct components, as illustrated in Figure 5.3. Each link symbolizes communication between these components. Detailed explanations of these components and their functionalities are provided later in this section, along with data structures used for blocks, transactions, proposals, and messages.

The illustrated design corresponds to the implementation of the validator node. However, I also considered other types of nodes:

- *Validator node:* Holds the full copy of the blockchain, broadcasts received transactions to other validators, can create its own transactions, and most importantly participates in validation and the process of reaching consensus.

- *Server node:* Do the same as a validator node including consensus, but with the important change. Servers do not create proposals for consensus. They observe proposals created by their chosen validators, ensuring they remain informed and updated with the network's activity.
- *Stock node:* These are just user applications that can connect to a server node or directly to a validator and create transactions.

Although these different types of nodes are designed, only the validator node will be implemented, as it contains all the necessary functionality for testing. Implementation of the server node and stock node would make sense without using the mentioned abstraction. In this scenario, server nodes would be intermediaries for distributing blockchain state information to stock nodes. Most of the code implementation would be the same as for the validator node. However, it would be necessary to add special messages that would be used for communication between the server and the stock node. For example, a request for information about the current state of blockchain, individual accounts, transactions, etc.

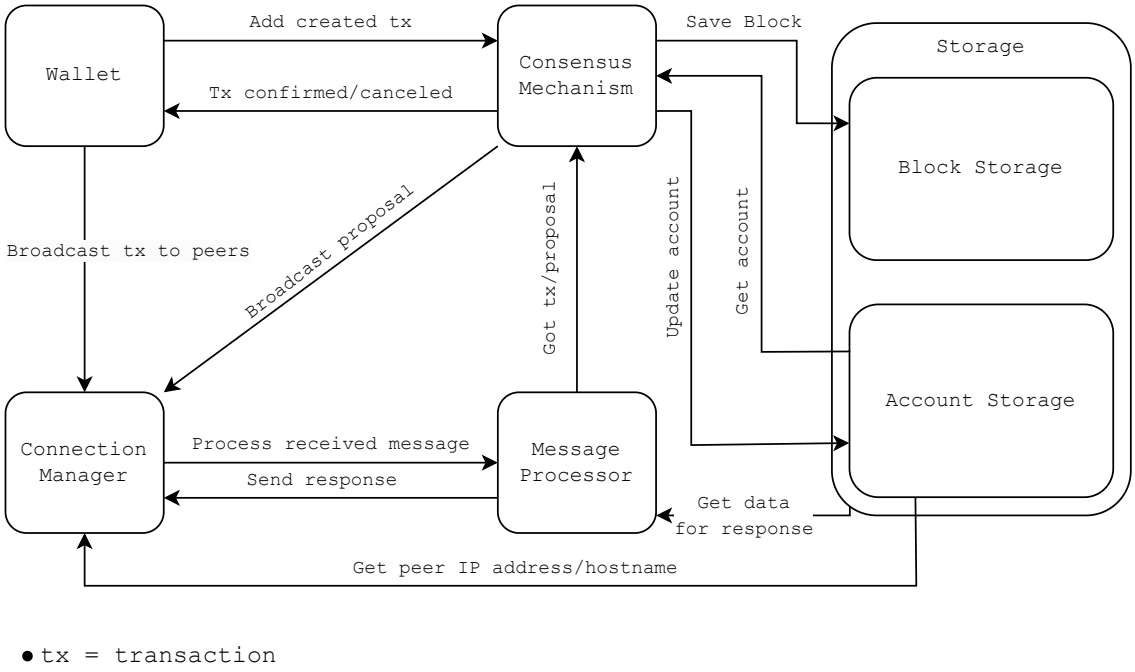

Figure 5.3: Design of blockchain node components

5.6.1 Components

Signer

While this component is not directly presented in Figure 5.3, it plays a crucial role in this application. It is used by all of the components, except the *Storage*, and its primary task involves generating key pairs, signing transactions and proposals, and verifying signatures.

Wallet

The Wallet stores and manages configurations of the local node account. This encompasses the generation and storage of key pairs for digital signatures, creating new transactions, and storing records of transactions associated with the local account. Additionally, it maintains information about the node's UNL, current available balance, and unique sequence number used for creating transactions.

When a transaction is created, it is added to the transaction pool[2] in the consensus mechanism and broadcasted to known peers.

Storage

It is an interface that enables communication with the local database for blocks and accounts. Changes in the database are performed only by the consensus mechanism. Upon validating transactions, it updates the status of individual accounts and appends a new block to the database. However, other components also require read-only access to the database. This includes a Connection Manager for retrieving a node's IP address or hostname and a Message Processor that responds to *GetData* type messages (message types will be discussed later in this section).

Connection Manager

The Connection Manager enables P2P communication between nodes. It is responsible for accepting connections from peers and initializing connections with them. Additionally, the Connection Manager handles scenarios where peers disconnect and manages connections to ensure only one connection with each peer. Another crucial function is sending and broadcasting messages, as well as receiving messages from other peers. Received messages are then delegated to the Message Processor for further processing.

Message Processor

The Message Processor is responsible for taking actions based on received messages. These actions involve informing other components about received information or providing direct responses to these messages.

Consensus mechanism

The role of the Consensus Mechanism is quite clear. It executes the algorithm that ensures nodes on the blockchain reach a consensus on the next block. I have decided to utilize the XRP Consensus Protocol which was described in Section 2.6. As illustrated in Figure 5.3, the Consensus Mechanism interacts with most other components. It adds new blocks and updates accounts in Storage, generates message proposals, and forwards them to the Connection Manager for transmission. Moreover, if the newly created block contains a transaction related to a local account, the Wallet will be notified about this transaction.

[2]Set of transactions waiting for inclusion in the next block.

5.6.2 Data structures

Account

The account structure is illustrated in Figure 5.4. Each account is assigned a unique identifier, generated by hashing the account's public key with SHA-512. Subsequently, the account possesses the following attributes:

- *Public key:* The public key is utilized for verifying signatures associated with an account.
- *Balance:* The current balance of an account.
- *Transaction sequence number:* This denotes the sequence number of the last transaction made. Upon the creation of a new account, this number initializes to 1. Subsequent transactions increment this number by 1. The purpose is to prevent the duplication of the same transaction multiple times. Transactions are validated only if the sequence number is higher than the last recorded one.
- *List of IP addresses/hostnames:* IPv4/IPv6 addresses or hostnames where this node can be reached.

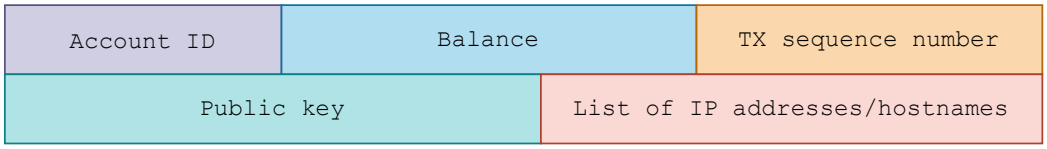

Figure 5.4: Account structure

Transactions

The main objective of transactions is to transfer resources between accounts. Transactions are simplified to always involve one sender and one receiver; there is no option for transactions with multiple receivers.

Each transaction must include a digital signature, the address of a recipient and sender, a timestamp, an amount of transferred funds, and a unique sequence number within a single sender to prevent duplication of transactions. The illustration of transaction structure is in Figure 5.5.

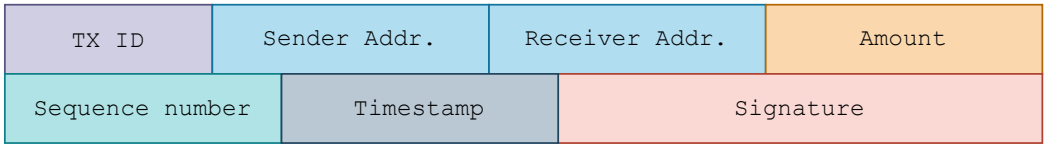

Figure 5.5: Transaction structure

Blocks

Figure 5.6 illustrates the structure of a single blockchain block. The block header incorporates all the essential elements detailed in Section 2.3. Meanwhile, the data section contains validated transactions.

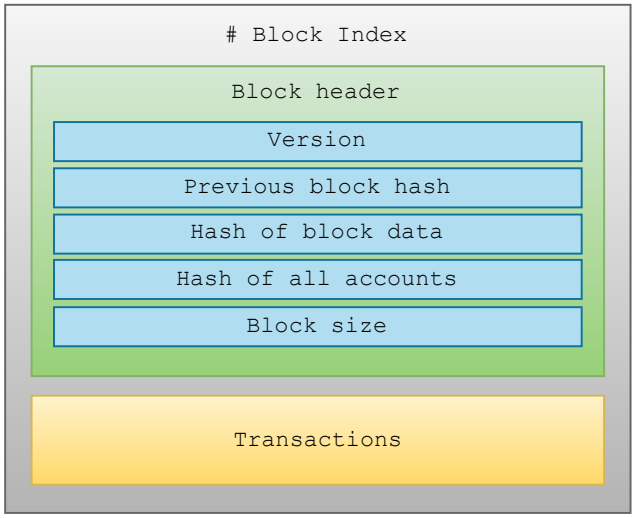

Figure 5.6: Block structure

Proposals

Proposals, illustrated in Figure 5.7, are specific structures within the chosen consensus mechanism, with two main types: *transaction set proposals* and *block proposals*.

A *transaction set proposal* provides information to other validators about the transactions being considered for inclusion by the issuing node. It typically includes the following components: the issuer's identifier, a timestamp, a sequence number of a proposal, unique for each issuer (reset at the start of each consensus round), the identifier of the proposed transaction set (represented by the Merkle root hash of the proposed transaction set), the proposed transaction set itself, and the issuer's signature.

On the other hand, a *block proposal* notifies other validators about the next block being considered by the issuer. It indicates that the issuer has reached a consensus on a specific set of transactions and is proposing the creation of the next block. This type of proposal contains the issuer's identifier, the identifier of the proposed block (represented by the block header's hash), the proposed block header, and the issuer's signature.

Messages

There are several types of messages, some of which were inspired by the messages used by the Bitcoin protocol [8]. Each message consists of a message header, which remains consistent across all message types. This header contains identifiers for the message type,

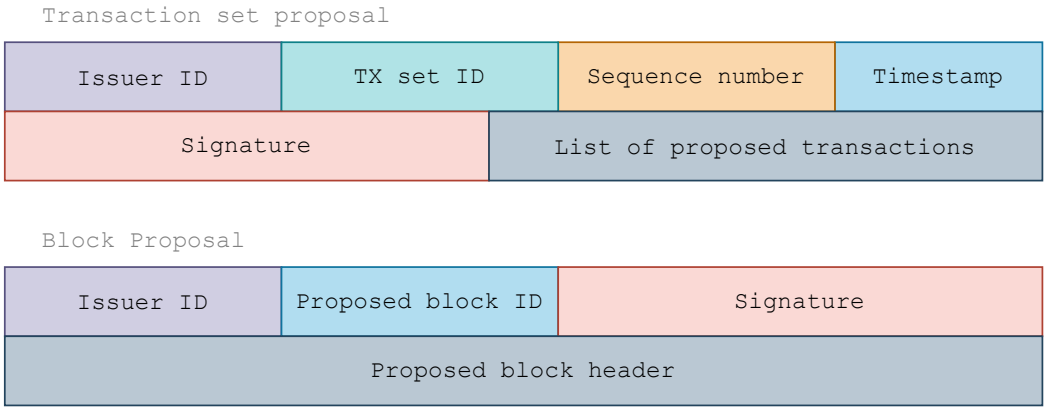

Figure 5.7: Proposals structure

message size, a control checksum, and a magic number, a constant used to ensure proper message parsing.

Types of messages:

- **Version:** The message sent by the node that initializes a connection. It contains a version number, node type identifier, and node identifier. Based on this message, a node can decide whether to accept the connection.
- **Ack:** A positive response to the Version message. If a node does not wish to accept a connection from another node, it simply closes the connection without sending any additional messages.
- **Transaction:** A message containing a transaction.
- **Account:** A message containing an account.
- **Block:** A message containing a block.
- **GetAccount:** A message requesting an inventory message with account identifiers from a particular point in the blockchain.
- **GetBlock:** A message requesting an inventory message with block header identifiers from a particular point in the blockchain. (inspired by the Bitcoin protocol [8])
- **Transaction set proposal message:** A message containing a transaction set proposal.
- **Block proposal message:** A message containing a block proposal.
- **Inventory:** A message containing inventories known to the transmitting peer. Inventories consist of an inventory type and identifier, which can be used to identify a transaction, account, or block. These messages can be sent to announce new objects or in response to GetAccount or GetBlock messages. (inspired by the Bitcoin protocol [8])

- **GetData:** This message requests data from another node. It shares a similar structure to the inventory message. As a response to this message, a Transaction, Account, or Block message with the specified object identifier will be sent. (inspired by the Bitcoin protocol [8])

The Block, Account, GetAccount, and GetBlock messages are listed here for completeness, but will not be used due to the chosen abstraction.

Figure 5.8 illustrates possible scenarios with designed messages. There is no scenario for proposal messages because they are in competence of the consensus mechanism and there is no response for them.

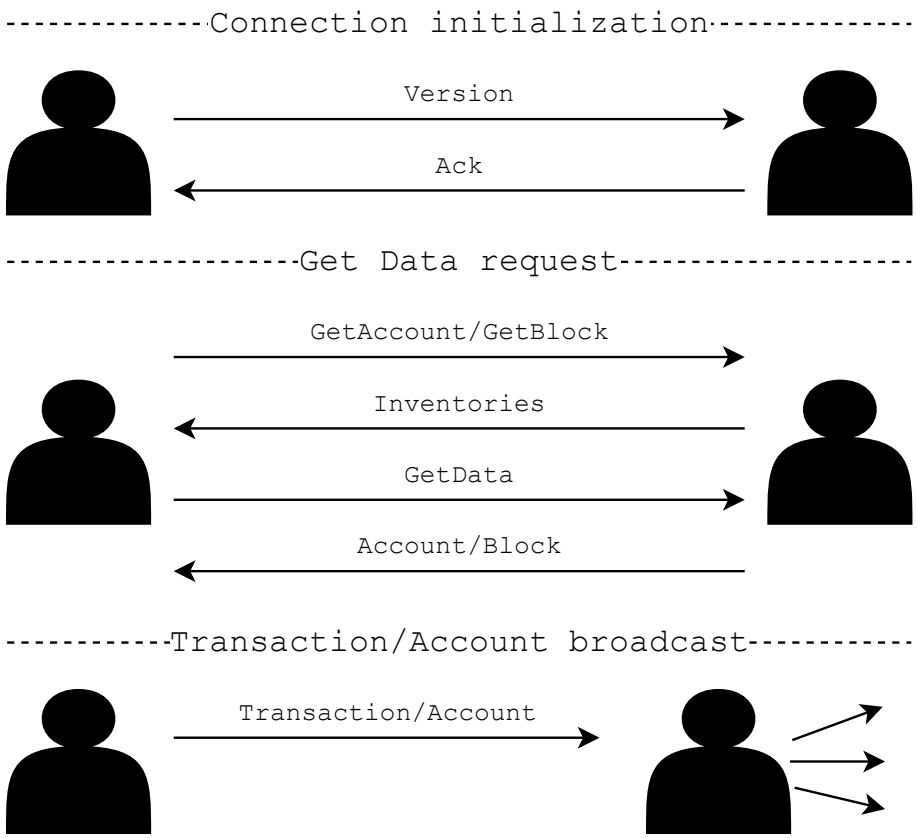

Figure 5.8: Designed message scenarios

Chapter 6

Implementation

This chapter provides a detailed overview of the design implementation outlined in Section 5.6. The development platform selected for this project is Linux Ubuntu 22.04, and the entire implementation is conducted using the C++ programming language.

6.1 Signer

As outlined in the design, various algorithms for digital signatures were utilized. PQ digital signature algorithms were sourced from the `PQClean` library [30, 29], which contains original implementations of algorithms from the NIST competition. I chose implementations only with AVX2 hardware optimization. Currently used algorithms such as Ed25519 and ECDSA were sourced from the `Crypto++` library [13], which also provides the implementation of the SHA-512 algorithm.

The class model of this component is illustrated in Figure 6.1. The Signer is always initialized with the chosen algorithm at the start of the program. Subsequent calls to the Signer then utilize the selected algorithm.

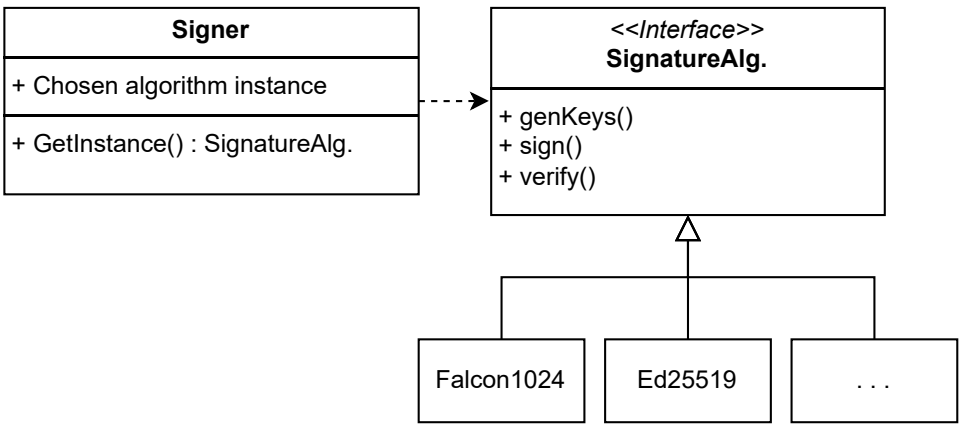

Figure 6.1: Signer class diagram

6.2 Storage

The *Storage* component was implemented with the use of LevelDB[1], which is an embedded key-value database. This database stores keys and values as arbitrary byte arrays. The decision to use LevelDB was based on its simplicity and efficiency in storing all the necessary data. The database's data storage structure is illustrated in Figure 6.2. Blocks are stored under the block identifier, which is the hash of the block header. Account data are stored under the account identifier. For optimization purposes, accounts' IP addresses/hostnames are stored separately from other account data. This is because IP addresses/hostnames are only used when establishing a connection with a node. Other account data are accessed more frequently, so loading IP addresses/hostnames from the database along with other data would be unnecessary.

The substantial implementation primarily consists of methods that enable communication with the database. These methods typically include operations such as *Get*, *Set*/*Update*, etc.

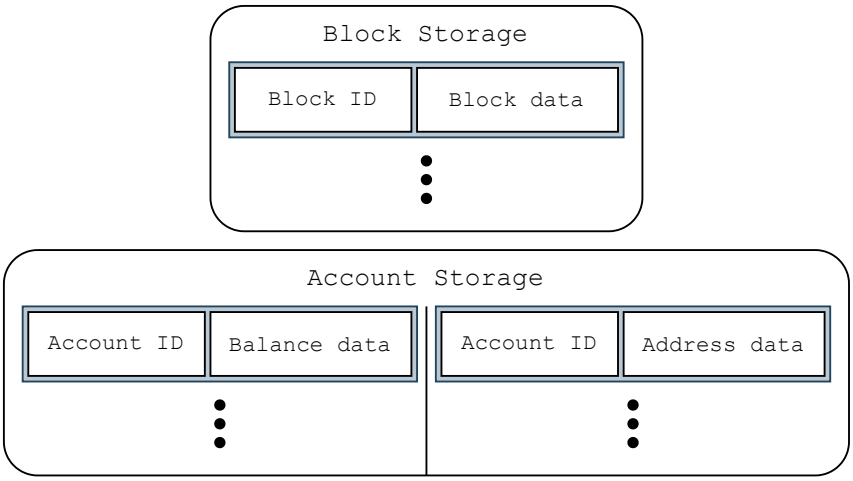

Figure 6.2: Structure of data storage

6.3 Peer-to-peer network

To enable P2P communication, nodes must be capable of creating and accepting connections from other nodes. In essence, within the traditional client-server architecture, a node in a P2P network operates as both a „client“ and a „server“ simultaneously. This functionality is facilitated by the *Connection Manager*, which is implemented in three parts:

„Server:“ Implemented using the BSD socket interface, the server listens on port `8330` for incoming connections from both `IPv4` and `IPv6` addresses. Unlike traditional server implementations, it has no dedicated thread that continuously checks for connection requests. Instead, this functionality is managed by the connection management part.

[1] https://github.com/google/leveldb

„Client:“ Also implemented using BSD sockets, the client can establish TCP connections with other nodes using their `IPv4`, `IPv6` address, or hostname. After establishing a TCP connection, the client sends a *Version* message to the connected node. If the node accepts the connection, it responds with an *Ack* message; otherwise, it closes the TCP connection. In cases where two nodes simultaneously attempt to establish a connection, one connection has to be closed so there will be just one connection between two nodes. To deterministically resolve this issue, the connection initialized by the node with the lexicographically smaller identifier is always closed.

Connection management: This is the largest part of the *Connection Manager* responsible for managing all established connections and handling new connection requests. It was implemented as a separate thread running a loop that performs the following tasks (see also Figure 6.3):

1. Utilizing the `poll()`[2] function, it checks for incoming connection requests and notifies the server to accept these connections.
2. It iterates through sockets of established connections, and using the `poll()` function, it checks for input events. Such events indicate that a new message was received or a connection was terminated. The received messages are also filtered. For example, if a node receives different kinds of messages before the Version message or consensus-related messages from nodes not included in its UNL.
3. It processes requests from other components to send messages by iterating through these requests and sending messages to other nodes.
4. Similarly, it processes requests for creating new connections by iterating through these requests and attempting to establish new connections.

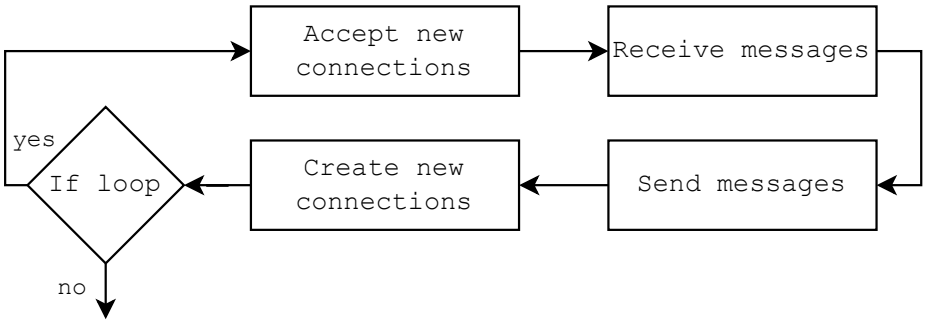

Figure 6.3: Connection Manager tasks

Multiple components can generate requests for sending messages, such as the Wallet for broadcasting newly created transactions and the Consensus Mechanism for broadcasting

[2]The `poll()` function is a system call in Unix operating systems that can monitor I/O events of file descriptors. In the context of this implementation, it is used to monitor the input events of socket descriptors. https://man7.org/linux/man-pages/man2/poll.2.html

proposals, etc. To ensure synchronization, there is a priority queue with exclusive access, where other components can add requests to send messages. In this queue, messages are sorted according to message priority. For example, Transaction messages have higher priority than Inventory messages, as the propagation of new transactions is important for faster progress. These requests are periodically processed within the mentioned loop.

The same mechanism is applied to create new connections, though with a standard queue.

6.4 Message processing

Once the *Connection Manager* receives a message, it delegates it to the *Message Processor*. The *Message Processor* can either process the message immediately or insert it into the processing queue, which is a priority queue where messages are ordered based on priority, similar to the *Connection Manager*. Messages that do not require access to the database, such as *Version* or *Ack* messages, are processed immediately. However, messages requiring more intensive processing or database access are placed into the queue.

The Message processing functionality was implemented as a separate thread that retrieves messages from the queue, processes them, and potentially generates a response to messages, which are then delegated to the *Connection Manager* for sending. The interaction between the *Connection Manager* and the *Message Processor* is illustrated in Figure 6.4.

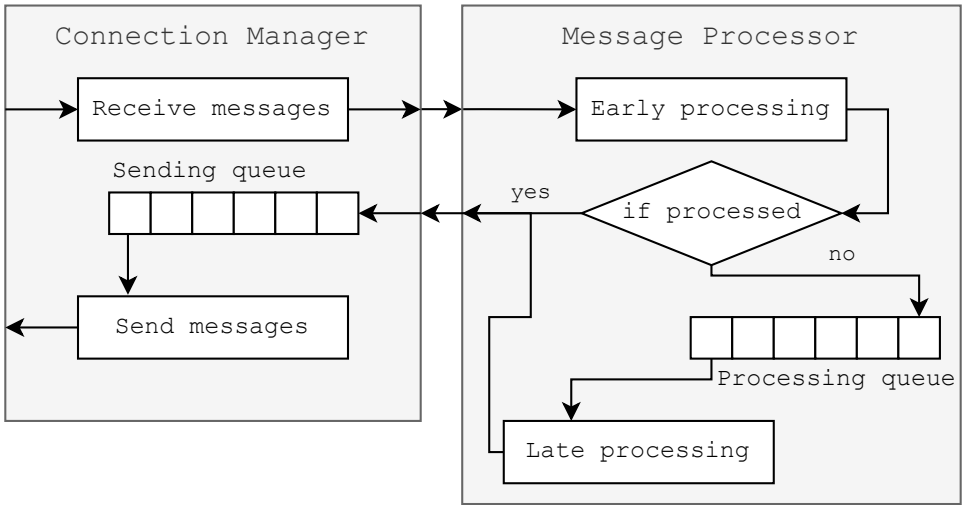

Figure 6.4: Communication of Connection Manager and Message Processor

Different types of messages were discussed in the previous Chapter 5.6.2, they are implemented as designed. However, it is worth mentioning how transactions are propagated between nodes. When a node creates a transaction it directly sends it to its known peers (other nodes). These nodes then propagate only the Inventory message of the transaction, as it is smaller than the actual Transaction message. If some node requires the actual transaction, it can request it with the GetData message.

6.5 Consensus Mechanism

The high-level concepts of utilized consensus mechanism were already described in the Section 2.6. This section will describe more in-depth details about the final functioning and implementation. This implementation was inspired by the original generic implementation of the Ripple (XRP) Consensus Protocol [45] and the article by Amores-Sesar et al. [5], which provides a simplified pseudo-code explanation of the original implementation.

I have decided against copying the entire original implementation because it strongly depends on different code structures and it would be problematic to integrate it into my solution with noted abstraction.

The *Consensus Mechanism* is the last component that functionality was implemented to run in a separate thread. It employs a *transaction pool* structure, which is a set of candidate transactions that may be considered for inclusion in the next block. Transactions in a block and proposals must be in a deterministic order so each node would calculate the same Merkle tree hash of transactions. The process of reaching consensus runs whenever there is a transaction in the transaction pool, and it operates in three phases within a loop:

- **Open:** During this phase, consensus waits for a certain duration for transactions to accumulate in the transaction pool. The duration of this phase depends on the duration of the previous consensus round.
- **Established:** After the time for the Open phase expires, consensus decides to close a block and moves to the established phase. In this phase, validators propose and update their transaction sets.
- **Accepted:** Once there is 80% consensus among a node's chosen validators[3] on a transaction set, the block with this set is considered valid, and the block proposal is broadcasted. However, transactions in this block are executed only if the same block is proposed by 80% of a node's chosen validators. Subsequently, consensus restarts from the beginning.

Figure 6.5 with the description below illustrates a more detailed explanation of consensus steps:

- **Get preferred block:** At the start of each iteration, a node checks if it is working on the same block as its chosen validators. If other validators work on a different block, the node switches to this block and restarts the consensus. The node obtains information about the preferred blocks of other nodes from block proposal messages.
- **Begin consensus:** Restarts the consensus process.
- **Close block:** Transactions in the transaction pool are added to a new block and this transaction set is proposed to other nodes.
- **Create transactions disputes:** A transaction becomes disputed when it is proposed by the node itself and some other node does not propose it, or vice versa. Creating disputed transactions involves comparing the node's transaction set with the transaction sets of other nodes.

[3] Node's chosen validators are nodes on the Unique Node List (UNL). UNL is a list of trusted validators. Each node manages its own UNL.

- **Update proposals:** The node's proposal updates based on disputes. If the majority of other nodes propose a transaction not in the node's transaction set, it may add it there or erase a transaction if the majority of other nodes do not propagate it.
- **Accept block:** When 80% of the node's chosen validators agree on a specific set of transactions, it creates a block proposal for a new block with this transaction set. If 80% of the node's validators agree with this new block, transactions are executed, and the new block is fully validated.

Consensus failures can occur due to insufficient overlap between validators' UNLs or if many validators disconnect from the network. In such cases, the network may stop making further progress. However, delving into the limitations of the utilized consensus mechanism is beyond the scope of this work. The analysis of this consensus can be an extensive topic. For more in-depth insights refer to articles [11, 5] that delve into the analysis of the Ripple consensus protocol.

There may occur a scenario where a node accepts a new block early and the rest of the nodes make some more additional changes. In such cases, the node must synchronize with other nodes and restart the consensus process. However, because the synchronization process is abstracted this scenario will be mitigated by ensuring the initial synchronization and very good overlap of nodes UNLs.

6.6 Wallet

The purpose of the wallet, as described in Section 5.6.1, has been fully implemented with all the mentioned functionality. Configuration data for the wallet are stored in JSON format. For operating with JSON files was utilized the Nlohmann JSON library[4]. While it is not the fastest JSON library available for C++, it is simple and sufficient for this purpose.

6.7 Console Interface

The application is controlled through a command-line interface based on the Model-View-Controller (MVC) design. The model manages and communicates with all components mentioned above. The View enables users to enter commands for the application and displays outputs. The Controller processes the commands entered from the console, checks them, and invokes model functionality.

[4] https://github.com/nlohmann/json

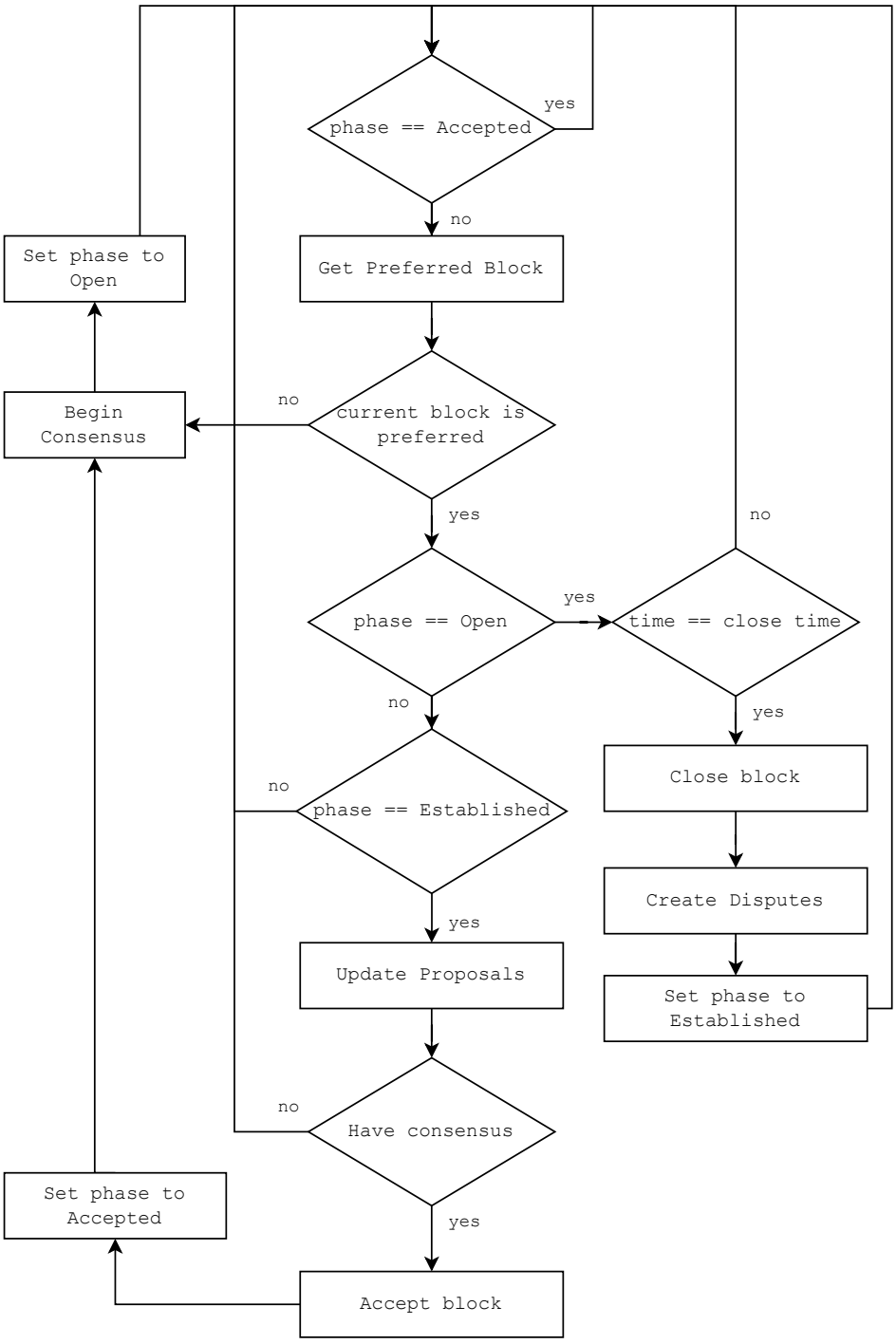

Figure 6.5: Steps of implemented consensus mechanism

Chapter 7

Testing

The testing aimed to evaluate the overall functionality, security parameters, and performance of the implemented blockchain with the utilization of various algorithms for digital signatures. Testing was performed by initializing several Docker[1] containers, with each container running one node instance. As described in the design Chapter 5.3 for testing purposes the blockchain will have a fixed number of nodes with pre-initialized accounts and account databases. The test setup process can be outlined as follows:

1. Generate configuration files for blockchain accounts.
2. Create the Docker image for a node and copy the generated account configuration files.
3. Create a YAML file for Docker Compose.
4. Start the Docker containers.
5. Initialize each node's account database.
6. Run the main application.

All these steps were automated, so it is enough to create one configuration file where each line specifies the settings of one node, including IP address, initial balance, and UNL. For fully automatic testing, there is also a script that sends commands to the application input according to a predefined scenario.

7.1 Functionality testing

Before testing the overall functionality, there are also unit tests implemented using the GoogleTest library[2]. These tests cover all blockchain components, except for the Connection Manager and Consensus Mechanism, as they require communication between multiple nodes.

To test the *Connection Manager* and *Consensus Mechanism*, as well as the overall functionality, was employed the test setup described above. Tests were performed with 3, 5, 10, 15, and 20 nodes. However, checking the test results was no longer automatic but

[1] https://www.docker.com/
[2] https://github.com/google/googletest

manual by inspecting the final state of the blockchain (created blocks and account states), and possibly reviewing logs generated by each node.

The test scenario involved each node creating 20 transactions over 50 seconds. After that, another 30 seconds were left to reach the consensus and new blockchain state. This scenario was repeated multiple times for each configuration of 3, 5, 10, 15, and 20 nodes.

An important consideration was the overlap of nodes' UNLs. With a 100% overlap (meaning each node has all other nodes in its UNL) the blockchain always succeeded in reaching the correct final state. However, if a node's UNL overlap is not sufficient the network might fail to reach a consensus. The lower the overlap in a node's UNL, the greater the likelihood of failure to reach the correct final state. Tests with different overlaps were also performed, but finding sufficient overlap of nodes' UNLs and testing the precise properties of the consensus mechanism was not within the scope of this work. For this work, it is sufficient to use the overlap of nodes's UNLs above 80%. Under these circumstances, the network should always reach a consensus.

7.2 Evaluating security parameters

The security of the blockchain mainly relies on the employed cryptography. However, practical tests on PQ cryptography are not currently possible due to the lack of sufficiently powerful quantum computers. Therefore, the security parameters can be evaluated just theoretically, based on mathematical proofs, scientific papers, and recommendations from relevant security institutions. These aspects were previously discussed in Section 3.2.

7.3 Performance testing

Performance testing was made with PQ algorithms Flacon1024, Falcon512, Dilithium5, Dilithium3, Dilithium2, and currently used algorithms Ed25519 and ECDSA. All tests were done on the operating system Ubuntu 22.04 and processor AMD Ryzen 5 5600.

Measured attributes were the number of processor cycles, memory utilization of the application in the 35th second of the runtime, and the amount of data sent/received during the runtime. Processor cycles were measured using the Linux utility `perf stat`[3], memory utilization data was obtained from Docker stats, and the amount of transferred data was tracked by the application itself. Although measuring memory utilization in this way might seem unconventional, the proposed scenario generated the transactions evenly, and the consensus was also constantly validating new blocks, so the allocated memory was almost at the same level during the entire program runtime. The 35th second was chosen because it is in the middle of the transaction-generating phase. All these measured attributes correspond to the testing design outlined in Section 5.2. These metrics align with other studies, mentioned in that section, that were made on similar topics.

The testing scenario was the same as for functionality testing. The purpose of this scenario is to measure how size and performance requirements grow with the increasing number of nodes and transactions. Tests were performed with 3, 5, 10, 15, and 20 nodes, each creating 20 transactions over 50 seconds with an additional 30 seconds to reach a consensus. After this time, all nodes were stopped and performance statistics were captured. The overlaps of UNLs were always 100% to keep the same conditions for each test round. This scenario was used for all tests, and the only changing aspect was the algorithm utilized

[3] https://perf.wiki.kernel.org/index.php

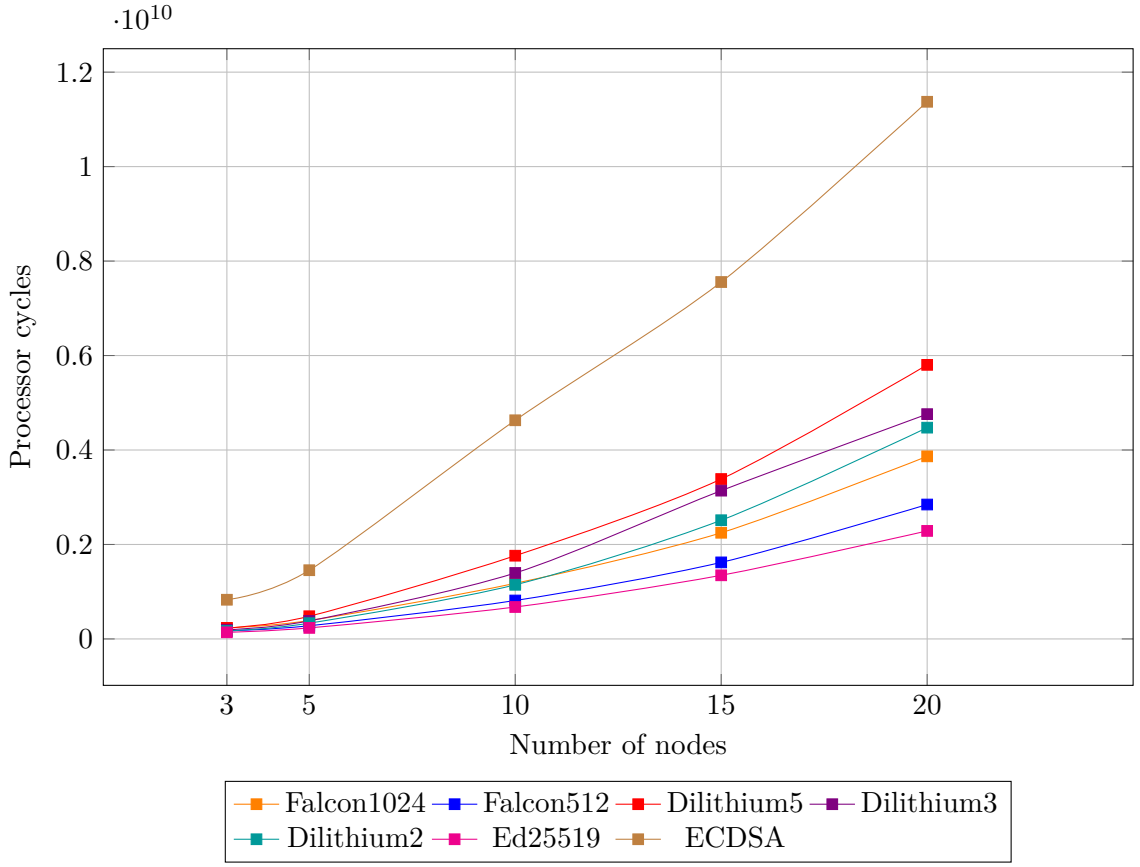

Figure 7.1: Measured complexity of the validator application with different digital signature algorithms.

to make digital signatures. Five tests were performed for each algorithm in each configuration (3, 5, 10, 15, 20 nodes). Subsequently, the results from all nodes across all five rounds were averaged. So, for example, with 10 nodes, there were averaged performance results from 10 nodes, and this process was repeated five times, and finally, all results were averaged.

Complexity

The chart in Figure 7.1 shows the average number of processor cycles performed by the application using different algorithms for digital signatures. Falcon's PQ algorithms are almost at the same level as the currently used Ed25519 algorithm. Dilithium's PQ algorithms are somewhat worse, but their results are not bad for practical use. In the categories of PQ algorithms, Falcon dominates mainly because it has a very fast signature verification process (see Table 3.4). Although the Dilithium algorithms have a much faster signing process, in blockchains the signing is only used when creating transactions and proposals and is performed only by their creator, while each node performs the signature verification of all transactions.

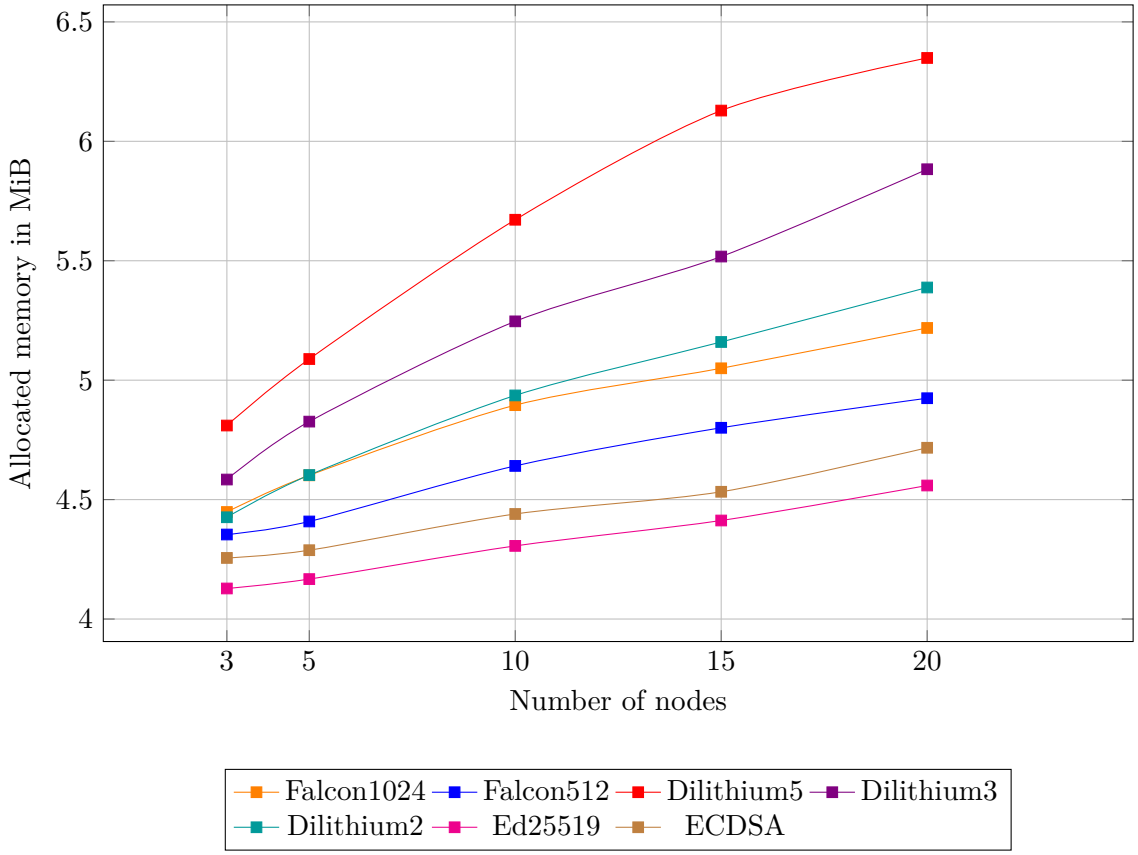

Figure 7.2: Measured memory allocation of the validator application with different digital signature algorithms.

Allocated memory

Chart 7.3 shows the average amounts of allocated memory during tests with different digital signature algorithms. The growth in memory allocation is not too steep across configurations with different node numbers. This is primarily because the utilized consensus mechanism creates a new block every few seconds, and after that, the new block is saved, and allocated data are freed. As a result, transactions do not accumulate and take up less memory.

Amounts of transferred data

The chart in Figure 7.2 shows average amounts of sent/received data during tests with different digital signature algorithms. Currently used algorithms Ed25519 and ECDSA are almost at the same level. The results of PQ algorithms correspond to their security levels; higher security levels entail larger keys and signatures, which results in a higher volume of transferred data. This graph highlights one of the major disadvantages of PQ cryptography for blockchains. The data transfer requirements grow significantly as the number of executed transactions or nodes in the network increases. While the growth is relatively moderate with the currently used algorithms it becomes more influential with PQ algorithms. For example, in the case of Dilithium5, when doubling the number of nodes from 10 to 20, the amount of transferred data is more than tripled. This growth is logical,

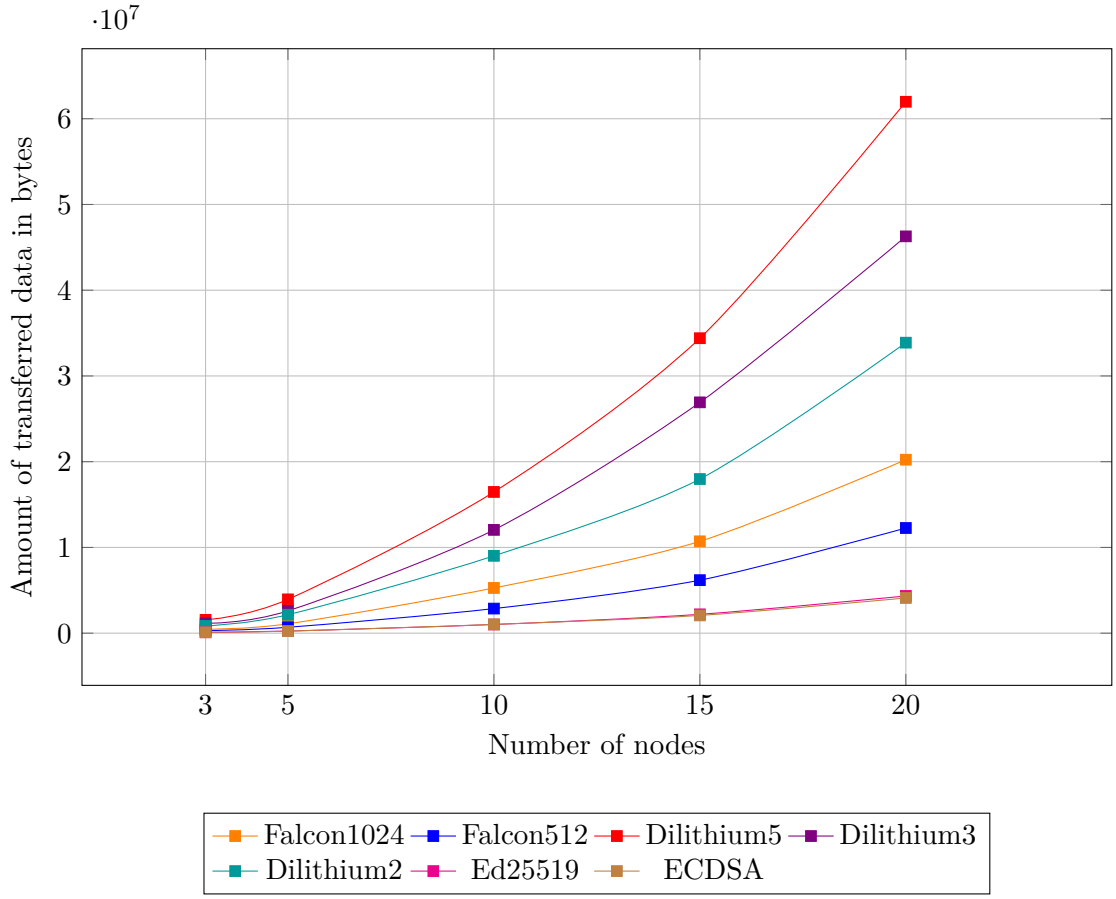

Figure 7.3: Measured amounts of transferred data by the validator application with different digital signature algorithms.

and the results of this observation did not even need to be tested practically, as they can be approximately calculated.

7.4 Performance testing discussion

The important findings from performance testing are:

- The measured complexity of PQ cryptography compared to currently used ECDSA and Ed25519 algorithms is not so terrible. The most significant factor is the effectiveness of a signature verification process. In this metric dominate Falcon algorithms, which are in terms of performance very close to the currently widely used algorithm Ed25519. Dilithium algorithms are somewhat worse, but in my opinion, they are still usable.
- Because of the sizes of PQ keys and signatures, applications utilizing PQ cryptography may necessitate the allocation of additional heap memory. This could potentially impact performance, as extensive heap memory usage might result in increased swapping to disk swap space. For blockchains, the results show that a consensus mechanism that creates new blocks more often with fewer transactions in one block can ensure that the program uses less allocated memory.

- The primary concern for blockchains arises from the sizes of PQ keys and signatures. The results indicate that with PQ digital signature algorithms, individual nodes are required to transfer larger volumes of data compared to the presently employed ECDSA and Ed25519 algorithms. This not only escalates network traffic but also accelerates the overall growth of the blockchain's size. The key factor here is the number of active nodes in the blockchain network and the volume of transactions they generate. Furthermore, this issue may get even more significant if a hybrid combination of classical and PQ cryptography is employed for enhanced security.

As mentioned in Section 5.2 it is not possible to comparatively evaluate this implementation with other blockchain platforms. Nevertheless, it is viable to estimate the potential impact of PQ cryptography on the overall size of other blockchains. For instance, let's consider Bitcoin. The following data was sourced from the *Blockchain.com* project [9]. In contrast to the implemented blockchain, Bitcoin transactions involve both a digital signature and a public key. Bitcoin employs various transaction types, but for simplicity, let's assume each transaction contains only one signature and one key, although this is not true and it may lead to significant inaccuracies. Bitcoin utilizes the ECDSA algorithm for digital signatures, with a key size of 32 bytes and a digital signature size of 72 bytes. Currently, the average block size in Bitcoin is approximately 1,5MB, and it contains an average of 3800 transactions, this results in an average transaction size of about 400 bytes.

Suppose Bitcoin were to adopt the Falcon-512 PQ algorithm, which, according to Table 3.2, has a key size of 876 bytes, and a signature size is 666 bytes. In this case, the size of one transaction would increase to 1 859 bytes, and only 800 transactions would fit into a 1,5MB block.

Considering the largest keys and signatures from the mentioned PQ algorithms, the Dilithium5 algorithm has a key size of 2 592 bytes and a signature size of 4 595 bytes. In this scenario, the size of one transaction would be approximately 7 483 bytes, with only 200 transactions fitting into a 1,5MB block.

From the available data of the Bitcoin.com project, it is also possible to track the growth of the Bitcoin blockchain, which closely follows an exponential curve due to Bitcoin's increasing popularity over time. Presently, the entire Bitcoin blockchain size is around 569GB, with the total number of transactions reaching one billion. This indicates that average transaction size is 570 bytes. If Bitcoin had utilized the Falcon-512 PQ algorithm throughout its existence, its size today would be around 2 terabytes. In the case of the Dilithium5 algorithm, it would be around 7,6 terabytes.

This is just a very simplified illustration. The purpose was to highlight the major problem with the employment of PQ cryptography in the blockchain. Bitcoin was not originally designed to accommodate cryptography with such large keys and signatures. Future blockchain developers face the challenge of adapting the structure and functionality of new blockchain platforms to optimize them for the utilization of PQ cryptography.

Testing summary

The implementation still has several limitations due to the abstraction used, but it was also mainly designed for performance testing of different digital signature algorithms. However, according to the performed tests, the application performs well and meets the defined goals.

Performance testing reveals that a PQ blockchain's primary challenge is likely its size. This is primarily due to the larger PQ signatures and keys compared to current algorithms.

Chapter 8

Conclusion

This thesis aimed to design and implement a blockchain that would be secure against threats that may possess future quantum computers. These declared aims were mainly achieved. The solution describes the complete design and implementation of the PQ blockchain with some degree of abstraction.

The important part of this thesis was also the analysis of blockchain and its vulnerabilities against quantum threats. Most of the threads are related to cryptography. For this reason, this thesis analyzed new PQ algorithms and examined their suitability for use in blockchains. These analyses were then used to design and implement a blockchain resistant to quantum attacks. The implementation employs multiple PQ and currently used algorithms, which were compared in terms of performance to highlight the differences between PQ and current cryptography in blockchains. It would be interesting to compare the results with existing implementations; however, not many blockchains have implemented PQ cryptography yet, and because my implementation uses a certain degree of abstraction, the results might also be inaccurate.

The performance tests showed that PQ cryptography in blockchains is usable. Beneficial are mainly algorithms with fast digital signature verification. However, the main problem is the size of the public keys and digital signatures. The nodes in the blockchain network have to transfer larger amounts of data, and the overall size of the blockchains also grows significantly faster.

Personally, this work gave me a very detailed insight into blockchain technologies and cryptography. I was really interested in decentralization and blockchain systems. It is easy to understand the concepts, but after writing this thesis, I know many more details and insights into practical solutions.

Future works may try to find solutions to the downsides of using PQ cryptography in blockchains. It can be a challenging task. The main problem will be a reduction of blockchain sizes and optimization of data transfers between nodes. It would also be interesting to create stochastic models for testing, bottlenecks searching, or analyzing the use of PQ cryptography for specific blockchain platforms.

Bibliography

[1] Alagic, G., Apon, D., Cooper, D., Dang, Q., Dang, T. et al. *Status report on the third round of the NIST post-quantum cryptography standardization process* [online]. US Department of Commerce: NIST, july 2022 [cit. 2023-11-15]. Available at: https://nvlpubs.nist.gov/nistpubs/ir/2022/NIST.IR.8413-upd1.pdf.

[2] Albrecht, M. R., Bernstein, D. J., Chou, T., Cid, C., Gilcher, J. et al. *Classic McEliece* [online]. 2023-03-04 [cit. 2023-12-21]. Official McEliece website. Available at: https://classic.mceliece.org.

[3] Albrecht, M. R., Bernstein, D. J., Chou, T., Cid, C., Gilcher, J. et al. *Classic McEliece-potential draft ISO standard* [online]. April 2023 [cit. 2023-12-21]. Available at: https://classic.mceliece.org/iso-mceliece-20230419.pdf.

[4] Allende, M., León, D. L., Cerón, S., Pareja, A., Pacheco, E. et al. Quantum-resistance in blockchain networks. *Scientific Reports* [online]. Springer Nature. april 2023, vol. 13, 5664 (2023), p. 23, [cit. 2024-01-15]. DOI: 10.1038/s41598-023-32701-6. Available at: https://rdcu.be/dvY0m.

[5] Amores Sesar, I., Cachin, C. and Micic, J. Security Analysis of Ripple Consensus. *CoRR* [online]. arXiv. december 2020, abs/2011.14816, [cit. 2024-04-23]. DOI: https://doi.org/10.48550/arXiv.2011.14816. Available at: https://arxiv.org/abs/2011.14816.

[6] Bavdekar, R., Chopde, E. J., Agrawal, A., Bhatia, A. and Tiwari, K. Post Quantum Cryptography: A Review of Techniques, Challenges and Standardizations. In: *2023 International Conference on Information Networking (ICOIN)* [online]. Bangkok, Thailand, 2023: IEEE Comput. Soc. Press, February 2023, p. 146–151 [cit. 2023-11-13]. DOI: 10.1109/ICOIN56518.2023.10048976. ISBN 978-1-6654-6268-6. Available at: https://ieeexplore.ieee.org/abstract/document/10048976.

[7] Belotti, M., Božić, N., Pujolle, G. and Secci, S. A Vademecum on Blockchain Technologies: When, Which, and How. *IEEE Communications Surveys & Tutorials*. 2019, vol. 21, no. 4, p. 3796–3838, [cit. 2023-11-04]. DOI: 10.1109/COMST.2019.2928178.

[8] Bitcoin Project. P2P Network. *Bitcoin developer* [online]. [cit. 2024-04-27]. Available at: https://developer.bitcoin.org/reference/p2p_networking.html. Path: Bitcoin;Reference;P2P Network.

[9] Blockchain.com, Inc.. *Blockchain.com* [online]. 2011 [cit. 2024-05-04]. Available at: https://www.blockchain.com/.

[10] Chandel, S., Cao, W., Sun, Z., Yang, J., Zhang, B. et al. A Multi-dimensional Adversary Analysis of RSA and ECC in Blockchain Encryption. In: Arai, K. and Bhatia, R., ed. *Advances in Information and Communication* [online]. Cham: Springer International Publishing, February 2020, vol. 70, p. 988–1003 [cit. 2024-05-04]. DOI: https://doi.org/10.1007/9. ISBN 978-3-030-12385-7. Available at: https://link.springer.com/chapter/10.1007/978-3-030-12385-7_67.

[11] Chase, B. and MacBrough, E. Analysis of the XRP Ledger Consensus Protocol: Ripple Research. *ArXiv preprint arXiv:1802.07242* [online]. arXiv. february 2018, [cit. 2023-12-21]. DOI: 10.48550/arXiv.1802.07242. Available at: https://arxiv.org/abs/1802.07242v1.

[12] Cohen, D., Schwartz, D. and Britto, A. Consensus Structure. *XRP Ledger* [online]. 17. may 2019. 2023-12-05 [cit. 2023-12-21]. Available at: https://xrpl.org/docs/concepts/consensus-protocol/.

[13] Dai, W. *C++ library Crypto++* [online]. 2015. 2023-06 [cit. 2024-01-15]. Available at: https://cryptopp.com/.

[14] Dam, D.-T., Tran, T.-H., Hoang, V.-P., Pham, C.-K. and Hoang, T.-T. A Survey of Post-Quantum Cryptography: Start of a New Race. *Cryptography* [online]. Basel, Switzerland: [b.n.]. august 2023, vol. 7, no. 40, 2023-07-05, [cit. 2023-11-13]. DOI: 10.3390/cryptography7030040. ISSN 2410-387X. Available at: https://www.mdpi.com/2410-387X/7/3/40.

[15] Fan, C., Ghaemi, S., Khazaei, H. and Musilek, P. Performance Evaluation of Blockchain Systems: A Systematic Survey. *IEEE Access* [online]. june 2020, vol. 8, p. 126927–126950, [cit. 2024-05-04]. DOI: 10.1109/ACCESS.2020.3006078. ISSN 2169-3536. Available at: https://ieeexplore.ieee.org/abstract/document/9129732.

[16] Fernández Caramès, T. M. and Fraga Lamas, P. Towards Post-Quantum Blockchain: A Review on Blockchain Cryptography Resistant to Quantum Computing Attacks. *IEEE Access* [online]. IEEE. december 2020, vol. 8, p. 21091–21116, 2022-01-23, [cit. 2023-11-13]. DOI: 10.1109/ACCESS.2020.2968985. ISSN 2169-3536. Available at: https://ieeexplore.ieee.org/abstract/document/8967098.

[17] Ferreira, J., Antunes, M., Zhygulskyy, M. and Frazão, L. Performance of Hash Functions in Blockchain Applied to IoT Devices. In: *2019 14th Iberian Conference on Information Systems and Technologies (CISTI)* [online]. Coimbra, Portugal: [b.n.], June 2019, p. 1–7 [cit. 2024-05-04]. DOI: 10.23919/CISTI.2019.8760885. ISSN 2166-0727. Available at: https://ieeexplore.ieee.org/document/8760885.

[18] Fouque, P.-A., Hoffstein, J., Kirchner, P., Lyubashevsky, V., Pornin, T. et al. *Falcon* [online]. November 2017 [cit. 2023-12-21]. Official Falcon website. Available at: https://falcon-sign.info/.

[19] Gomes, J., Khan, S. and Svetinovic, D. Fortifying the Blockchain: A Systematic Review and Classification of Post-Quantum Consensus Solutions for Enhanced Security and Resilience. *IEEE Access* [online]. july 2023, vol. 11, p. 74088–74100,

[cit. 2023-12-27]. DOI: 10.1109/ACCESS.2023.3296559. ISSN 2169-3536. Available at: https://ieeexplore.ieee.org/abstract/document/10185538.

[20] Gomes, J., Khan, S. and Svetinovic, D. Fortifying the Blockchain: A Systematic Review and Classification of Post-Quantum Consensus Solutions for Enhanced Security and Resilience. *IEEE Access* [online]. IEEE. july 2023, vol. 11, p. 74088–74100, [cit. 2024-02-04]. DOI: 10.1109/ACCESS.2023.3296559. ISSN 2169-3536. Available at: https://ieeexplore.ieee.org/abstract/document/10185538.

[21] Grover, L. K. A fast quantum mechanical algorithm for database search. In: *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing - STOC '96* [online]. Philadelphia, Pennsylvania, USA: Association for Computing Machinery: [b.n.], July 1996, p. 212–219 [cit. 2023-11-13]. DOI: 10.1145/237814.237866. ISBN 978-0-89791-785-8. Available at: https://arxiv.org/abs/quant-ph/9605043.

[22] Homoliak, I., Venugopalan, S., Hum, Q. and Szalachowski, P. A Security Reference Architecture for Blockchains. *CoRR* [online]. University of Technology and Design, Singapore: arXiv. april 2019, abs/1904.06898, [cit. 2024-05-04]. DOI: https://doi.org/10.48550/arXiv.1904.06898. Available at: http://arxiv.org/abs/1904.06898.

[23] Information Technology Laboratory. *Module-Lattice-Based Digital Signature Standard* [online]. Gaithersburg, MD 20899-8900: National Institute of Standards and Technology, august 2023 [cit. 2023-12-21]. Available at: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.204.ipd.pdf.

[24] Information Technology Laboratory. *Module-Lattice-based Key-Encapsulation Mechanism Standard* [online]. Gaithersburg, MD 20899-8900: National Institute of Standards and Technology, august 2023 [cit. 2023-12-21]. Available at: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.203.ipd.pdf.

[25] Information Technology Laboratory. *Stateless Hash-Based Digital Signature Standard* [online]. Gaithersburg, MD 20899-8900: National Institute of Standards and Technology, august 2023 [cit. 2023-12-21]. Available at: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.205.ipd.pdf.

[26] IOHK. *Documentation for the Cardano ecosystem* [online]. 2015 [cit. 2024-05-04]. Available at: https://docs.cardano.org/.

[27] IOTA Foundation. *The complete reference for IOTA* [online]. 2021 [cit. 2024-05-04]. Available at: https://wiki.iota.org/.

[28] Jacak, M. M., Jóźwiak, P., Niemczuk, J. and Jacak, J. E. Quantum generators of random numbers. *Scientific Reports* [online]. Springer Nature. august 2021, vol. 11, no. 16108, [cit. 2024-04-13]. DOI: https://doi.org/10.1038/s41598-021-95388-7. ISSN 2045-2322. Available at: https://www.nature.com/articles/s41598-021-95388-7.

[29] Kannwischer, M. J., Schwabe, P., Stebila, D. and Wiggers, T. Improving Software Quality in Cryptography Standardization Projects. In: *IEEE European Symposium on Security and Privacy, EuroS&P 2022 - Workshops, Genoa, Italy, June 6-10, 2022* [online]. Los Alamitos, CA, USA: IEEE Computer Society, 2022, p. 19–30

[cit. 2024-01-15]. DOI: 10.1109/EuroSPW55150.2022.00010. Available at: https://eprint.iacr.org/2022/337.

[30] Kannwischer, M. J., Schwabe, P., Stebila, D. and Wiggers, T. *PQClean library* [online]. 2022. 2024-01-04 [cit. 2024-01-15]. Available at: https://github.com/PQClean/PQClean.

[31] Lyubashevsky, V. Fiat-Shamir with Aborts: Applications to Lattice and Factoring-Based Signatures. In: Matsui, M., ed. *Advances in Cryptology – ASIACRYPT 2009* [online]. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, p. 598–616 [cit. 2023-12-21]. DOI: 10.1007/978-3-642-10366-7_35. ISBN 978-3-642-10366-7. Available at: https://link.springer.com/chapter/10.1007/978-3-642-10366-7_35.

[32] Marchenkova, A. How Far Away Is The Quantum Threat? *BTG* [online]. 29. august 2023 [cit. 2024-05-04]. Available at: https://btq.com/blog/how-far-away-is-the-quantum-threat.

[33] Melchor, C. A., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O. et al. *HQC (Hamming Quasi-Cyclic)* [online]. [cit. 2023-12-21]. Official HQC website. Available at: https://pqc-hqc.org/.

[34] Merkle, R. C. Digital signature system and method based on a conventional encryption function. november 1989, [cit. 2023-11-05]. U.S. Patent 4 881 264.

[35] Nakamoto, S. *Bitcoin: A Peer-to-Peer Electronic Cash System* [online]. October 2008 [cit. 2023-11-04]. Available at: http://www.bitcoin.org/bitcoin.pdf.

[36] National Institute of Standards and Technology (NIST). *Stateful Hash-Based Signatures* [online]. December 2018. 2023-01-27 [cit. 2023-11-17]. Available at: https://csrc.nist.gov/Projects/stateful-hash-based-signatures.

[37] National Security Agency (NSA). *Announcing the Commercial National Security Algorithm Suite 2.0* [online]. 1.0. Fort Meade, MD: NSA, september 2022, 2022-09 [cit. 2023-11-15]. Available at: https://media.defense.gov/2022/Sep/07/2003071834/-1/-1/0/CSA_CNSA_2.0_ALGORITHMS_.PDF.

[38] National Security Agency (NSA). *The Commercial National Security Algorithm Suite 2.0 and Quantum Computing FAQ* [online]. 1.0. Fort Meade, MD: NSA, september 2022, 2022-09 [cit. 2023-11-15]. Available at: https://media.defense.gov/2022/Sep/07/2003071836/-1/-1/0/CSI_CNSA_2.0_FAQ_.PDF.

[39] Nosouhi, M. R., Shah, S. W. A., Pan, L. and Doss, R. Bit Flipping Key Encapsulation for the Post-Quantum Era. *IEEE Access* [online]. version 9.0. Waurn Ponds, Australia: IEEE. june 2023, vol. 11, no. 23279405, p. 56181–56195, 2023-05-04, [cit. 2023-11-16]. DOI: 10.1109/ACCESS.2023.3282928. ISSN 2169-3536. Available at: https://ieeexplore.ieee.org/abstract/document/10143624.

[40] Národní úřad pro kybernetickou a informační bezpečnost (NÚKIB). *Příloha k dokumentu: Minimální požadavky na kryptografické algoritmy* [online]. 3.0. Brno: NÚKIB, july 2023 [cit. 2023-11-15]. Available at: https://www.nukib.cz/download/uredni_deska/Minimalni%20pozadavky%20na%20kryptograficke%20algoritmy.pdf.

[41] Národní úřad pro kybernetickou a informační bezpečnost (NÚKIB). *Příloha k dokumentu: Minimální požadavky na kryptografické algoritmy* [online]. 1.0. Brno: NÚKIB, july 2023 [cit. 2023-11-15]. Available at: https://www.nukib.cz/download/uredni_deska/Priloha%20-%20Minimalni%20pozadavky%20na%20kryptograficke%20algoritmy.pdf.

[42] Národní úřad pro kybernetickou a informační bezpečnost (NÚKIB). *Útoky s využitím kvantového počítače mohou prolomit současné šifrování: řešením je včasná a efektivní implementace nových standardů* [online]. 5151/2023-NÚKIB-E/630th ed. Brno: NÚKIB, july 2023 [cit. 2023-11-13]. Available at: https://www.nukib.cz/download/publikace/podpurne_materialy/Utoky%20s%20vyuzitim%20kvantoveho%20pocitace%20mohou%20prolomit%20sousasne%20sifrovani%20resenim%20je%20vcasna%20a%20efektivni%20implementace%20novych%20standardu.pdf.

[43] Rajasekaran, A. S., Azees, M. and Al Turjman, F. A comprehensive survey on blockchain technology. *Sustainable Energy Technologies and Assessments* [online]. 2022, vol. 52, p. 102039, [cit. 2023-11-04]. DOI: https://doi.org/10.1016/j.seta.2022.102039. ISSN 2213-1388. Available at: https://www.sciencedirect.com/science/article/pii/S2213138822000911.

[44] Regev, O. On Lattices, Learning with Errors, Random Linear Codes, and Cryptography. *J. ACM* [online]. New York, NY, USA: Association for Computing Machinery. september 2009, vol. 56, no. 6, p. 40, [cit. 2023-12-21]. DOI: 10.1145/1568318.1568324. ISSN 0004-5411. Available at: https://doi.org/10.1145/1568318.1568324.

[45] Ripple Labs Inc.. *Ripple generic consensus algorithm* [online]. 2012. 2023-11 [cit. 2024-01-15]. Available at: https://github.com/XRPLF/rippled/tree/develop/src/ripple/consensus.

[46] Schwabe, P. Kyber. *CRYSTALS-Cryptographic Suite for Algebraic Lattices* [online]. 2017. 2020-12-23 [cit. 2023-12-21]. Official Kyber website. Available at: https://pq-crystals.org/kyber/index.shtml.

[47] Schwabe, P. Dilithium. *CRYSTALS-Cryptographic Suite for Algebraic Lattices* [online]. 2017. 2021-02-16 [cit. 2023-12-21]. Official Dilithium website. Available at: https://pq-crystals.org/dilithium/index.shtml.

[48] Schwabe, P. Stateless hash-based signatures. *SPHINCS*$^+$ [online]. 2017. 2023-08-02 [cit. 2023-12-21]. Official SPHINCS$^+$ website. Available at: https://sphincs.org/.

[49] Shor, P. W. Algorithms for quantum computation: Discrete logarithms and factoring. In: *Proceedings 35th Annual Symposium on Foundations of Computer Science* [online]. Santa Fe, NM, USA: IEEE Comput. Soc. Press, August 1994, p. 124–134 [cit. 2023-11-13]. DOI: 10.1109/SFCS.1994.365700. ISBN 0-8186-6580-7. Available at: https://ieeexplore.ieee.org/abstract/document/365700.

[50] Stackpole, B. Quantum computing: What leaders need to know now. *MIT Sloan School of Management* [online]. 11. january 2024 [cit. 2024-05-04]. Available at: https://mitsloan.mit.edu/ideas-made-to-matter/quantum-computing-what-leaders-need-to-know-now.

[51] Stebila, D., Schwabe, P. and Wiggers, T. *KEMTLS* [online]. 2022 [cit. 2024-01-15]. Available at: https://kemtls.org/.

[52] The Open Quantum Safe Project. OQS algorithm performance visualizations. *Open Quantum Safe* [online]. 2017. 2023-12-19 [cit. 2023-12-21]. Available at: https://openquantumsafe.org/benchmarking/.

[53] XRP Ledger. *XRP Ledger Developer Resources* [online]. 2024 [cit. 2024-05-04]. Available at: https://xrpl.org/docs/.

[54] Yaga, D., Mell, P., Roby, N. and Scarfone, K. Blockchain Technology Overview. *CoRR* [online]. 2019, abs/1906.11078, [cit. 2023-11-05]. Available at: http://arxiv.org/abs/1906.11078.

[55] Yang, Z., Alfauri, H., Farkiani, B., Jain, R., Pietro, R. D. et al. A Survey and Comparison of Post-Quantum and Quantum Blockchains. *IEEE Communications Surveys & Tutorials* [online]. IEEE. october 2023, p. 1–1, [cit. 2023-11-13]. DOI: 10.1109/COMST.2023.3325761. ISSN 1553-877X. Available at: https://ieeexplore.ieee.org/abstract/document/10288193.