Designing a Decentralized Quantum-Resistant Notary Scheme for Blockchain Interoperability with a Dynamic Trust Model

by

Koosha E.Khorasani

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
June 2025

Thesis advisor

Sara Rouhani

Author

Koosha E.Khorasani

Designing a Decentralized Quantum-Resistant Notary Scheme for Blockchain Interoperability with a Dynamic Trust Model

Abstract

Blockchain interoperability is essential for enabling advanced functionalities in decentralized applications, particularly for secure and reliable interactions across disparate blockchain networks. The Notary scheme is a common approach to achieving interoperability; however, it faces significant challenges: vulnerability to quantum computing threats and the complexities of trust management in dynamic, adversarial environments. We investigate a decentralized, quantum-resistant interoperability framework designed to address these challenges through a novel protocol and trust model. The protocol integrates post-quantum cryptographic techniques, including ML-KEM for key distribution, ML-DSA for digital signatures, and AES-256 for message encryption, ensuring security and resilience against emerging quantum threats. Complementing this, we introduce a dynamic trust model that leverages Popoviciu's inequality to manage and update reputation scores in real time, enhancing resilience against trust degradation and enabling recovery from malicious behavior and network fluctuations. Experimental evaluation shows that our proposed protocol achieves quantum resistance while maintaining performance comparable to traditional crypto-

graphic approaches. Moreover, the novel trust model provides accurate and dynamic trust computation under varying network conditions, contributing to more robust and trustworthy interoperability.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Our understanding of decentralized systems has been revolutionized by blockchain technology, beginning with its first major implementation in Bitcoin [Nakamoto, 2008], which itself is based on previous work such as Karma [Vishnumurthy et al., 2003]. The fundamental attributes of blockchain—transparency, immutability, security, availability, and traceability—have been key drivers in the development of a wide range of protocols, networks, and platforms. Some protocols, such as Ethereum [Buterin et al., 2013], are designed to support decentralized applications and smart contracts, while others, such as Hyperledger Fabric [Androulaki et al., 2018], cater to enterprise applications in permissioned environments. Furthermore, certain protocols focus on user privacy and anonymous transactions, as seen in Monero [Alonso et al., 2020], among others. These examples represent only a subset of the numerous blockchain protocols, each tailored to specific use cases and technological priorities. The rapid proliferation of blockchain networks, each with its own protocol, consensus mechanism, and tokenization system, has resulted in fragmented and isolated

ecosystems. These silos lack native support for cross-chain transactions and secure communication mechanisms, creating fundamental challenges for achieving interoperability among blockchains.

This limitation has driven the development of blockchain interoperability solutions. According to the IEEE definition, interoperability among systems refers to 'the ability of two or more systems or components to exchange information and use the information that has been exchanged' [Geraci, 1991]. In the context of blockchain, interoperability can be understood as the ability to share data between multiple blockchains and execute cross-chain operations based on this shared data [Wang et al., 2023]. Achieving blockchain interoperability enables seamless access to resources across networks, fostering collaboration, data exchange, and multi-chain application development. These advances improve efficiency, scalability, and adaptability within decentralized ecosystems [Harris, 2023].

Several interoperability approaches have been proposed, including Blockchain of Blockchains, Sidechains, Trusted Relays, Hashed Time-Lock Contracts, and Notary Schemes [Ren et al., 2023; Belchior et al., 2021]. Although these interoperability solutions address the challenge of connecting heterogeneous blockchains and facilitate cross-chain connectivity, they also introduce new security risks and vulnerabilities that may result in significant financial losses [Augusto et al., 2024]. Therefore, ensuring security and integrity in cross-chain interactions is a fundamental requirement for effective interoperability solutions.

Among the various security issues in blockchain interoperability, this research focuses on two primary concerns. The first is confidentiality, traditionally achieved

through cryptographic methods such as RSA [Rivest et al., 1978] and elliptic curve cryptography [Koblitz, 1987]. While these algorithms are effective against classical computational threats, they are vulnerable to quantum attacks, as demonstrated by Shor [Shor, 1999] and Grover [Grover, 1996]. Although the quantum threat has been acknowledged in blockchain research [Holcomb et al., 2021; Sharma and Mishra, 2021], few interoperability solutions have attempted to address it [Yi, 2023; Wang et al., 2022]. To date, no blockchain interoperability protocol has leveraged the newly finalized post-quantum cryptographic standards, such as Modular Lattice Based Key Encaptulation Mechanism (ML-KEM) [FIPS, 2024a] for key exchange and Modular Lattice Based Digital Signature Algorithm (ML-DSA) [FIPS, 2024b] for digital signatures. These standards are designed to withstand quantum attacks and enhance long-term security in decentralized systems. By integrating these advanced cryptographic primitives into a blockchain interoperability protocol, this work provides a technically robust and practical solution that strengthens cross-chain security against quantum threats, addressing both immediate vulnerabilities and future-proofing decentralized ecosystems.

The second major challenge is trust management, particularly in notary schemes, where notary nodes are responsible for validating cross-chain interactions. These nodes are critical to the security of interoperability solutions, yet existing trust mechanisms often lack the flexibility to dynamically adjust trust in response to malicious activities or evolving network conditions. This limitation arises because the weights assigned to trust factors are typically fixed and non-adaptive [Cao et al., 2023; Chen et al., 2024]. Some solutions, designed for blockchain-based trust management, at-

tempt to mitigate malicious actions by adjusting trust scores. However, these adjustments are often static, leading to either abrupt or gradual trust fluctuations [Putra et al., 2023] or relying on a single static adjustment pattern [Putra et al., 2021]. A dynamic approach proposed by Hu et al. [Hu et al., 2024] introduces penalty factors and a Dynamic Evaluation Window Mechanism. However, this method lacks parameters for rebuilding adaptive trust after malicious actions. Additionally, its reliance on multiple thresholds across five stages complicates implementation, reduces adaptability, and limits real-time responsiveness to network changes.

To address these challenges, we design a quantum-resistant notary scheme for blockchain interoperability based on lattice-based cryptography [Lyubashevsky, 2024]. Furthermore, we develop a novel trust management mechanism that dynamically handles trust degradation, trust rebuilding, and responses to malicious behavior among notary nodes.

1.1 Research Questions

This thesis revolves around answering these key research questions:

1. How does the adoption of post-quantum cryptography affect the processing time—used as the performance metric in this research—and the security of blockchain interoperability compared to conventional solutions such as elliptic curve cryptography (ECC)?

2. How can a dynamic trust model for notary nodes be developed to adaptively update trust levels in real time in response to detected malicious behavior within

the system?

3. What are the advantages and limitations of a dynamic trust model compared to the conventional EigenTrust model in the context of blockchain interoperability?

1.2 Methodology and Approach

To investigate answers to our questions, this thesis designs and develops a notary scheme-based interoperability solution to connect heterogeneous blockchain networks. The proposed solution integrates key aspects of an interoperability framework, including decentralization, security, and consensus mechanisms.

To address the first question, a cryptographic infrastructure is developed to support both post-quantum cryptography and elliptic curve cryptography for key exchange and digital signatures. This allows for a direct comparison of their performance and security implications within the interoperability framework.

To answer our second and third research questions, a novel dynamic trust model as well as an EigenTrust-based algorithm [Putra et al., 2023] are implemented and applied within the proposed notary scheme. A comparative analysis is conducted to evaluate their effectiveness, focusing on the detection latency of malicious behavior, trust degradation rate, and the mechanisms for trust recovery. This analysis offers insights into the strengths and limitations of each approach in the context of blockchain interoperability.

1.3 Contributions

To address the above research questions, we make the following key contributions:

- **Designing, Implementing, and Evaluating a Decentralized Quantum-Resistant Notary Scheme Interoperability Protocol**: This protocol, tailored for a permissioned interoperability network, ensures decentralization by distributing validation responsibilities among multiple independent participants from distinct networks, eliminating reliance on a central authority. The protocol integrates recently standardized post-quantum cryptographic techniques, including ML-KEM for key distribution and ML-DSA for digital signatures, addressing the vulnerability of classical cryptographic methods to quantum attacks. Additionally, it employs AES-256 for message encryption, leveraging its recognized resistance to quantum threats [Rao et al., 2017]. This design enhances the security and longevity of blockchain interoperability protocols against emerging cryptographic threats while maintaining the decentralized nature of the interoperability process.

- **A Dynamic Trust Model for Notary-Based Blockchain Interoperability**: This thesis introduces a novel adaptive trust management framework that dynamically adjusts reputation scores based on the variance in malicious and normal behaviors of notary nodes. The model leverages Popoviciu's inequality [Egozcue and García, 2018] to enable adaptive trust degradation and recovery, allowing for real-time adjustments to network conditions and adversarial behaviors. Unlike conventional models with static trust thresholds, this approach

improves the responsiveness and resilience of trust mechanisms in blockchain interoperability.

Chapter 2

Related Works

2.1 Blockchain Interoperability

Given the numerous approaches to achieving blockchain interoperability, along with varied designs, architectures, and use cases, a substantial body of literature aims to classify blockchain interoperability solutions. One approach structures various layers of blockchain interoperability using principles similar to those of the OSI model [Lohachab et al., 2021], while also examining different perspectives essential for designing interoperability solutions, including technical, system, and operational viewpoints. Another categorization method classifies interoperability solutions based on their stability [Zuo et al., 2021], identifying mature solutions—such as the notary scheme, distributed notary scheme, sidechains, and hashed time-lock—as more stable and established, and solutions with newer features but potentially lower stability as innovative, including Deterministic Cross-Blockchain Token Transfers [Borkowski et al., 2019], Blockchain Router [Wang et al., 2017], Satellite Chain [Li et al., 2017],

and HyperService [Liu et al., 2019].

Blockchain interoperability solutions are often categorized by their architectural approach into three broad groups: public connectors, which include sidechains, relays, notary schemes, and hashlock contracts; blockchain of blockchains; and hybrid connectors, which represent solutions that combine elements of multiple models or fall outside the first two categories[Belchior et al., 2021]. Another framework identifies several distinct mechanisms: notary scheme solutions, sidechain/relay solutions, smart contract solutions that leverage smart contracts, bridging solutions that establish non-blockchain bridges between different ledgers, and blockchain router solutions, where a blockchain functions as a router to facilitate interactions among multiple blockchains [Monika and Bhatia, 2020].

Another existing categorization of blockchain interoperability solutions groups them into sidechains, notary schemes, hashed time-lock contracts (HTLCs), relays, and blockchain-agnostic protocols [Ren et al., 2023], and analyzes them based on system characteristics and security attributes. System characteristics include decentralization, locking, verification, and communication, while security aspects cover liveness, trust, safety, and atomicity.

A different approach focuses on the characteristics and properties of interoperability solutions, introducing key properties such as administration, flexibility, performance, security, and networking, with networking identified as the most influential among them [Kannengießer et al., 2020]. Another classification groups interoperability solutions into manual asset ex-chang—referring to cross-chain transactions conducted through off-chain or in-person solutions such as negotiations—, notary

schemes, sidechains (relays), and hybrid solutions.

Interoperability solutions have also been examined based on the role of smart contracts, dividing them into three categories: heterogeneous blockchains and homogeneous smart contracts, homogeneous blockchains and homogeneous smart contracts, and heterogeneous blockchains and heterogeneous smart contracts [Khan et al., 2021]. Additionally, classification based on four characteristics—privacy and security, scalability, degree of confidence, and bidirectional transactions—has been proposed.

Lastly, some surveys focus specifically on the security dimensions of interoperability [Haugum et al., 2022; Zhang et al., 2023; Lee et al., 2023]. One of the most prominent security surveys provides a comprehensive review of the security aspects of blockchain interoperability [Augusto et al., 2023]. This study discusses different security layers such as the operational, implementation, protocol, and network layers, along with privacy and security properties characterizing a secure cross-chain system. Security approaches in interoperability solutions include trusted third parties, distributed trust, native state validation, secret-based and time-based locks, and privacy-preserving methods such as zero-knowledge proofs, blind signatures, ring signatures, adaptor signatures, homomorphic encryption, and trusted execution environments. Furthermore, vulnerabilities and mitigation strategies have been explored to enhance the security of blockchain interoperability.

2.2 Interoperability Solutions

Drawing on the classifications above, we group interoperability solutions into five categories: sidechain, blockchain of blockchains, trusted relay, hashed time-lock con-

tract, and notary schemes. The following sections provide a detailed overview of each category.

2.2.1 Sidechain

A sidechain is defined as "a blockchain that validates data from other blockchains" [Back et al., 2014]. This approach typically involves a *mainchain* and a *sidechain*. The mainchain is responsible for maintaining a ledger of assets and is connected to the sidechain, which is a separate system linked to the main chain through a cross-chain communication protocol. These two blockchains can achieve interoperability utilizing solutions such as the *two-way peg* [Singh et al., 2020].

Loom network [Loom Team, 2022] is an interoperability solution which is based on sidechain architecture. Loom has a blockchain network named *Basechain* on which developers can publish their Decentralized Application (DApps). Basechain can connect DApps to different blockchain such as Ethereum, Binance Chain, and Tron. It is based on a *federated two-way peg* scheme to swap assets.

Another prominent sidechain platform, RooStock (RSK) [Lerner, 2019], integrates the security of Bitcoin's Proof-of-Work with the capabilities of Ethereum's smart contracts. In the RSK framework, Smart Bitcoin (RBTC) maintains a two-way peg with Bitcoin (BTC). When a transaction occurs, BTC is locked within the Bitcoin network, and an equivalent amount of RBTC becomes accessible within RSK. To convert RBTC back into BTC, the RBTC is locked on RSK, triggering the release of an equal amount of BTC on the Bitcoin network.

Liquid [Nick et al., 2020], functioning as a sidechain solution, enables the transfer

of bitcoins into *Liquid Bitcoin (LBTC)* within the Liquid Network and supports conversion back to BTC through a cryptographic peg. This mechanism includes *peg-in* and *peg-out* operations, which handle conversions between LBTC and BTC. The network also comprises *Blocksigner*—consortium members responsible for validating blocks—and *Watchmen*, who validate Bitcoin transactions during peg-out operations. Blocksigners and Watchmen are collectively referred to as *Functionaries*, and they collaboratively manage the federated peg validation process, ensuring transaction integrity throughout peg-in and peg-out operations.

2.2.2 Blockchain of Blockchains

Blockchain of Blockchains is an interoperability solution wherein a blockchain is designated to store and track transactions generated by DApps distributed across multiple blockchains [Belchior et al., 2021].

In a blockchain ecosystem, two primary components govern its operation: the state of the network, managed through a state machine, and security, ensured through consensus mechanisms. In Polkadot [Wood, 2016], the authors delineate these components by introducing a relay chain, responsible for security and consensus, and multiple *parachains*, each managing its own state. To append a block to a parachain, a group known as *collators* retrieves the block and forwards it to the *validators* designated for that specific parachain. The validators verify the information against the relay chain, and upon validation, integrate it into the relay chain. Through a format known as *XCM* (Cross-Chain Message), parachains can communicate with each other via the relay chain, thereby fostering interoperability among them. More-

over, to broaden interoperability further, bridges are employed to connect Polkadot parachains to external sovereign blockchains.

Axelar [Team, 2021] is a blockchain designed to link various chains together. Two key concepts on which Axelar operates are the *Cross-Chain Gateway Protocol (CGP)* and the *Cross-Chain Transfer Protocol (CTP)*. The CGP facilitates cross-chain routing and delivery across independent blockchains, and the CTP enables applications to conduct simple queries for cross-chain operations. Axelar is composed of several essential components, one of which is the Axelar Blockchain, a proof-of-stake network supported by a group of permissionless validators who consistently produce blocks containing transactions. Another critical component is the *Gateway*, which acts as the access point for cross-chain messages sent by DApp users. Every blockchain connected to Axelar has its own gateway deployed, allowing it to accept messages from DApps and transmit them to the Axelar network for further routing to other connected chains. The final component is the *Validators*, who play a crucial role in verifying submitted events.

Another notable implementation of the Blockchain of Blockchains concept is the solution proposed by Ghaemi et al. [Ghaemi et al., 2021]. In their approach, they employ a *publish/subscribe* architecture to achieve interoperability. This solution comprises three main components: the *Publisher Blockchain*, which serves as the source of data; the *Subscriber Blockchain*, which acts as the receiver of the data; and the *Broker Blockchain*, which acts as the intermediary connecting publishers to subscribers.

2.2.3 Trusted Relays

Trusted Relays function as essential bridges that connect blockchains directly, without the need for third-party intermediaries. In this scheme, to enable Blockchain A to verify data on Blockchain B, Blockchain A incorporates a smart contract. This smart contract retrieves the block header of a specific block on Blockchain B and verifies it using the procedure dictated by Blockchain A's consensus algorithm [Buterin, 2016].

Weaver [1] is an interoperability solution centered around relays, with an architectural structure similar to that described in [Abebe et al., 2019]. In Weaver, the connection between two distinct blockchains relies on two fundamental components. The first is the *Relay*, which acts as an intermediary facilitating protocol communication between networks. Each *Relay* consists of three key elements: the *relay service*, a gRPC server that receives and manages incoming requests from other *Relays*; the *app service*, another gRPC server that handles requests from applications seeking assets from remote networks; and the *driver*, responsible for managing all communication between the *Relay* and its respective network. The second component is the *IOP Module*, which encapsulates logic for membership management, verification policies, and access control rules.

Testimonium [Frauenthaler et al., 2020] is another platform employing the Trusted Relay model. Designed to reduce the cost of Simplified Payment Verification (SPV) [Nakamoto, 2008], it uses an on-demand verification strategy. The system consists of an *on-chain relay* and two off-chain components: *Submitters*, who transmit block

[1] `https://github.com/hyperledger-labs/weaver-dlt-interoperability`

headers from the source to the destination blockchain, and *Disputers*, who detect and contest any invalid or fraudulent block headers.

Automated Gateways [Khorasani et al., 2024] utilizes a modular, smart contract-driven architecture to enable interoperability across blockchains. In this design, relays act as gateways that connect individual blockchains within the network. Access control is managed through a smart contract-based mechanism, ensuring secure and decentralized authorization. A key principle of this approach is self-reliance—enabling interoperability with minimal dependence on third-party products or infrastructure.

2.2.4 Hashed Time-Lock Contract

The Hashed Time-Lock Contract (HTLC) protocol combines hash-lock [2] and time-lock [3] mechanisms to enable atomic cross-chain transactions. In this approach, the initiating party must provide cryptographic proof of the transaction within a specified time window to a smart contract in order to ensure successful execution [Belchior et al., 2021].

Xclaim [Zamyatin et al., 2019] is an interoperability solution based on Hashed Time-Lock Contracts (HTLCs). In this approach, a user u_A who owns an asset on blockchain A (ast_A) can issue a corresponding asset on blockchain B ($\text{ast}_B^{u_A}$), backed by the original asset on blockchain A. The issued asset $\text{ast}_B^{u_A}$ can then be exchanged with any other token on blockchain B owned by another user u_B.To ensure the security of such exchanges, Xclaim relies on a *vault* on blockchain A responsible for handling redeem requests, and a *smart contract* on blockchain B that manages the issuance

[2] `https://en.bitcoin.it/wiki/Hashlock`
[3] `https://en.bitcoin.it/wiki/Timelock`

and exchange processes. The smart contract also monitors the vault's behavior by verifying proofs generated by a relay chain, ensuring the vault acts honestly.

Another HTLC-based interoperability solution is the Multiparty Hash Time-Lock Contract (MP-HTLC) [Barbàra and Schifanella, 2023], which enables cross-chain exchanges between groups of users on blockchain A and blockchain B. This design enhances privacy and reduces transaction fees for participants. The protocol begins with a pre-commitment phase, where all participants collaboratively generate an aggregated public key and a shared secret using a multiparty computation (MPC) protocol; the secret is known only to one group. In the commitment phase, users demonstrate their intention and capability to proceed by either deploying a smart contract or locking funds in escrow. Finally, the redemption phase mirrors traditional HTLCs, using a time-lock and the previously constructed aggregated secret to complete the exchange securely.

2.2.5 Notary Schemes

A notary is characterized as a trusted witness or group of witnesses tasked with validating the claims of typically untrusted blockchains involved in interoperability-based processes, such as proof of asset ownership [Wang et al., 2023].

An illustration of a Notary scheme platform is Bifröst [Scheid et al., 2019]. Operating as a trusted notary, Bifröst facilitates the connection of distributed applications to various blockchains. It manages public and private keys, as well as the credentials hash for completed transactions. The platform consists of three integral components: (1) the *API*, which receives user input and initiates communication with the appro-

priate adapter; (2) the *Adapters*, which format the user input to match the blockchain node requirements and then transmit the transaction; and (3) the *Databases*, which store critical information, such as credentials and transaction hashes, after successful execution.

Focusing on the blockchain interoperability in the realm of the energy sector, BAILIF [Karumba et al., 2023] exemplifies the application of Notary Schemes. It is an interoperability solution that incorporates relays and a *Decentralized Notary Service (DNS)*. Each relay is supported by a *notary node* within the blockchain network, selected via a DNS algorithm based on reputation, reliability, and availability. There are two types of notary nodes: the *source notary* and the *destination notary*. The source notary verifies transaction validity and generates a Merkle proof, which is then sent to the destination notary. The destination notary checks the Merkle proof, executes the transaction on the target blockchain, and returns block commit messages to the source. This process is formalized as the *Cross-chain Attestation and Verification (XCAV)* protocol.

Several notary-based solutions also emphasize *privacy preservation*. One such approach, designed for e-commerce systems, employs elliptic curve cryptography (ECC) and AES for privacy, while incorporating *BLS* [Bacho and Loss, 2022] and *ERC20* [Victor and Lüders, 2019] standards for supervision [Tan et al., 2024]. Another notable example is proposed by Wu et al. [Wu et al., 2023], who introduce a notary scheme using group signatures and blind identities. Their system includes a pair of regulators capable of mapping blind identities to real notary identities—ensuring accountability while maintaining user privacy.

In this context, notary schemes are significantly improved by post-quantum cryptography approaches. For example, one solution in an exchange system employs notaries to verify transactions using a multi-signature scheme based on multivariate cryptographic primitives [Yi, 2023]. Another method introduces an *identity-based quantum multi-signature* mechanism tailored for interoperability in quantum environments. In this system, participants and notaries generate, transmit, and recover qubits while employing a *Quantum Freeze* mechanism to ensure consistency in cross-chain operations [Wang et al., 2022].

2.3 Trust and Notary Scheme

Trust management in blockchain systems has been widely applied across various domains, including the Internet of Things (IoT) [Putra et al., 2023], supply chain management [Batwa and Norrman, 2021], and vehicular networks [Yang et al., 2018], among others. However, since our proposed solution specifically addresses blockchain interoperability using notary schemes, this will be the primary focus of our discussion.

Trust calculation in notary schemes has been explored through various methodologies. Chen et al. [Chen et al., 2024] employed the *PageRank algorithm* [Page, 1999] combined with an *entropy weight method* to evaluate notary node trustworthiness, considering factors such as historical transaction data, node similarity, and trust relationships. Fan and Li [Fan and Li, 2025] proposed an enhanced PageRank algorithm tailored for cross-chain notary schemes, refining trust assessment based on four key factors: transaction efficiency, notarial trust, delay, and user evaluation.

Guo and Hu [Guo and Hu, 2024] introduced a reputation scheme incorporating a

modified *EigenTrust* algorithm [Kamvar et al., 2003] to ensure fairness via transaction history. Their model also utilizes a *verifiable random function (VRF)* for master notary selection and a *cumulative probability method* for selecting notary nodes. Building on this, Cao and Yang [Cao and Yang, 2023] enhanced the PageRank approach by including node quality, initial trust scores, and age bias mitigation. They combined VRF with a *half-life algorithm* to reduce reputational bias toward older nodes.

A broader framework is proposed by Cao et al.[Cao et al., 2023], who designed a cross-chain data traceability system comprising three components: one blockchain for data authorization, another for access control policy management, and a network of notaries. Notary reputation in this system is calculated based on voting outcomes, transaction efficiency, and user feedback, creating a comprehensive evaluation mechanism.

Taking a more dynamic approach, Hu et al. [Hu et al., 2024] developed an interoperability solution with a dynamic reputation system. Their model evaluates notary nodes using a combination of network trust and periodic committee assessments. It introduces a penalty factor for malicious actions, a dynamic evaluation window using a recurrent neural network (RNN) to adapt intervals based on network behavior, and a reputation decay mechanism to penalize inactivity over time.

Chapter 3

Background

In this chapter, we explore the foundational concepts essential to understanding the architecture of the proposed system. We begin by introducing blockchain technology, which serves as the cornerstone of this research. We then proceed to discuss the authentication protocol employed, the consensus mechanism adopted among notaries, and the post-quantum cryptographic schemes integrated into our design.

3.1 Blockchain

Blockchain technology, first introduced by Satoshi Nakamoto [Nakamoto, 2008], is a form of distributed ledger that enables the decentralized recording, verification, and sharing of transactions across a network of nodes. Transactions, once validated, are grouped into blocks and cryptographically linked to preceding blocks, forming a tamper-evident sequence known as a blockchain. This structure ensures data integrity, immutability, and transparency without relying on a central authority. The validation

and agreement process is governed by consensus mechanisms, which define the rules under which nodes agree on the state of the ledger [Guo and Yu, 2022].

Blockchain systems employ a variety of consensus algorithms, including Proof of Work (PoW) [Nakamoto, 2008], Proof of Stake (PoS) [Nguyen et al., 2019], and Proof of Authority (PoA) [Manolache et al., 2022]. Each of these mechanisms presents distinct trade-offs in terms of security, performance, scalability, and energy efficiency.

3.1.1 Structure of a Blockchain

This section begins by outlining the structure of a blockchain transaction and its constituent components. It then explores the structure of a block as the container that holds and organizes these transactions.

In general, a blockchain transaction consists of two fundamental components—inputs and outputs—although specific implementations may include additional or fewer elements [Yaga et al., 2018]:

- **Input:** Refers to the source of funds or assets being transferred. Each input references a previous transaction output that assigned value to the current sender. In systems like Bitcoin, multiple inputs may be aggregated to meet the total amount to be transferred.

- **Output:** Specifies the destination of the transferred value. Each output includes the amount, the recipient's address or public key, and optional conditions for spending. If the total input exceeds the intended transfer amount, a secondary output (change output) is typically created to return the excess to the sender.

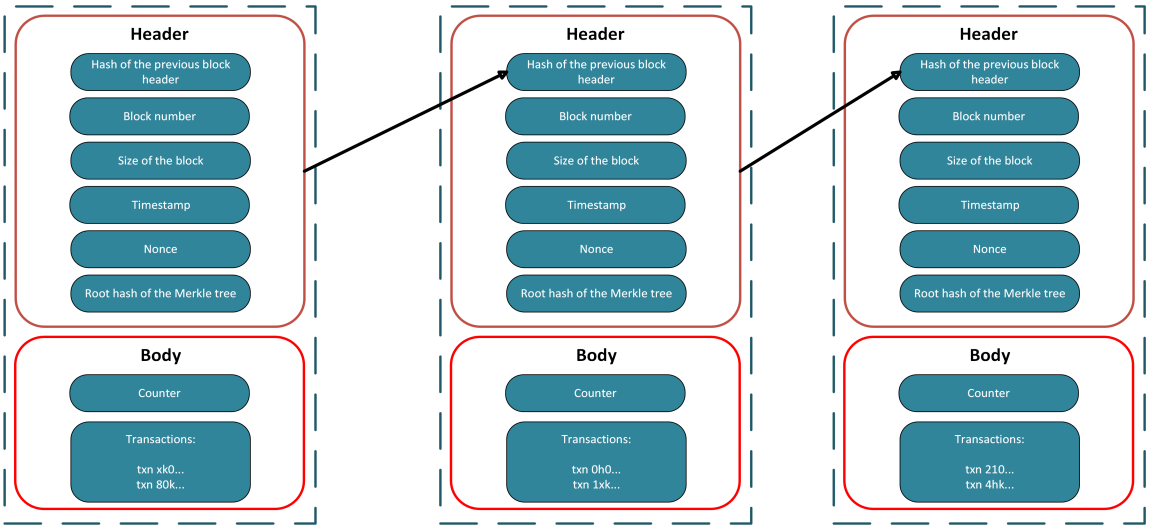

Figure 3.1: An abstract depiction of blocks in a blockchain network

Although the input / output structure is a defining feature of blockchains based on the Unspent Transaction Output (UTxO) model, such as Bitcoin, other platforms, including Ethereum, use an account-based model. In this approach, transactions directly update account balances and are conceptually similar to operations on a global state ledger, without referencing specific previous outputs.

Each block contains multiple transactions and is composed of two main parts: the block header and the block body (or block data) [Golosova and Romanovs, 2018; Yaga et al., 2018; Zheng et al., 2017]. The block header typically includes:

- Block number
- Hash of the previous block header
- Root hash of the Merkle tree representing the block's transactions
- Timestamp indicating when the block was created

- Size of the block
- Nonce (used in consensus mechanisms like PoW)

The block body contains:

- A list of transactions and ledger events
- A transaction counter indicating the number of transactions in the block

Figure 3.1 shows a simplified structure of blocks in a blockchain network.

3.1.2 Blockchain Networks

Blockchain networks are generally categorized into two types: permissioned and permissionless. Permissionless, or public, blockchains—as the name suggests—are open networks that allow anyone to read or submit transactions, create blocks, and participate in the consensus process. These systems are typically open-source, enabling anyone to download the node software and operate a node. However, the openness of such networks introduces a higher risk of malicious activity. To mitigate these risks, permissionless blockchains rely on incentive structures that promote honest behavior, along with consensus mechanisms that require significant computational or financial resources. These mechanisms enhance the network's resilience by making it prohibitively expensive for adversaries to disrupt consensus or gain control. In public blockchains, authentication and identity protocols are typically based on public-key cryptography, allowing anyone to join the network by generating a key pair.

In contrast, permissioned blockchains include an access control layer, with participation governed by network administrators or designated authorities. Only authorized participants can access the system, submit transactions, and interact with the network. As a result, anonymity and pseudonymity are generally not supported. This restricted access enables the use of more efficient consensus mechanisms—such as Practical Byzantine Fault Tolerance (PBFT)—that are less computationally intensive. Moreover, permissioned networks offer high flexibility and customizable configurations, making them well-suited for enterprise and institutional applications. However, a key drawback is their semi-decentralized nature, due to the limited and predefined set of trusted participants. In permissioned blockchains, participants must be registered or whitelisted, and each node may be required to run authentication protocols prior to joining the network.

3.2 Kerberos Authentication Protocol

Kerberos [Neuman and Ts'o, 1994] is a distributed authentication protocol developed based on the Needham and Schroeder authentication protocol [Needham and Schroeder, 1978]. This mechanism allows a client to prove its identity to a service or server. In its basic model, Kerberos consists of the following components:

- **Source Node (Client)**: The node that initiates a request for a service or process.
- **Destination Node (Verifier)**: The node that verifies the ticket and performs the requested process.

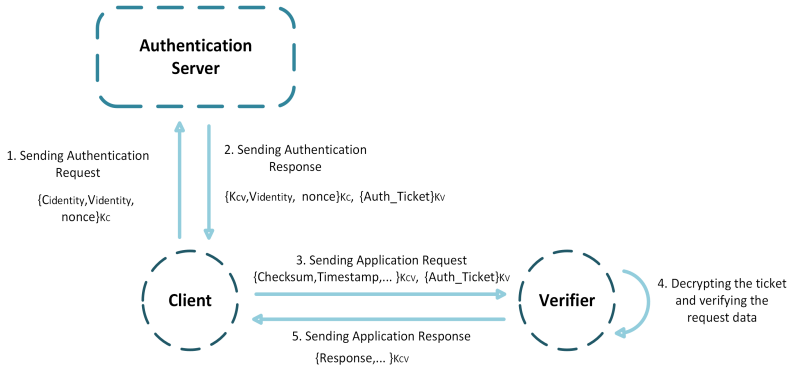

Figure 3.2: Stages of basic Kerberos mechanism.

- **Session Key**: A temporary encryption key generated by the Authentication Server to secure communication between the client and the verifier.
- **Authentication Server (AS)**: A trusted entity that authenticates the client, issues tickets, and generates session keys for establishing secure connections.
- **Authentication Ticket**: A certificate indicating that the client has been authenticated. It includes the session key, client identity, and an expiration time.

The basic Kerberos process, as illustrated in Figure 3.2, consists of the following stages:

1. The client (source node) sends an authentication request to the Authentication Server, which includes the client's identity, the verifier's identity, and a random number (nonce) to match the request and response.
2. The Authentication Server responds with:
    - A message encrypted with the client's key K_C containing the nonce, verifier's identity, and a session key K_{CV}.

- An authentication ticket encrypted with the verifier's key K_V, containing the session key and client information.

3. Upon receiving the response, the client creates an application request consisting of:
   - The authentication ticket received from the AS.
   - An authenticator, which includes a checksum, timestamp, and optionally a sub-session key, encrypted with K_{CV}.
4. The verifier decrypts the ticket using its own key K_V, extracts K_{CV}, then uses it to decrypt the authenticator. It verifies the checksum and checks the timestamp to ensure freshness.
5. The verifier then sends a confirmation response encrypted with the session key K_{CV}.

In a more complete version of the Kerberos protocol, the client does not transmit its identity each time an authentication ticket is needed. Instead, the protocol incorporates a Single Sign-On (SSO) mechanism, in which the client provides its identity once and receives a reusable credential known as the **Ticket Granting Ticket (TGT)**. This TGT is accompanied by a session key $K_{\mathrm{c,tgt}}$, which the client uses to request additional authentication tickets without having to re-authenticate.

To support this functionality, the full Kerberos system introduces two dedicated servers responsible for ticket issuance:

- **Authentication Server (AS)**: As in the basic Kerberos model, the AS is

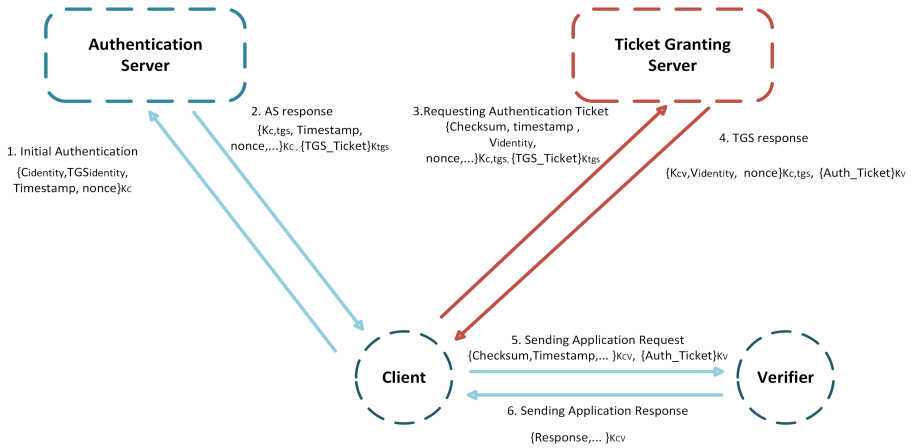

Figure 3.3: Stages of complete Kerberos mechanism.

responsible for authenticating the client. However, instead of issuing a verifier-specific authentication ticket, it issues a TGT along with a session key $K_{\mathrm{c,tgs}}$. These credentials are used by the client to request future tickets from the Ticket Granting Server.

- **Ticket Granting Server (TGS)**: The TGS issues authentication tickets for specific services (verifiers). It verifies the TGT and, if valid, returns an authentication ticket encrypted with the verifier's secret key K_v, as well as a session key $K_{\mathrm{c,v}}$ for secure communication between the client and the verifier.

The complete Kerberos process, as illustrated in Figure 3.3, consists of the following stages:

1. **Initial Authentication**: The client sends an authentication request to the Authentication Server (AS), including its own identity, the identity of the Ticket Granting Server (TGS), a timestamp, and a nonce.

2. **AS Response**: The AS responds with two items:

   - A session key $K_{\text{c,tgs}}$ and a timestamp, encrypted with the client's secret key K_c.
   - A Ticket Granting Ticket (TGT), encrypted with the TGS's secret key K_{tgs}, containing the client's identity, $K_{\text{c,tgs}}$, expiration time, and other metadata.

3. **Requesting a Service Ticket**: When the client wants to access a specific service (verifier), it sends the TGT and an authenticator (e.g., timestamp and checksum encrypted with $K_{\text{c,tgs}}$) to the TGS.

4. **TGS Response**: Upon validating the TGT and authenticator, the TGS issues:

   - A service ticket encrypted with the verifier's secret key K_v.
   - A session key $K_{\text{c,v}}$, timestamp, and other metadata encrypted with $K_{\text{c,tgs}}$.

5. **Client-Verifier Authentication**: The client uses the session key $K_{\text{c,v}}$ to encrypt a new authenticator and sends it along with the service ticket to the verifier. This authenticator confirms the client's identity and the freshness of the request.

6. **Verifier Response**: The verifier decrypts the service ticket using its key K_v, extracts the session key $K_{\text{c,v}}$, and verifies the client's authenticator. If valid, it responds with a message encrypted using $K_{\text{c,v}}$, completing mutual authentication.

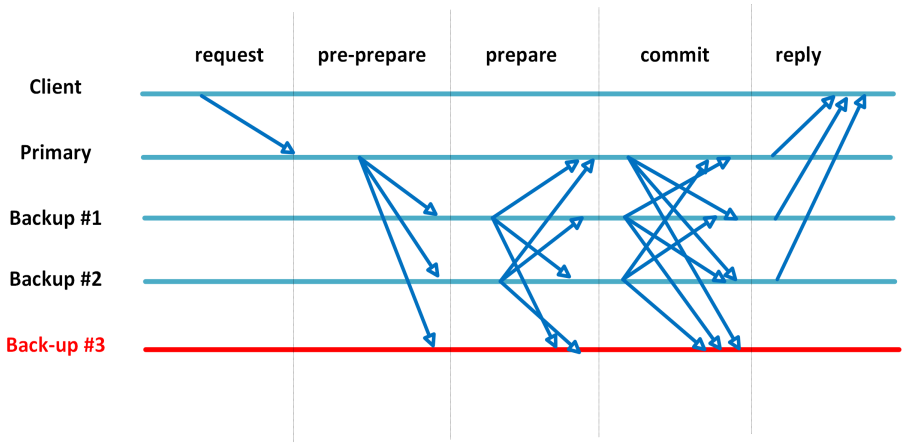

Figure 3.4: Illustration of the normal-case PBFT protocol involving one primary and three backup replicas, with replica 3 exhibiting faulty behavior [Castro et al., 1999].

3.3 Practical Byzantine Fault Tolerance (PBFT)

Practical Byzantine Fault Tolerance (PBFT) [Castro et al., 1999] is a consensus algorithm that allows nodes in a distributed system to reach agreement even in the presence of Byzantine faults. Specifically, PBFT can tolerate up to f faulty nodes out of $3f + 1$ total nodes, provided that at least $2f + 1$ nodes operate correctly. The protocol models the service as a replicated state machine distributed across all participating nodes. Each replica progresses through sequential configurations known as *views*. In each view, one node is designated as the *primary* (or *leader*), while the others function as *backups*.

The view number increments over time, and the primary for a given view is selected using the following formula:

$$p = v \bmod R \tag{3.1}$$

where p is the index of the primary node, v is the view number, and R is the total number of replicas. This mechanism ensures that all nodes eventually take turns

acting as the primary.

The PBFT protocol proceeds through the following steps:

1. The client sends a service request to the primary node.
2. The primary multicasts the request to all backup nodes.
3. Each replica executes the request and replies to the client with the result.
4. The client waits for $f + 1$ matching responses to confirm the correctness of the result.

When the primary node receives a request from a client, it initiates a three-phase protocol consisting of the *pre-prepare*, *prepare*, and *commit* phases, as illustrated in Figure 3.4.

In the *pre-prepare* phase, the primary node sends a message containing the sequence number n, view number v, the digest of the client request d, and the request message m, all encapsulated in a `PRE-PREPARE` message in the following format:

$$((\text{PRE-PREPARE}, n, v, d, \sigma), m)$$

where σ denotes the primary's cryptographic signature.

Upon receiving this message, each replica performs several validation checks: it verifies the digest, sequence number, view number, and the digital signature. If all checks pass, the replica accepts the `PRE-PREPARE` message and proceeds to the next phase.

In the *prepare* phase, the replica constructs and multicasts a `PREPARE` message to all other replicas in the following format:

$$(\text{PREPARE}, n, v, d, \sigma, id)$$

where *id* denotes the replica's identifier. The replica waits to receive $2f$ `PREPARE` messages from distinct replicas. It then ensures that all received `PREPARE` messages, along with the accepted `PRE-PREPARE` message, agree on the sequence number n, view number v, and digest d. If so, the replica proceeds to the *commit* phase.

In the *commit* phase, the replica multicasts a `COMMIT` message in the following format:

$$(\text{COMMIT}, n, v, d, \sigma, id)$$

Once a replica receives $2f + 1$ valid `COMMIT` messages from distinct replicas—each matching the sequence number, view number, and digest—it executes the corresponding service operation associated with the client request.

3.4 Lattice Based Cryptography

In order to understand lattice-based cryptography, we must first define what a lattice is. Given a set of linearly independent vectors $a_1, a_2, \ldots, a_n$, a *lattice* is the set of all linear combinations of these vectors with integer coefficients $x_i \in \mathbb{Z}$ [Paar et al., 2024]. These vectors are referred to as *basis vectors* of the lattice.

$$L = \{x_1 a_1 + x_2 a_2 + \cdots + x_n a_n \mid x_i \in \mathbb{Z}\}$$

The *dimension* of the lattice corresponds to the dimension of the space spanned by the basis vectors, and the number of basis vectors determines the *rank* of the lattice.

Another essential concept in lattice-based cryptography is the *Learning With Er-*

rors (LWE) problem, which serves as the foundational hard problem for many lattice-based cryptographic schemes, including the one used in this thesis, ML-KEM, which utilizes an extended version of LWE.

In simple terms, the LWE problem can be described as follows[Paar et al., 2024]: Given a matrix $A \in \mathbb{Z}_q^{m \times n}$ the goal is to find a secret vector $s = (s_1, s_2, \ldots, s_n)$ with $s_i \in \mathbb{Z}_q$ such that:

$$A \cdot s + e \equiv t \pmod{q} \tag{3.2}$$

Here, e is an unknown error vector composed of small integers.

In order to introduce a more formal definition of the Learning With Errors (LWE) problem [Lyubashevsky, 2024], we first explain some notations.

We define the notation $[\beta]$ for any $\beta \in \mathbb{Z}$ with $\beta > 0$ as:

$$[\beta] := \{x \in \mathbb{Z} \mid -\beta \leq x \leq \beta\}. \tag{3.3}$$

The notation $a \leftarrow [\beta]$ means that all integer coefficients a_i of the fixed-degree polynomials a are chosen uniformly at random from the set $[\beta]$.

Similarly, the notation $\mathbf{s} \leftarrow [\beta]^m$ indicates that each polynomial s_i of the vector $\mathbf{s} \in \mathbb{Z}[X]^m$ is independently sampled from the distribution $[\beta]$ as $s_i \leftarrow [\beta]$. Our final notation is $\lceil x \rfloor$, which denotes the integer closest to x , with ties being broken upwards.

Having known the notations, we can define the Learning With Errors (LWE) problem more formally:

Given positive integers m, n, q, and $\beta < q$, the $\mathrm{LWE}_{n,m,q,\beta}$ problem is defined as the task of distinguishing between the following two distributions:

- $(A, As + e)$, where $A \leftarrow \mathbb{Z}_q^{n \times m}$, $s \leftarrow [\beta]^m$, and $e \leftarrow [\beta]^n$;

- (A, u), where $A \leftarrow \mathbb{Z}_q^{n \times m}$ and $u \leftarrow \mathbb{Z}_q^n$.

The hardness of the LWE problem is based on the assumption that no efficient algorithm can reliably distinguish between samples drawn from the two distributions.

Having established the necessary background, we now present an encryption scheme based on the hardness of the $\text{LWE}_{m,q,\beta}$ problem. Assume we want to encrypt a message $\mu \in \{0, 1\}$.

We define the key pair as follows:

$$\text{Secret key (sk)}: \quad s \leftarrow [\beta]^m$$

$$\text{Public key (pk)}: \quad A \leftarrow \mathbb{Z}_q^{m \times m}, \quad t = As + e_1, \quad \text{where } e_1 \leftarrow [\beta]^m$$

To encrypt a message $\mu \in \{0, 1\}$, we first sample random vectors $r \leftarrow [\beta]^m$, $e_2 \leftarrow [\beta]^m$, and $e_3 \leftarrow [\beta]$. Then the ciphertext is computed as:

$$(u^T = r^T A + e_2^T, \quad v = r^T t + e_3 + \lceil \frac{q}{2} \rfloor \mu) \tag{3.4}$$

In this formaula, In order to decrypt, we will compute $v - u^T s$:

$$v - u^T s = r^T (As + e_1) + e_3 + \frac{q}{2}\mu - (r^T A + e_2^T)s \tag{3.5}$$

as a result, we will have:

$$v - u^T s = r^T e_1 + e_3 + \frac{q}{2}\mu - e_2^T s \tag{3.6}$$

Considering the equation above, one can see that the $(r^T As)$ terms cancel out, and what remains is a combination of the error terms. As all the error term coefficients

are between $-\beta$ and β, the vector products $r^T e_1$ and $e_2^T s$ each consist of m terms with magnitudes at most β^2.

As a result, we can rewrite the equation above as:

$$e + \frac{q}{2}\mu \tag{3.7}$$

where $e \in [\ 2m\beta^2 + \beta]$. If the parameters are set such that:

$$2m\beta^2 + \beta < \frac{q}{4} \tag{3.8}$$

then μ can be correctly determined by checking whether the value of $v - u^T s$ is closer to 0 or to $\frac{q}{2}$.

Chapter 4

Architecture

4.1 General Architecture

To investigate the proposed research questions, we designed and developed an interoperability solution that connects two distinct blockchains—Hyperledger Fabric

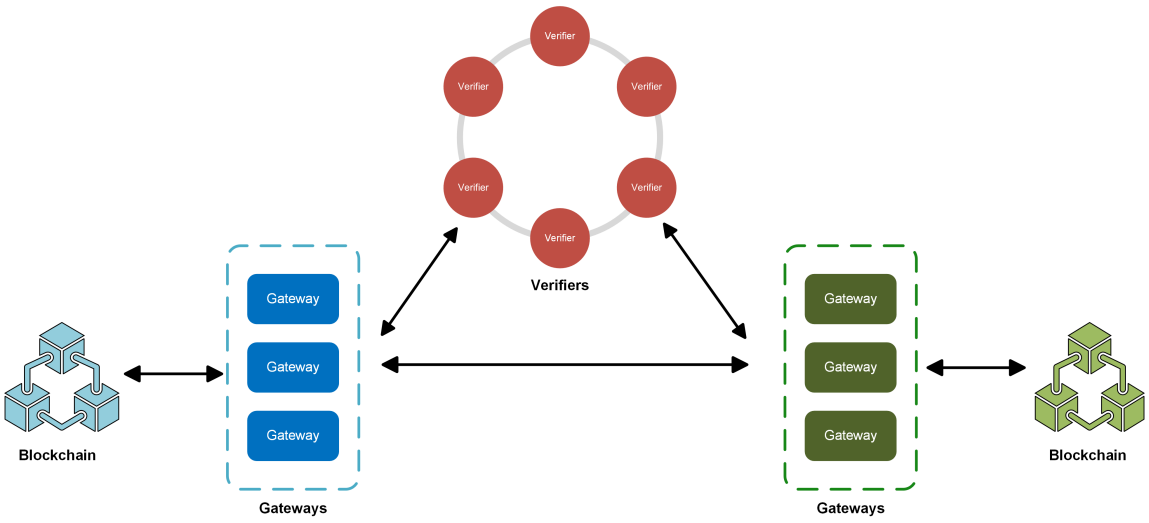

Figure 4.1: An overview of the general architecture

and Ethereum—within a permissioned environment, specifically enabling contract call operations. For the architectural design, component allocation, and role assignment, we adopted a distributed notary scheme approach to interoperability.

This approach was selected for its scalability, flexibility, and ability to enable the direct involvement of participants by allowing them to act as notaries. It aligns well with the specific requirements of the proposed project, particularly in scenarios where accountability, identity, and permissioning are critical.

We chose the notary scheme over the Blockchain of Blockchains approach because the solution focuses on service-based contract calls rather than financial interactions. Introducing an additional ledger in an environment without significant added value would unnecessarily increase complexity. Additionally, the permissioned setting assumes a baseline level of trust among participants, making the use of an intermediary blockchain to reinforce trust largely redundant.

Similarly, we opted against the Hashed Time-lock Contract (HTLC) approach, as the system does not involve token transfers or asset swaps that require cryptographic hash locks or time constraints.

Compared to Trusted Relays, the notary scheme provides better resilience and verifiability. In a relay-based architecture, a compromised relay node could jeopardize the security and reliability of the entire system. In contrast, a decentralized notary scheme based on the Practical Byzantine Fault Tolerance (PBFT) model can tolerate up to f faulty nodes in a $3f + 1$ configuration. As long as at least $2f + 1$ nodes behave correctly, the system maintain its integrity and continue to operate reliably. Moreover, because notary nodes independently verify and sign events from the

connected blockchains, the scheme provides enhanced transparency and auditability.

Lastly, while sidechain-based or two-way peg interoperability solutions are prevalent in rollup architectures, these approaches fall outside the scope of this research.

4.1.1 Components

The proposed system consists of two main components: *Gateways* and *Verifiers*. Gateways are nodes that connect a blockchain to the interoperability network, acting as the entry and exit points for cross-chain communication. Verifiers are responsible for validating claims made by Gateways, initiating the proof generation process, and issuing authentication tickets for each interaction between Gateways.

To participate in the permissioned interoperability network, each participant must provide at least one Gateway—to connect their blockchain to the system—and one Verifier—to engage in the verification and authentication processes. Figure 4.1 illustrates the relationship between these components.

Gateways

Gateways serve as the bridge between a blockchain platform and the interoperability network. They are responsible for invoking contracts on other blockchain platforms, generating cryptographic proofs, initiating the verification process, and triggering the Kerberos-based authentication mechanism. As illustrated in Figure 4.2, a Gateway comprises the following main components:

- **Network**: Handles communication between the Gateway and the interoperability network. This component is responsible for creating and processing requests

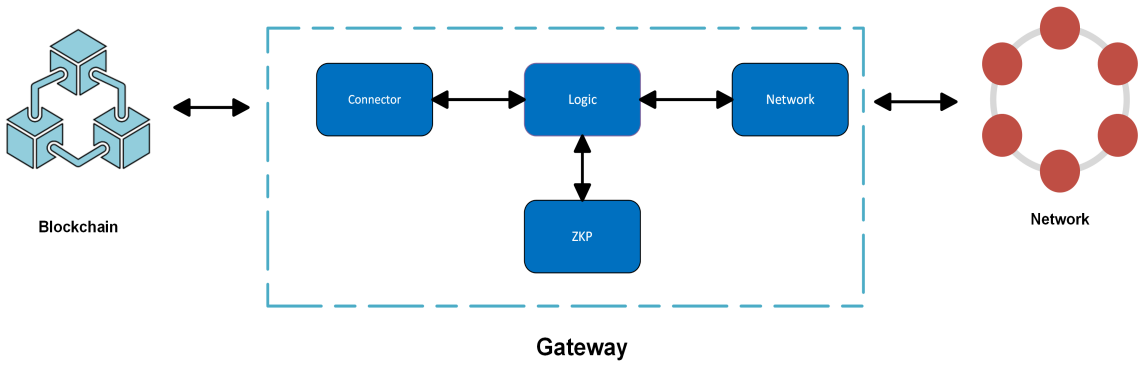

Figure 4.2: An overview of the components of a Gateway

and responses, as well as managing the marshaling and unmarshaling of data.

- **Logic**: Serves as the core of the Gateway, connecting all internal components. It manages cryptographic operations, analyzes incoming requests, and applies the appropriate logic to process them.
- **Connector**: Links the Gateway to its respective blockchain platform using the platform's software development kit (SDK). It is responsible for invoking smart contracts, calling services, and returning the results.
- **ZKP (Zero-Knowledge Proof)**: Manages the generation and verification of zero-knowledge proofs. Whenever the Gateway needs to produce a proof related to blockchain activity, this module executes all required procedures.

Verifiers

Verifiers play a central role in the proposed interoperability solution. Each participant in the network must deploy at least one Verifier node. A Verifier performs several critical functions within the system, including:

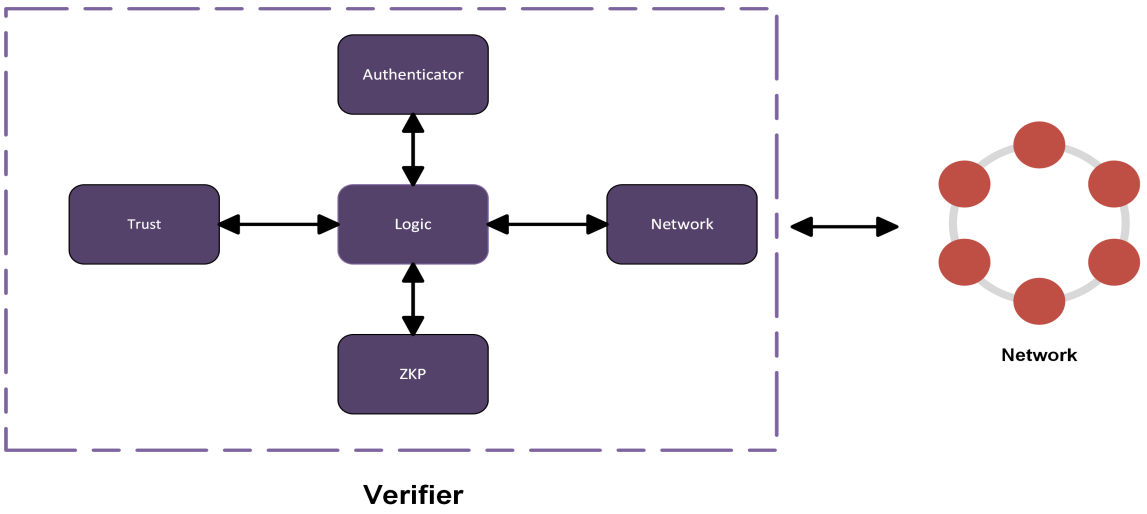

Figure 4.3: An overview of the components of a Verifier

1. **Notary Function:** Acts as a notary node by monitoring changes within the blockchain network, verifying the correctness of operations, and ensuring the integrity of other Verifiers through participation in the consensus mechanism and validation of generated proofs.

2. **Trusted Setup for zk-SNARKs:** Serves as the trusted setup authority for the zk-SNARK proof mechanism by generating the cryptographic parameters required for both proof generation and verification.

3. **Authentication Server:** Operates as the authentication server within the Kerberos mechanism, issuing tickets to enable secure communication between Gateways.

4. **Trust and Behavior Monitoring:** Monitors the behavior of peer Verifiers to detect malicious activity and dynamically adjust trust levels. This is achieved

through the consensus mechanism, where Verifiers compare verification results and update trust scores accordingly.

The Verifier entity consists of several internal components that support these core responsibilities. These roles, illustrated in Figure 4.3, are as follows:

- **Network:** Connects the Verifier to other Verifiers and Gateways. It is responsible for creating and transmitting requests and responses, and, like in the Gateway, it manages the marshaling and unmarshaling of data.
- **Logic:** Interprets incoming messages, applies system logic, connects internal components, and manages cryptographic operations and message verification.
- **ZKP:** Handles two key tasks: (i) generating zk-SNARK parameters based on the Powers of Tau ceremony [Wang et al., 2025], and (ii) verifying the zero-knowledge proofs submitted by Gateways.
- **Trust:** Collaborates with the ZKP component to manage the consensus mechanism. It maintains trust scores associated with peer Verifiers, uses these scores to elect consensus committees, participates in verification, and updates trust scores based on observed behavior during consensus.
- **Authenticator:** Manages Gateway authentication using the Kerberos protocol. It issues authentication tickets and enables secure communication through symmetric keys established during the Kerberos exchange with participating Gateways.

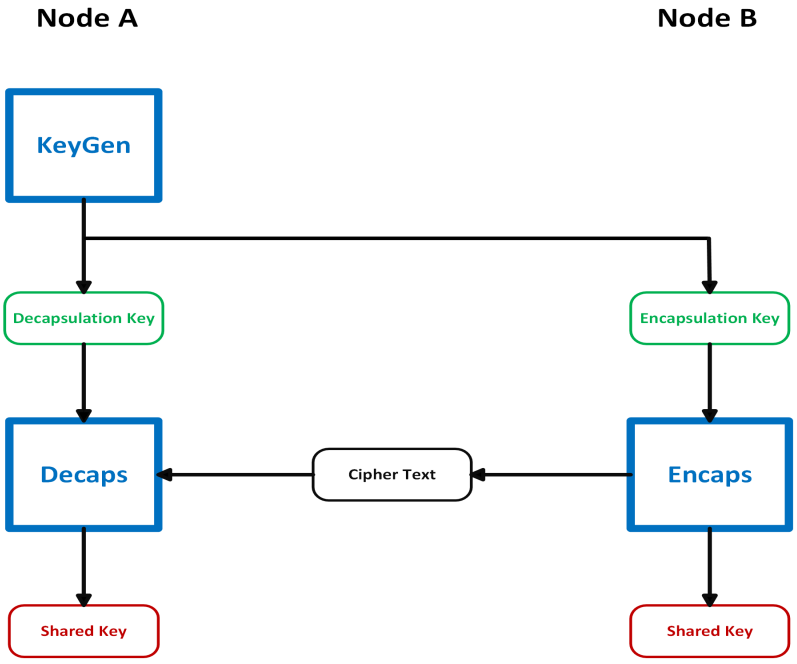

Figure 4.4: ML-KEM mechanism between two nodes in the system [FIPS, 2024a].

4.2 Security Architecture

This section examines the security architecture of the proposed interoperability solution. Key aspects include the use of cryptographic mechanisms such as key encapsulation, encryption strategies, message authentication between Gateways, message integrity and validity checks, the proof generation and verification process, and the consensus mechanism employed by Verifiers.

4.2.1 Key-Encapsulation Mechanism (KEM)

To create shared keys between different nodes in the system, we employ ML-KEM as the key encapsulation mechanism. based on this algorithm, during the initialization

phase, each node generates a key pair. When communication is initiated, the source node transmits its encapsulation key to the destination node. The destination node then generates a ciphertext and a shared key, and returns the ciphertext to the source node. Using its decapsulation key, the source node extracts the shared key.

This process is repeated twice to derive two independent symmetric keys: one for encryption and the other for verifying authenticity and integrity using *HMAC*. Figure 4.4 illustrates the ML-KEM key exchange process between a Gateway and a Verifier. Table 4.1 presents the different versions of ML-KEM along with their respective key sizes. All three versions of ML-KEM are supported in the proposed system.

Version	**Encapsulation Key**	**Decapsulation Key**	**Ciphertext**
ML-KEM-512	800	1632	768
ML-KEM-768	1184	2400	1088
ML-KEM-1024	1568	3168	1568

Table 4.1: Different versions of supported ML-KEM with associated key sizes (in bytes).

4.2.2 Encryption

In the proposed system, encryption is employed for two primary purposes:

1. **Message Encryption Between Nodes:** Communication between nodes is secured using AES-256 encryption. The encryption key for this purpose is derived during the key encapsulation process, ensuring that only the intended recipient can decrypt the message.

2. **Secure Storage of the Shared Key:** Once established, the shared key is

securely stored in a database. To encrypt this key, a separate encryption key is derived using parameters such as an administrator password, a salt, and an iteration count. These parameters are passed to the PBKDF2 (Password-Based Key Derivation Function) algorithm [Ertaul et al., 2016], which generates a strong cryptographic key. To ensure the encryption key can be regenerated in the future, the salt and iteration count are also stored.

4.2.3 Digital Signature Algorithm (DSA)

To ensure the integrity and authenticity of messages—particularly in cases where non-repudiation is required—We employ digital signature algorithms. To maintain the quantum-resistant properties of the proposed solution, we used ML-DSA as the digital signature scheme.

In the proposed system, each participating node generates a key pair for ML-DSA. When a node needs to send a message that requires non-repudiation, it signs the message using its private key. The recipient node then verifies the signature using the sender's public key. As indicated in Table 4.2, ML-DSA has three versions, all of which are supported in the proposed system.

Version	**Private Key**	**Public Key**	**Signature**
ML-DSA-44	2560	1312	2420
ML-DSA-65	4032	1952	3309
ML-DSA-87	4896	2592	4627

Table 4.2: Supported ML-DSA versions with associated key and signature sizes (in bytes).

Given the distributed nature of the system, no centralized certificate authority is used. Instead, public key verification is handled using a model inspired by the Web

of Trust [Caronni, 2000]. In this model, the authenticity of a public key is endorsed by a trusted participant in the network. Nodes that trust this participant also extend trust to the certificates it issues.

In this system, each participant must operate at least one Verifier and one Gateway. A Verifier can join the network if an existing Verifier—designated as the *Bootstrap Verifier*—trusts it and signs its public key. Upon receiving this certificate, the new Verifier is trusted by all Verifiers that already trust the Bootstrap Verifier. A participant's Gateway may either join the network simultaneously with its associated Verifier or independently request a certificate from the Bootstrap Verifier.

4.2.4 Hash-based Message Authentication Code (HMAC)

In the proposed system, *Hash-based Message Authentication Code (HMAC)* [Turner, 2008] is used to verify the authenticity and integrity of messages in scenarios where non-repudiation is not required—such as during regular information exchanges between nodes. To implement this mechanism, nodes first establish a dedicated shared secret key specifically for use with HMAC. The system employs SHA-256 in conjunction with a 32-byte shared key to generate a unique hash for each message. This hash is transmitted alongside the message, enabling the receiving node to validate the authenticity and integrity of the received content.

4.2.5 Authentication Among Gateways

To authenticate communication between Gateways, the proposed system implements a mechanism inspired by the Kerberos protocol [Neuman and Ts'o, 1994]. The

process, illustrated in Figure 4.5, proceeds as follows:

1. The source Gateway requests a communication ticket from a Verifier node.

2. The Verifier authenticates the source Gateway by verifying the digital signature (DSA) attached to the request.

3. Upon successful authentication, the Verifier generates a random session key.

4. The Verifier constructs a ticket containing the session key, the source Gateway's IP address, the destination Gateway's IP address, the ticket's creation time, and its lifetime. This ticket is then encrypted using the destination Gateway's public key.

5. The Verifier sends a response to the source Gateway. This response includes the session key and the encrypted ticket, both encrypted with the source Gateway's public key.

6. The source Gateway sends the actual message—encrypted with the session key—along with the encrypted ticket to the destination Gateway.

7. The destination Gateway decrypts the ticket using its private key and verifies its contents (IP addresses and validity period).

8. If the ticket is valid, the destination Gateway uses the session key from the ticket to decrypt the received message.

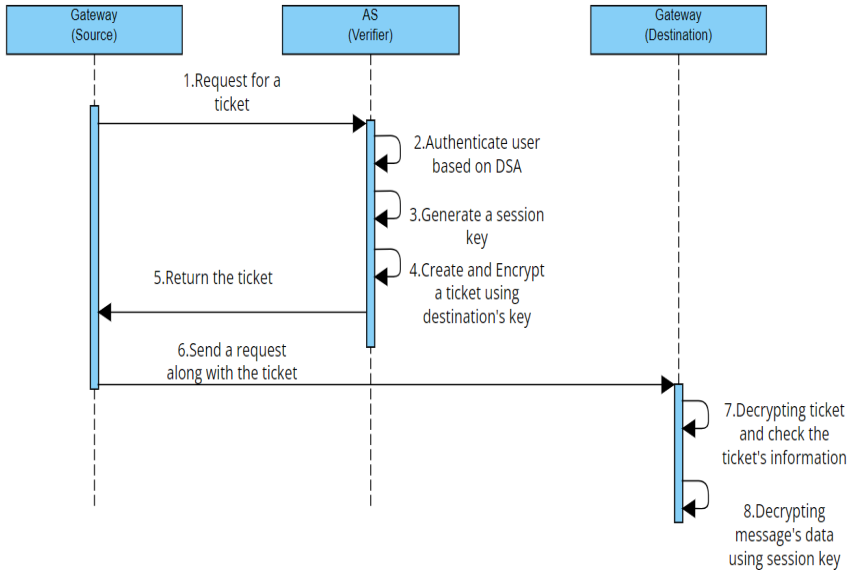

Figure 4.5: Kerberos-based authentication mechanism used in the proposed system.

4.2.6 Proof Mechanism

To provide verifiable evidence of specific actions performed on the blockchain, the system requires the Gateway to generate a cryptographic proof. The proof mechanism is implemented using *zk-SNARKs* [Petkus, 2019], specifically adopting the *Groth16* protocol [Groth, 2016] due to its efficiency and succinct proof size.

Within the system architecture, the process proceeds as follows: the source Gateway initiates a request for a service. The destination Gateway, which provides the service, generates a zero-knowledge proof attesting that the requested service has been executed correctly. This proof is then sent back to the source Gateway, which forwards it to the Verifier consensus committee. Upon reaching consensus on the validity of the proof, the source Gateway acknowledges the successful execution of the service on the destination blockchain and proceeds to apply the corresponding

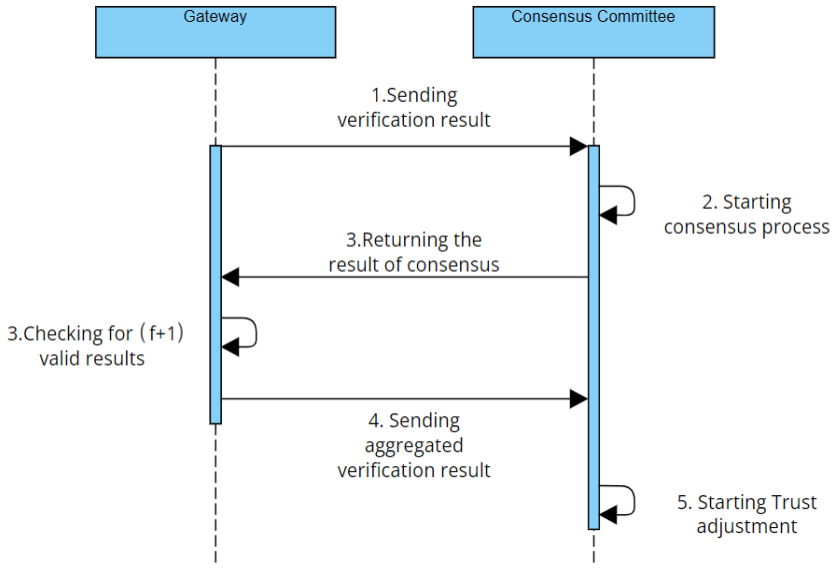

Figure 4.6: Consensus and trust adjustment mechanism in the proposed notary-based architecture.

business logic.

4.2.7 Consensus Mechanism and Trust Adjustment

Consensus plays a crucial role in ensuring the reliability, security, and trustworthiness of the proposed notary-based blockchain interoperability system. Its importance is highlighted across the following four dimensions:

1. **Validation of Actions and Proofs**

   No single Verifier is trusted to unilaterally validate a proof. Instead, multiple Verifiers independently assess the submitted proof, and consensus ensures that a predefined threshold of agreement—specifically, $f+1$ out of $3f+1$ Verifiers—is reached before an action is accepted as valid. This mechanism mitigates the risk of erroneous or dishonest validations.

2. **Ensuring Trust and Decentralization**

   The system is inherently decentralized, with no single Verifier holding absolute authority. Consensus distributes the decision-making process across multiple Verifiers, reducing the risk of single points of failure and enhancing overall system fault tolerance.

3. **Prevention of Malicious Behavior**

   Verifiers may act maliciously or become compromised. The consensus protocol ensures that such behavior cannot influence outcomes unless a majority of Verifiers are dishonest—an assumption the system design considers highly improbable.

4. **Accountability and Trust Score Management**

   By comparing individual verification results with those shared by peers, Verifiers can identify inconsistencies. This enables dynamic trust score adjustments and allows the system to penalize Verifiers that consistently deviate from the consensus.

To meet these objectives, the system adopts the Practical Byzantine Fault Tolerance (PBFT) consensus mechanism [Castro et al., 1999].

When a Gateway receives a proof of action (e.g., a zk-SNARK proof), it forwards the proof to a designated Verifier committee. Each Verifier independently validates the proof and returns its result to the Gateway. For the proof to be accepted, the Gateway must receive at least $f+1$ positive responses, where f denotes the maximum number of faulty Verifiers the system can tolerate.

After consensus is reached, Verifiers share their verification outcomes with one another. The Gateway also broadcasts its aggregated result to the Verifiers. By comparing these shared results, each Verifier can detect discrepancies and adjust the trust levels of its peers accordingly. Figure 4.6 illustrates the full consensus workflow and the trust adjustment process.

Chapter 5

System and Trust Modeling

In this chapter, we present a formal model of the proposed architecture, along with the novel trust model. To model the architecture, we begin by describing its core components, followed by a detailed walkthrough of the main stages of the solution. After establishing the architectural model, we introduce a dedicated model for the trust mechanism, which is grounded in Popoviciu's inequality and incorporates dynamic trust adjustment.

5.1 System Model

To effectively model the proposed solution, we divide its main operations into three distinct phases:

Initialization: This phase focuses on the system's initial processes, including key pair generation, key encapsulation for symmetric encryption, and key encapsulation for the HMAC mechanism. In this phase, we also model the information-sharing

procedures that occur when new nodes join the system.

Cross-chain Interaction: This phase covers the cross-chain communication process, including ticket requests and message authentication using the Kerberos-based protocol. We then model how the destination Gateway parses the ticket and completes the cross-chain interaction.

Proof Mechanism and Trust Management: In this final phase, we model the steps involved in proof generation and verification, followed by the consensus process among Gateways. Finally, we present the trust adjustment mechanism among Verifiers and the triggers that influence it.

5.1.1 Modeling components

First, we define the system's main components. Formally, we represent a blockchain as a state machine $B = (S, A, T, S_0)$, where S represents the set of states of the blockchain network, A is the set of actions that trigger state transitions (e.g., transactions), $T : S \times A \to S$ is the state transition function that maps a current state and an action to a new state, and S_0 is the initial state of the blockchain.

The system comprises a set of Blockchain networks defined as $B = B_{\text{ETH}} \cup B_{\text{HLF}}$, where:

- $B_{\text{ETH}} = \{B_{\text{ETH},1}, B_{\text{ETH},2}, \ldots, B_{\text{ETH},j}\}$ represents the EVM-based networks, and
- $B_{\text{HLF}} = \{B_{\text{HLF},1}, B_{\text{HLF},2}, \ldots, B_{\text{HLF},i}\}$ represents the Hyperledger Fabric networks (as permissioned networks) in our scheme.

We define the set of Gateways in our system as

$$G = \{g_1, g_2, \ldots, g_i\}$$

Each Gateway possesses a key pair for digital signatures $(pk_{\text{sig}}, sk_{\text{sig}})$; a key pair for key encapsulation used between the Gateway and Verifiers for encryption $(pk_{\text{kem}}^{\text{enc}}; sk_{\text{kem}}^{\text{enc}})$; and another key pair for key encapsulation used for HMAC operations $(pk_{\text{kem}}^{\text{hmac}}, sk_{\text{kem}}^{\text{hmac}})$ It also stores the address and public key of the bootstrap node $(\text{addr}_{\text{b}}, pk_{\text{b}})$; a symmetric key for encryption $\text{sym\_key}_{g_i}^{\text{enc}}$; a symmetric key for HMAC $\text{sym\_key}_{g_i}^{\text{hmac}}$; and a reference to the blockchain network with which it directly interacts, denoted as B_i.

Each Verifier is characterized by a set of properties, including a key pair for digital signatures $(pk_{\text{sig}}, sk_{\text{sig}})$; a key pair for key encapsulation used between Verifiers for encryption $(pk_{\text{kem}}^{\text{enc}}, sk_{\text{kem}}^{\text{enc}})$; another key pair for key encapsulation used for HMAC operations $(pk_{\text{kem}}^{\text{hmac}}, sk_{\text{kem}}^{\text{hmac}})$; the address and public key of the bootstrap node $(\text{addr}_{\text{b}}, pk_{\text{b}})$; and a trust score $\mathcal{TS}_{v_i}$ that is determined by the participation of a verifier in PBFT consensus. Each Verifier also maintains a symmetric key $\text{sym\_key}_{v_i}^{\text{enc}}$ derived from the key encapsulation process for encryption, and another symmetric key $\text{sym\_key}_{v_i}^{\text{hmac}}$ for HMAC-based message authentication. We define the set of Verifiers as

$$V = \{v_1, v_2, \ldots, v_i\}$$

5.1.2 Initialization

In this section, we model the three main steps of the initialization phase: Key Generation, Key Encapsulation, and Information Retrieval.

Key Generation

Upon launch, as the first step, each node in the system generates three key pairs. The first key pair is for the digital signature, generated using the ML-DSA scheme: $(pk_{\text{sig}}, sk_{\text{sig}})$ = ML-DSA.KeyGen(). The node then generates two additional key pairs using the ML-KEM scheme: one for deriving a symmetric key used for encryption under the AES-256 protocol: $(pk_{\text{kem}}^{\text{enc}}, sk_{\text{kem}}^{\text{enc}})$ = ML-KEM.KeyGen(), and another for deriving a symmetric key used in the HMAC mechanism: $(pk_{\text{kem}}^{\text{hmac}}, sk_{\text{kem}}^{\text{hmac}})$ = ML-KEM.KeyGen() Generating these three sets of keys prepares each node to participate in the subsequent steps of the initialization phase.

Key Encapsulation

In order to perform the key encapsulation mechanism, as depicted in Figure 5.1, each node must interact with a Verifier known as the bootstrap Verifier v_B, whose address and public key are assumed to be known in advance. To establish a shared symmetric key between a Gateway g_i and v_B, the Gateway first creates a key encapsulation request message structured as:

$$\text{Msg}_{\text{kem}} = (pk_{\text{sig}}, pk_{\text{kem}}, \sigma, \text{addr}_g, \text{dsa\_scheme\_name}, \text{kem\_scheme\_name})$$

where σ = ML-DSA.Sign$(\text{Msg}_{\text{kem}}, sk_{\text{sig}})$ is the digital signature of g_i over the message, kem_scheme_name specifies the version of the ML-KEM scheme being used, and dsa_scheme_name indicates the version of the ML-DSA scheme.

v_B, upon receiving the message, first checks the validity of the signature using

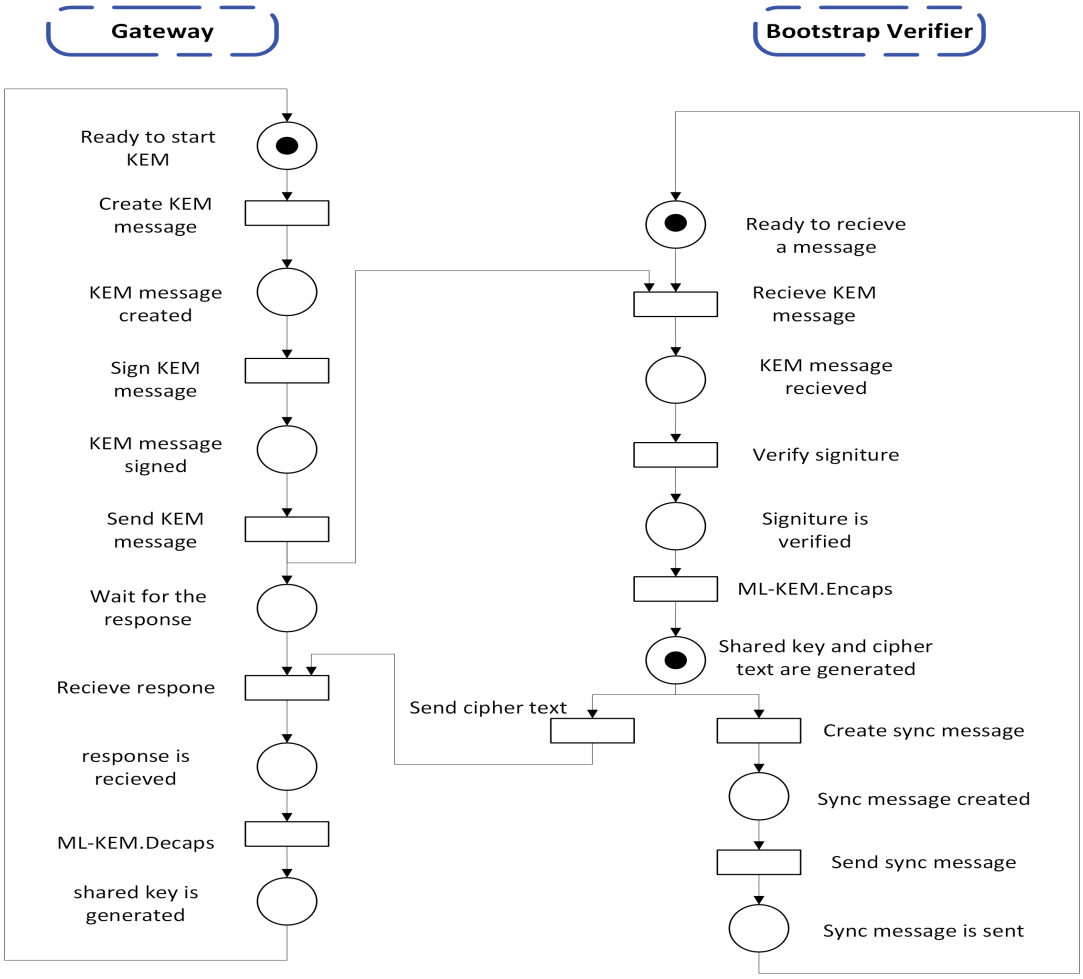

Figure 5.1: Illustration of the Key Encapsulation Mechanism.

ML-DSA.Verify($\text{Msg}_{\text{kem}}, \sigma, pk_{\text{sig}}$). If the verification is successful, v_B stores the Gateway's public keys in order to use it for the future verifications.

Next, it initiates the key encapsulation process. v_B generates a ciphertext and a shared key as: $\text{sym\_key}^{\text{enc}}_{v,g_i}, \text{ct} = \text{ML-KEM.Encaps}(pk_{\text{kem}})$. The ciphertext is signed and sent back to g_i.

Subsequently, v_B distributes the shared key and related metadata—such as the public key and gateway address, to the other Verifiers to facilitate secure communica-

tion between g_i and the Verifier network. The synchronization message is structured as:

$$\text{Msg}_{\text{sync}_{g_i}} = (\text{ct}, pk_{\text{sig}}, pk_{\text{kem}}, \sigma, \text{addr}_g, \text{dsa\_scheme\_name}, \text{kem\_scheme\_name})$$

On the Gateway side, g_i, using ct, this node regenerates the shared key using the private (decapsulation) key $\text{sym\_key}_{v,g_i}^{\text{enc}} = \text{ML-KEM.Decaps}(sk_{\text{kem}}, ct)$.

To ensure continued security, this mechanism is periodically repeated at fixed time intervals. The same process is also applied when new Verifiers join the system and need to establish trust with v_B.

Get Information

Upon completing the key distribution between a node and v_B, the node begins receiving information about the network from v_B. Since non-repudiation is not a major concern in this process, we utilize HMAC instead of the digital signature algorithm. We generate HMAC using SHA-256, the symmetric key generated during key encapsulation, and the message:

$$\text{hash} = \text{hmac}(\text{Msg}_{\text{GetInfo}_{g_i}}, \text{sym\_key}_{v,g_i}^{\text{hmac}})$$

In response, g_i receives detailed information about all Gateways and Verifiers in the system, which can be used for future communication. Specifically, the bootstrap node provides all the data it holds about these entities, including their IP addresses, ports, public keys, and other relevant metadata.

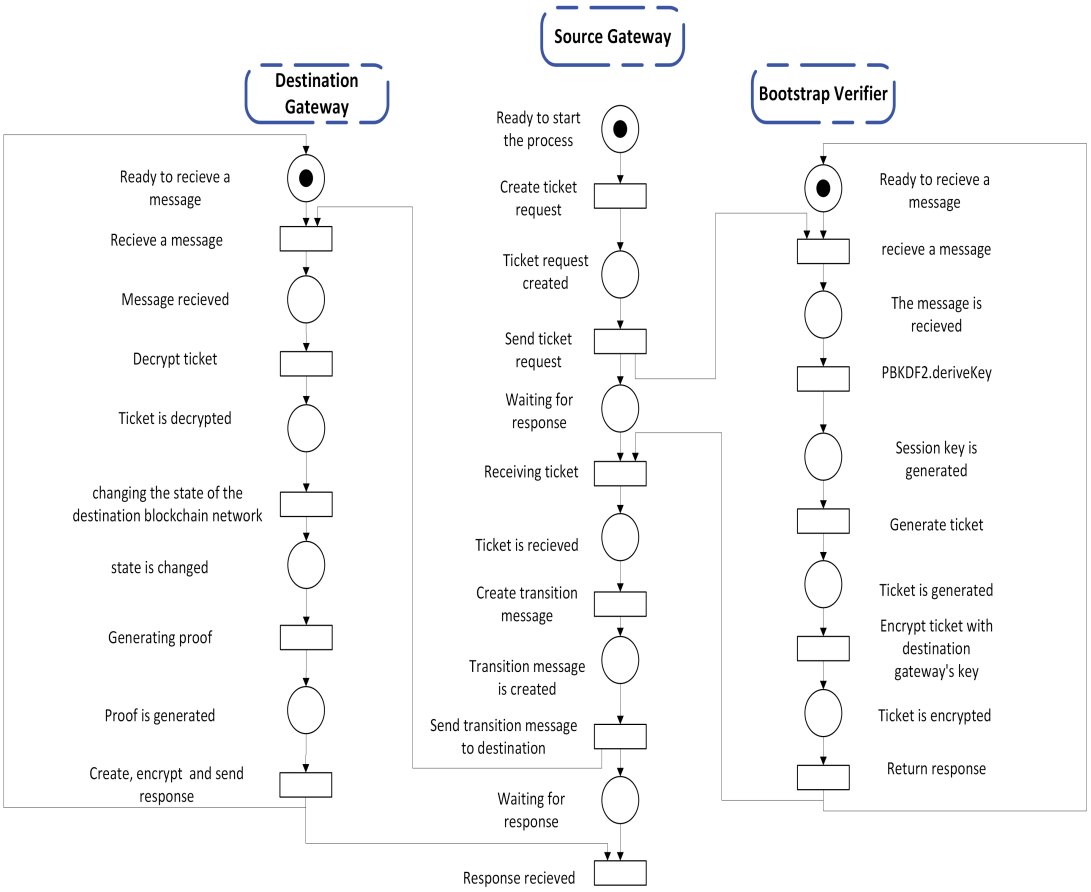

Figure 5.2: Illustration of the cross-chain process from ticket issuance to receiving the proof at the destination.

5.1.3 Cross-chain Interaction

To enable cross-chain interaction between two blockchains, B_i and B_j, their respective Gateways, g_i and g_j, must communicate. Following a Kerberos-inspired mechanism, the process begins with g_i requesting a Verifier to issue a ticket. Upon receiving this request, the Verifier generates a symmetric encryption key, $\text{session\_key}_{g_{ij}}^{\text{enc}}$, to secure communication between the Gateways. This key is derived using the PBKDF2 algorithm as follows:

$$\text{session\_key}_{g_{ij}}^{\text{enc}} = \text{PBKDF2.deriveKey}(\mathit{input}, \mathit{salt}, \mathit{iter}, \mathit{keyLen}, \mathit{hash\_method})$$

Here, *input* is a random string generated by the Verifier to serve as the basis for a temporary session key. *salt* is a cryptographic salt value, *iter* is the number of iterations, *keyLen* specifies the key length, which is set to 32 bytes in our system, and *hash_method* is the hash function used by PBKDF2, for which we use SHA-256.

Subsequently, the Verifier constructs a ticket, denoted as:

$$\text{Ticket} = (\text{session\_key}_{g_{ij}}^{\text{enc}}, \text{pk}_{g_i}^{\text{sig}}, \text{addr}_{g_i}, \text{lifetime})$$

where the *lifetime* parameter defines the validity period of the ticket. The Verifier then encrypts the ticket using the symmetric key previously established between g_j and the verifier, denoted $\text{sym\_key}_{g_j}^{\text{enc}}$, and sends the encrypted ticket, $\text{Ticket}_{\text{sym\_key}_{g_j}^{\text{enc}}}$, back to g_i, along with $\text{session\_key}_{g_{ij}}^{\text{enc}}$ encrypted under $\text{sym\_key}_{g_i}^{\text{enc}}$.

Upon receiving the response, g_i initiate a state transition on B_j by constructing a transition request, $\mathit{trans}_{\mathit{req}}$, which contains the operation details and is encrypted with $\text{session\_key}_{g_{ij}}^{\text{enc}}$. Next, g_i sends the following message:

$$msg_{\text{transition}} = (\mathit{trans}_{\mathit{req}}, \text{Ticket}, \sigma)$$

to g_j. When g_j receives this message, it first decrypts the ticket to obtain $\text{session\_key}_{g_{ij}}^{\text{enc}}$, which is then used to decrypt $\mathit{trans}_{\mathit{req}}$ and extract the public key of g_i for signature verification. Upon validating $\mathit{trans}_{\mathit{req}}$, g_j proceeds with executing the state transition on B_j, resulting in the creation of a response message:

$$msg_{\text{response}} = (trans_{res}, \text{Proof})$$

This response is encrypted using $\text{session\_key}_{g_{ij}}^{\text{enc}}$ and sent back to g_i. This process is illustrated in Figure 5.2.

5.1.4 Proof Mechanism and Trust Management

Upon receiving msg_{response}, Gateway g_i decrypts the message and extracts the proof, which is then forwarded to a subset of Verifiers known as the consensus committee, formally defined as:

$$\mathcal{V}_{\text{cc}} = \{v \in \mathcal{V} \mid \mathcal{TS}_v > T\}$$

Here, $\mathcal{V}_{\text{cc}}$ denotes the set of Verifiers whose trust scores exceed the threshold T, and $\mathcal{V}$ represents the complete set of Verifier nodes in the system. Each member of the consensus committee independently verifies the proof, sends the result back to g_i, and exchanges their validation outcomes with other committee members. This mutual exchange enables dynamic adjustment of trust scores based on observed behavior.

Once g_i collects at least $f + 1$ consistent responses from the committee, it makes a final decision: either to apply the result on B_i, or to initiate a rollback procedure. The latter, depending on the context of the cross-chain operation, may include a compensating action on B_j. This phase of the protocol is illustrated in Figure 5.3.

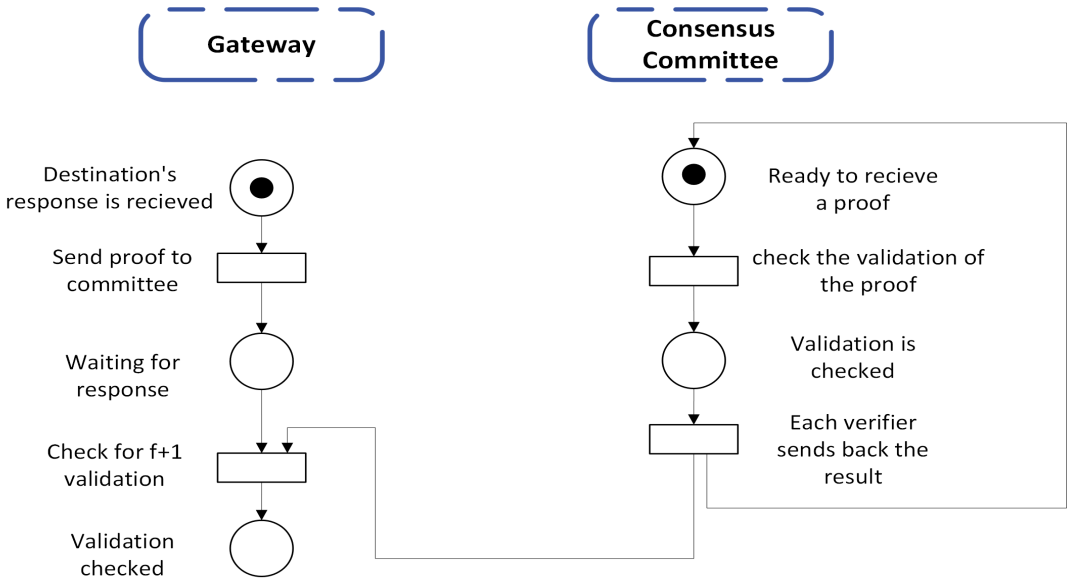

Figure 5.3: Illustration of the consensus mechanism.

5.2 Trust Model

In this section, we introduce our novel trust score calculation model, which extends traditional approaches by incorporating enhancements designed to increase its dynamism and adaptability. These enhancements allow the model to adapt seamlessly to the specific requirements of interoperability across private networks, enabling the implementation of diverse policies for adjusting the trust score $\mathcal{TS}$.

The first improvement involves incorporating the variance of a verifier's verification result history into the trust score calculation. When the variance of the verification results is zero—indicating that the system consistently generates either valid or invalid results—we assign the last verification result, $\phi_t^{v_i}$, as the trust score of the system in epoch t. Each epoch represents the completion of the consensus process and the acquisition of the corresponding verification result. Assigning a high trust

score to nodes in the very first round of operation may seem optimistic, but it is justified in private interoperability networks, where nodes are semi-trusted and added by known participants. Moreover, as shown in subsequent sections, our trust model can significantly reduce the trust score even upon detection of a few malicious actions, thereby effectively mitigating risk. This approach also enables the maintenance of a predefined trust threshold from the outset—something not possible in models that incrementally build trust from zero.

If the verification results exhibit inconsistency (i.e., non-zero variance), we utilize a dynamic trust score derived from conventional models. Consequently, the trust score for epoch t is defined as:

$$\mathcal{TS}_{v_i}(t) = \begin{cases} \phi_t^{v_i} & \text{if Var} = 0 \\ (1-\alpha)\sum_{i=1}^{t} \phi_t^{v_i} \alpha^{(t-i)} & \text{if Var} > 0 \end{cases} \tag{5.1}$$

where we define $\phi_t^{v_i}$, the verification result of Verifier v_i in epoch t, as follows:

$$\phi_t^{v_i} = \begin{cases} 1 & \text{if } v_i \text{ acts benevolently} \\ 0 & \text{if } v_i \text{ acts maliciously} \end{cases} \tag{5.2}$$

We define α as the discount factor, which assigns greater weight to recent activities compared to earlier ones in the calculation of $\mathcal{TS}$. Based on Equation 5.1, the large value of α corresponds to long-term memory, which means that the trust calculation gives significant weight to the entire history of observed behavior. In contrast, a small value of α reflects short-term memory, where the trust calculation emphasizes recent behavior over historical data.

To manage inconsistencies in verification proofs and enforce various policies, we introduce a dynamic discount factor. As mentioned before, lower discount factor prioritizes recent activity in the case of invalid responses, enabling a sharp decline in trust. Conversely, a higher discount factor places more emphasis on historical data, allowing for gradual trust recovery. This dynamic adjustment of the discount factor is computed based on the variance of the verification result history and is defined as:

$$\alpha = e^{-k \cdot \frac{\text{var}}{\text{var}_{\max}}} \tag{5.3}$$

Here, the variance, var, represents the variance of ϕ^{v_i}, while $\text{var}_{\max}$ denotes the maximum possible variance of ϕ^{v_i}. According to Popoviciu's inequality [Egozcue and García, 2018], since ϕ^{v_i} is bounded within the range $[0, 1]$, the maximum variance is computed as:

$$\text{var}_{\max} = \frac{(\phi_{\max} - \phi_{\min})^2}{4} \tag{5.4}$$

Thus, the value of $\text{var}_{\max}$ is computed as:

$$\text{var}_{\max} = \frac{(1-0)^2}{4}$$

The adjustment factor k controls the steepness of changes in the trust score.

Overall, in the presented trust model, recent validation results are prioritized over historical data due to the presence of a discount factor. However, our dynamic design allows the influence of recent results on trust to be modulated, as described by Equation 5.3. When instability occurs—reflected by high variance—the system automatically assigns greater weight to recent data. This influence can be further

fine-tuned using the parameter k , which adjusts the steepness of the variance effect. A higher k intensifies the impact of fluctuations in variance, resulting in a more significant increase or decrease in the trust score.

Chapter 6

Evaluation

To evaluate the proposed system and assess how effectively it addresses the research questions—specifically, the impact of post-quantum cryptography on performance, the usability of the trust mechanism in resisting malicious behavior, and the comparison between the novel trust model and traditional ones—we conducted a comprehensive evaluation based on our implementation of the proposed architecture.

This chapter first outlines the implementation details of the proposed framework, followed by an analysis of the system's performance and security. In the second part of the evaluation, we present and analyze the results related to the novel trust mechanism.

6.1 Implementation

A notary-based interoperability solution was developed to serve as a benchmark for the proposed architecture[1]. The implementation incorporates essential compo-

[1] `https://github.com/tcdt-lab/PQ-NS-IOP`

nents—namely, Verifiers and Gateways—designed in accordance with the system model outlined in the preceding sections. Additionally, we developed two dedicated modules: one for handling various cryptographic operations, including digital signature algorithms (DSA), key encapsulation mechanisms (KEM), HMAC generation and verification, and key derivation; and another for managing the network protocol. To support comparative analysis between different cryptographic approaches, the system is compatible with both post-quantum cryptography and elliptic-curve cryptography for all asymmetric cryptographic operations. In addition to the post-quantum schemes previously discussed, we implemented the P-256 algorithm for digital signatures [Adalier and Teknik, 2015] and X25519 for elliptic-curve Diffie-Hellman key exchange [Bernstein, 2006].

Regarding the technologies used, the system was implemented in Go (Golang). Go was chosen for its built-in support for concurrency, making it particularly well-suited for implementing key components such as message passing, cryptographic operations, and network communication. Its lightweight performance and efficient concurrency model enable the system to handle multiple simultaneous processes effectively—an essential capability in a decentralized and distributed architecture.

For data management, MySQL was used as the primary Relational Database Management System (RDBMS) due to its reliability and structured data handling capabilities. To enhance system performance, Redis database was integrated as a caching layer. Since Redis stores data in memory, it enables extremely fast read and write operations compared to traditional disk-based databases. This speed is particularly beneficial in a distributed system, where frequent access to recurring data—such as

session information, authorization tokens, or commonly used queries—can otherwise introduce latency. By caching such data in Redis, the system significantly reduces response times and computational overhead, improving overall efficiency.

To implement the proof mechanism, we utilized Circom [Bellés-Muñoz et al., 2022] to design and compile zero-knowledge proof circuits. In addition to Circom, we employed SnarkJS [Iden3, 2019] to generate the witness, construct the proof, and produce the corresponding verification key.

Since the cryptographic mechanisms in the proposed architecture operate at a low level, the network functionality relies on raw socket programming. This choice provides the necessary flexibility to implement custom communication protocols from scratch, without the constraints or abstractions imposed by higher-level networking libraries.

6.2 System Evaluation

In this section, we present the results obtained from evaluating the performance of the implemented system, with a particular focus on the impact of different cryptographic approaches. First, we compare the cryptographic methodologies used in the Key Encapsulation Mechanism (KEM). Next, we analyze performance related to message integrity and authentication, specifically evaluating the effectiveness and efficiency of digital signature algorithms and HMAC. Furthermore, we assess the system's performance within the context of the overall interoperability framework. Finally, we evaluate the proposed trust management mechanism to determine its potential benefits for enhancing blockchain interoperability as intended.

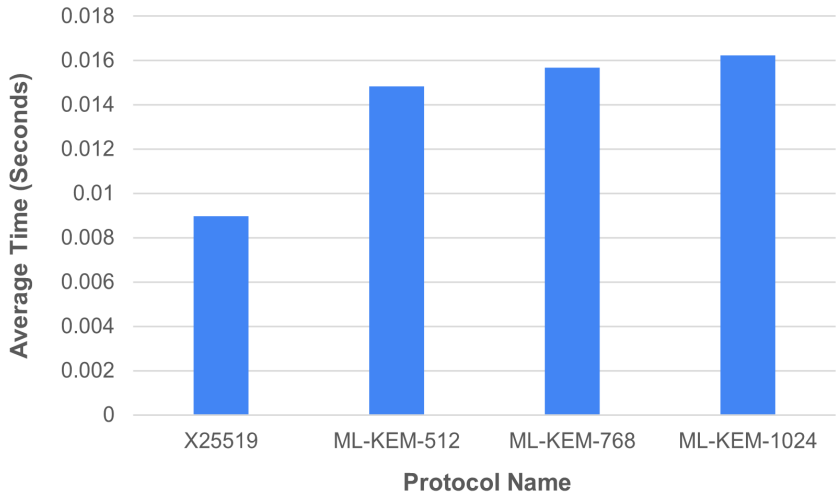

Figure 6.1: Average execution time of key distribution using different cryptographic approaches.

6.2.1 System Configuration

For all evaluation scenarios, we utilized a system equipped with 64 processing units, 128 GB of RAM, and 512 GB of SSD storage. This high-performance setup ensured reliable benchmarking, particularly under parallel load. In this environment, we launched two Gateways and five Verifiers to simulate the proposed architecture and evaluate its behavior under real-world-like conditions.

6.2.2 Key Encapsulation Mechanisms Evaluation

To evaluate the performance of the key encapsulation mechanism (KEM), we conducted experiments using various cryptographic approaches. Initially, we compared different versions of the ML-KEM algorithm. Subsequently, we evaluated ML-KEM against X25519, an elliptic-curve Diffie-Hellman (ECDH) algorithm.

In the first stage, we executed the key distribution process 1,000 times sequentially

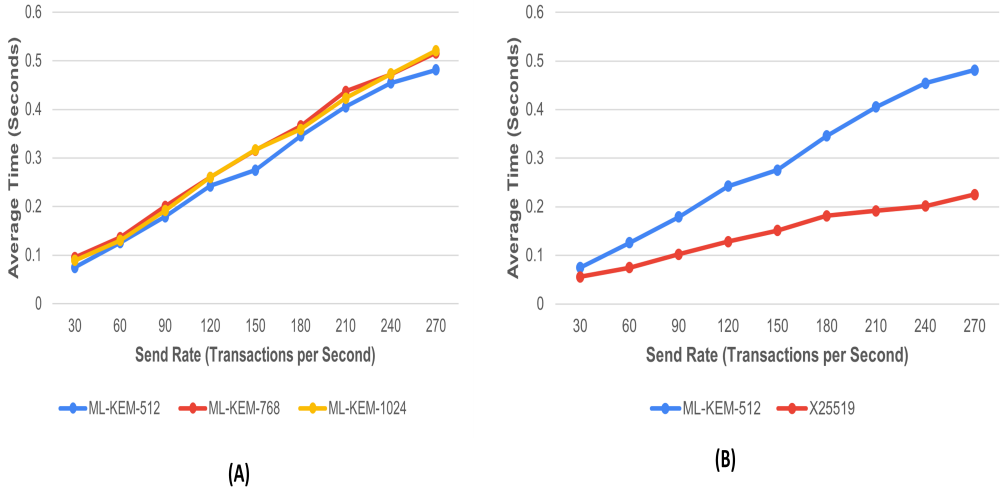

Figure 6.2: (A) Comparison of average key encapsulation time across different versions of ML-KEM. (B) Performance comparison between ML-KEM-512 and X25519, highlighting the impact of underlying cryptographic foundations on efficiency.

for each cryptographic method and calculated the average execution time to obtain a consistent performance metric for a single key encapsulation operation. As shown in Figure 6.1, the results indicate that the ECC-based approach (X25519) demonstrates the lowest average execution time. In contrast, post-quantum approaches (ML-KEM) incur higher computational costs, with execution time increasing proportionally to key size. Among them, ML-KEM-1024 exhibited the highest latency.

As the next step, we evaluated the impact of different cryptographic approaches on system performance under parallel transaction loads to simulate real-world scenarios. In this setup, the key encapsulation mechanism (KEM) was executed in parallel while the load was gradually increased. The experiment started with a sending rate of 10 transactions per second and was gradually scaled up to 270 transactions per second.

Initially, we compared various versions of ML-KEM to analyze the impact of key size on performance, as each version generates keys of different lengths. The results

of this analysis are presented in Figure 6.2 (A), which illustrates that larger key sizes negatively affect performance due to increased bandwidth consumption.

Subsequently, we compared ML-KEM-512 with X25519, as both offer comparable security levels based on fundamentally different mathematical foundations. As illustrated in Figure 6.2 (B), X25519 exhibits significantly better performance than ML-KEM-512. These findings emphasize the role of underlying mathematical models in determining cryptographic efficiency: while lattice-based cryptography offers stronger security against quantum attacks, it introduces more latency compared to elliptic-curve cryptography.

6.2.3 Message Authentication and Integrity Evaluation

To evaluate the performance of the component responsible for message authentication and integrity, we assessed three approaches: post-quantum digital signatures, elliptic curve-based digital signatures, and HMAC. The evaluation was conducted in two distinct scenarios. First, we executed the algorithms sequentially to measure the average performance of individual digital signature or HMAC operations. Second, we executed the same operations in parallel to simulate real-world conditions and assess how each approach performs under concurrent load.

In the sequential scenario, each protocol was executed 1,000 times, and the average execution time was calculated to provide a basis for comparison. The results are presented in Figure 6.3. As expected, HMAC demonstrated the best performance due to its reliance on symmetric cryptography, which is inherently faster than asymmetric approaches. P-256, the elliptic curve-based digital signature algorithm, followed in

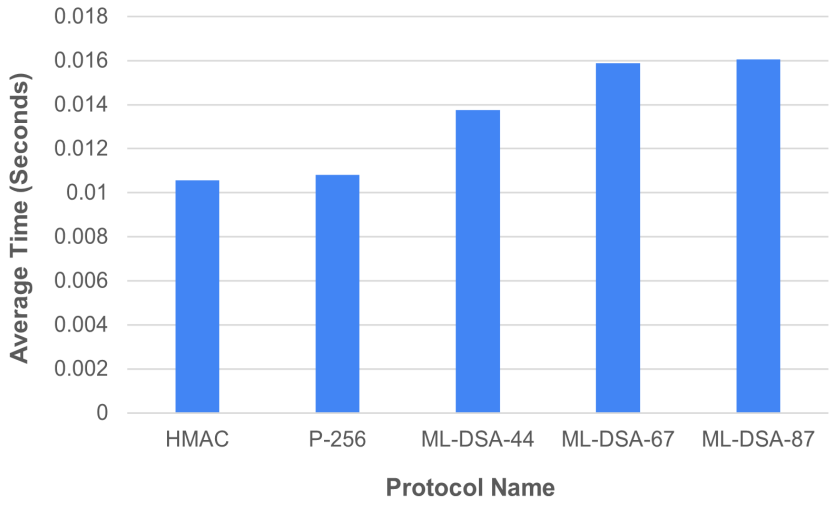

Figure 6.3: Average execution time of different message authentication and integrity mechanisms under sequential execution.

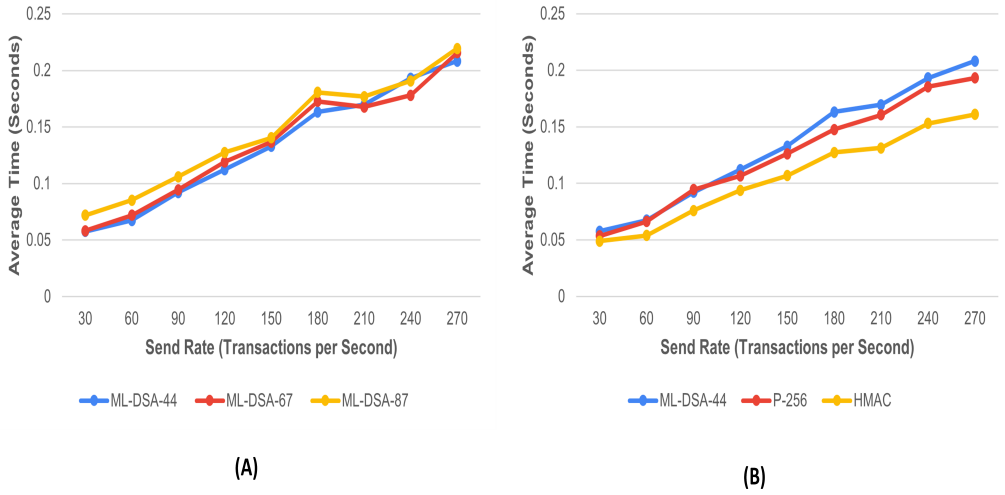

Figure 6.4: Authentication performance under concurrency: (A) shows the effect of key size in ML-DSA variants; (B) compares ML-DSA-44, P-256, and HMAC-SHA256, highlighting HMAC's efficiency.

terms of efficiency. Post-quantum digital signature mechanisms exhibited the lowest performance, with execution times increasing in proportion to key size.

In the parallel scenario, the system was subjected to increasing load by send-

ing transactions concurrently. Starting from 30 parallel transactions per second, the load was gradually increased up to 270 transactions per second. For each level of concurrency, we calculated the average time taken by the server to process a single transaction, aiming to evaluate the system's performance under real-world stress conditions.

As depicted in Figure 6.4(A), the processing time increases with higher send rates, and larger key sizes in post-quantum schemes introduce additional overhead due to bandwidth constraints. However, the underlying cryptographic algorithm used for authentication and validation has a more pronounced impact on performance, as shown in Figure 6.4(B). In this comparison, ML-DSA-44, P-256, and HMAC-SHA256—all offering 128-bit security—were evaluated. HMAC, relying on symmetric cryptography, significantly outperformed the others, demonstrating superior efficiency. While P-256 generally outperformed ML-DSA-44, the performance gap between the two was relatively small compared to the substantial advantage observed with HMAC.

6.2.4 End-to-End Evaluation

In addition to evaluating the cryptographic foundations of our platform, we assessed the overall functionality of the system by analyzing a complete interoperability interaction. In this scenario, a Gateway—referred to as the Source Gateway—first requests a ticket from a Verifier to initiate communication with another Gateway, termed the Destination Gateway. After receiving the ticket, the Source Gateway sends a message along with the encrypted ticket to the Destination Gateway. The Destination Gateway then performs the requested operation, generates a proof, and

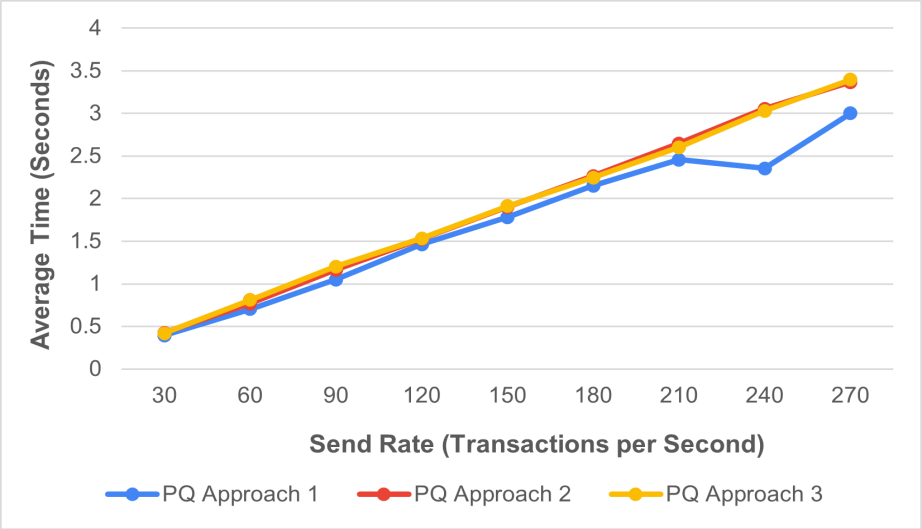

Figure 6.5: Assessment of three post-quantum approaches according to NIST Security Categories: Approach 1 (Category II), Approach 2 (Category III), and Approach 3 (Category V).

returns the result to the Source Gateway. Finally, the Source Gateway initiates the consensus process to validate the result. This evaluation scenario concludes at the end of the consensus phase.

To focus specifically on the performance of the interoperability platform as a generalized solution, we excluded the time taken by on-chain transactions. This is because each blockchain architecture employs different consensus mechanisms, resulting in varying transaction times. Moreover, differences in network deployment—such as testnet configurations and infrastructure conditions—can significantly impact on-chain performance. Including on-chain transaction times in the evaluation would therefore introduce variability unrelated to our system design, making it more difficult to accurately assess the effectiveness of the proposed architecture.

To conduct the assessment, we first compared three post-quantum digital signature schemes under a parallel load, starting at 30 and increasing up to 270 concurrent

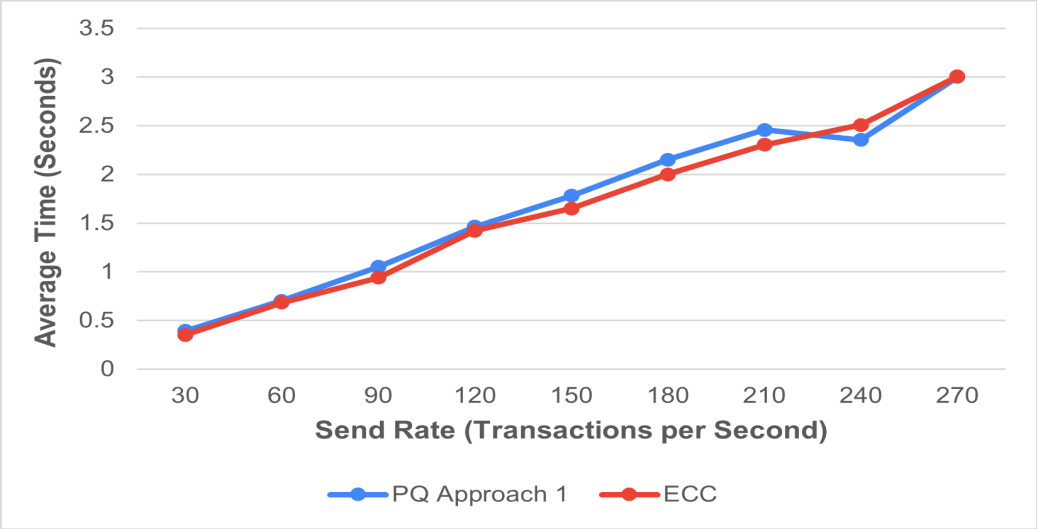

Figure 6.6: Performance comparison between post-quantum approach 1 and elliptic curve approach.

transactions. This scenario compares three distinct post-quantum approaches, each offering a different level of security [Barker, 2020]. Post-Quantum Approach 1 corresponds to NIST Security Category II, Approach 2 aligns with Category III, and Approach 3 achieves the highest level, Category V. Table 6.1 summarizes the characteristics associated with each security category. As illustrated in Figure 6.5, while key size does influence performance, the differences among these post-quantum schemes are minimal.

Security Category	**Strength Equivalent**
I	At least as hard to break as AES-128
II	At least as hard to break as SHA-256
III	At least as hard to break as AES-192
IV	At least as hard to break as AES-192
V	At least as hard to break as AES-256

Table 6.1: Security categories and their approximate strength equivalences.

For the comparison between post-quantum and elliptic curve approaches under the

same parallel scenario, we evaluated a representative post-quantum scheme against an elliptic curve counterpart. According to Figure 6.6, elliptic curve cryptography generally outperformed the post-quantum scheme. However, while the performance gap is more noticeable than among post-quantum variants, it remains relatively minor.

The relatively small performance difference observed between post-quantum and elliptic curve algorithms in the parallel scenario can be attributed to the architectural decision to use symmetric cryptography for the majority of inter-node communication and message encryption tasks. In the proposed system, asymmetric cryptography—whether post-quantum or elliptic curve—is primarily used for authentication and non-repudiation purposes (e.g., digital signatures), whereas actual message encryption and integrity assurance during communication are handled using symmetric algorithms such as AES.

This design choice significantly mitigates the performance impact typically associated with computationally expensive asymmetric cryptographic operations. As a result, even when substituting elliptic curve algorithms with post-quantum alternatives—known to be more demanding in terms of computational overhead—the overall effect on system performance remains relatively limited. This demonstrates that by effectively combining symmetric and asymmetric cryptography in a layered design, the proposed architecture can maintain strong security properties while preserving performance and scalability, even under high transactional loads.

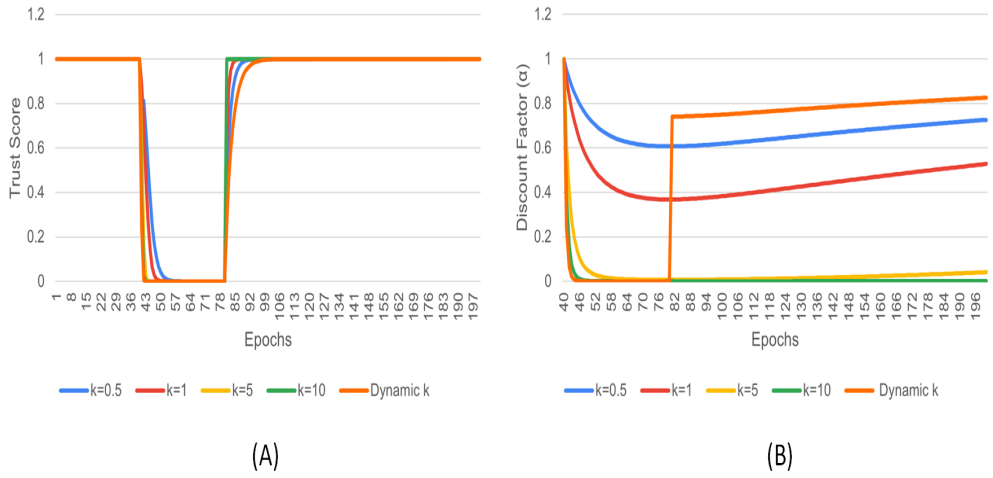

Figure 6.7: Variation of trust score (A) and discount factor (B) across different values of k over epochs.

6.2.5 Trust Model Evaluation

This section presents the evaluation of the proposed trust model and the dynamic behavior of the discount factor in trust score calculation, which represents one of the key novelties of this research. The results are benchmarked against the baseline model introduced by Putra et al. [Putra et al., 2023], which is based on the EigenTrust algorithm [Kamvar et al., 2003].

First, we analyze the effect of varying the parameter k on the trust score $\mathcal{TS}$, examining how the discount factor α evolves dynamically for each value of k. To demonstrate the flexibility of the trust model, we assessed both fixed and dynamic values of k. The dynamic k is designed to mimic the nature of trust in real-life scenarios. In the dynamic k setting, when incorrect validation results are detected, a higher k is assigned, enabling a quicker response to inconsistencies. Conversely, when the Verifier consistently produces correct validation results, a lower k is applied,

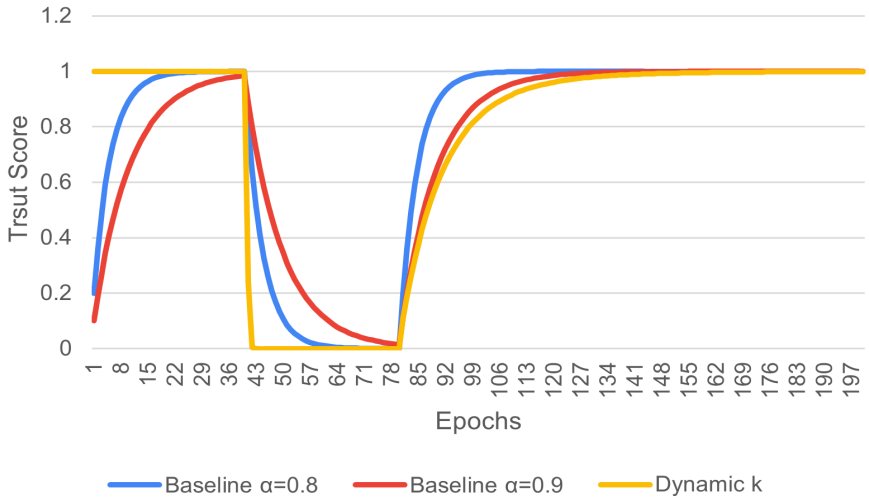

Figure 6.8: Comparison of the baseline model with fixed α values ($\alpha = 0.8$ and $\alpha = 0.9$) against the dynamic k setting.

reflecting the gradual process of rebuilding trust. The evaluation is conducted over 200 epochs, with incorrect transaction verifications deliberately introduced between epochs 40 and 80 to assess the model's response under adversarial conditions.

As illustrated in Figure 6.7, increasing the value of k from 0.5 to 10 results in more pronounced variations in the rate of trust score changes. Specifically, higher values of k lead to steeper rates of both decrease and recovery in trust. The dynamic adjustment of k illustrates a sharp decline in the trust score in response to malicious behavior, while also capturing the more gradual process of trust recovery when the previously malicious node resumes honest behavior.

Regarding the value of α, higher k values lead to a sharper decrease in α, emphasizing recent activities—particularly beneficial during periods of inconsistency. In contrast, lower k values give greater weight to historical data, providing stability as trust is re-established.

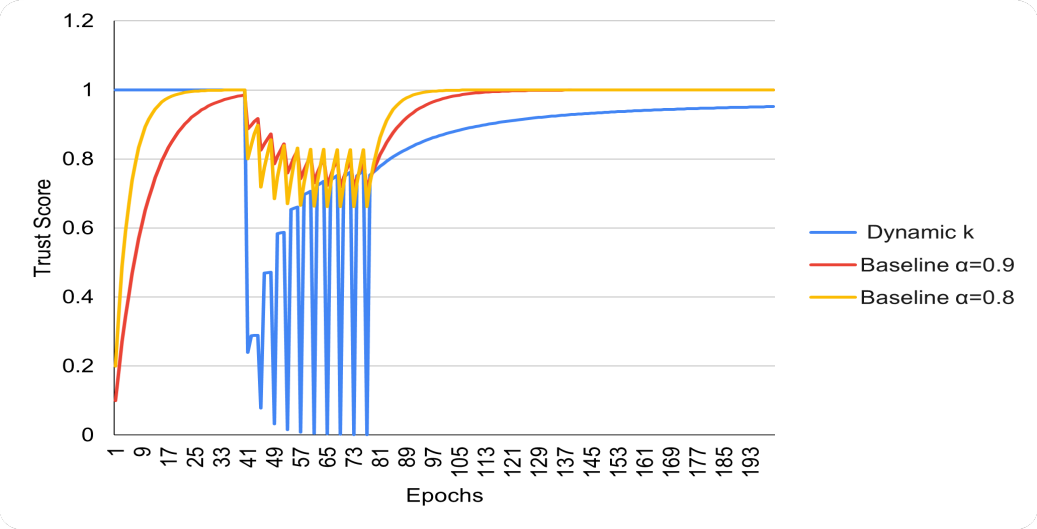

Figure 6.9: Comparison of the baseline model with fixed α values ($\alpha = 0.8$ and $\alpha = 0.9$) and the dynamic k setting in the proposed solution, where v_i exhibits malicious behavior every four epochs between epochs 40 and 80.

Figure 6.8 compares the proposed dynamic approach with the baseline model, using two fixed values for the discount factor (0.8 and 0.9). As illustrated in the figure, the proposed model demonstrates several advantages over the baseline. First, because the trust model initially aligns with the validation results, a threshold can be set early based on the variance of ϕ_{v_i} observed during the initial steps. This enables more precise adjustments from the outset. Moreover, the proposed solution is specifically designed to facilitate interoperability between private networks. In this context, we assume a certain degree of initial trust in the Verifiers, eliminating the need to establish trust entirely from scratch.

Finally, we evaluate our proposed trust model in a scenario where a Verifier behaves maliciously every four epochs between epochs 40 and 80. As illustrated in Figure 6.9, the results demonstrate that the proposed model is more sensitive to sporadic malicious behavior. This heightened responsiveness enables the system to effectively

detect Verifiers attempting to mask their dishonest actions among otherwise valid responses. This experiment highlights the superiority of the proposed dynamic trust model over baseline approaches, particularly in realistic scenarios where malicious nodes intermittently behave honestly to evade detection.

To conclude, our evaluation demonstrates that although post-quantum approaches generally exhibit lower performance compared to conventional cryptographic methods—such as elliptic curve or symmetric cryptography—the performance gap can be narrowed by employing a hybrid strategy that combines symmetric and asymmetric techniques. This enables the effective integration of post-quantum cryptography within blockchain interoperability frameworks.

Furthermore, our novel trust model proves effective in detecting malicious activities. Its dynamic and adaptable nature makes it particularly suitable for a range of interoperability scenarios, as the proposed solution can be reconfigured to accommodate varying trust degradation and recovery conditions.

Chapter 7

Discussion

This chapter provides a critical discussion of the proposed solution. We begin by analyzing its security aspects, including potential attack vectors and the corresponding mitigation strategies. Following this, we revisit and address each of the research questions outlined in the thesis, reflecting on how the proposed architecture meets the stated objectives.

7.1 Security

In this section, we examine various potential attack scenarios and evaluate the robustness of the proposed solution in mitigating these threats. We discuss how the system's design choices—particularly in architecture, protocol design, and trust management—contribute to its overall resilience.

DDoS Attack: The proposed solution operates within a permissioned environment, where network access is restricted to authenticated participants. This signif-

icantly reduces the likelihood of Distributed Denial-of-Service (DDoS) attacks compared to permissionless blockchains. Beyond access control, the use of public key verification ensures that only authorized nodes can engage in network activities, further limiting potential attack vectors. Additionally, the deployment of firewalls and rate-limiting mechanisms can serve as complementary defenses, helping to detect and block abnormal traffic patterns and providing an extra layer of resilience against DDoS attacks.

Man-in-the-Middle (MitM) Attack: Every node in the system is initialized with the public key and IP address of a bootstrap Verifier during system setup. This pre-distributed information allows nodes to validate the authenticity of messages by verifying their origin against trusted public keys. As a result, malicious actors cannot insert themselves between a Verifier and a Gateway to intercept or alter communications.

Misbehaving Notaries: The system leverages the Practical Byzantine Fault Tolerance (PBFT) consensus mechanism to address the risk of misbehaving notaries. As long as the number of faulty nodes remains equal or less than f, malicious actions can be reliably identified and isolated. The proposed trust model further enhances this capability by dynamically adjusting reputation scores and removing compromised nodes from critical roles. Evaluation results confirm that the system effectively mitigates the impact of malicious nodes, preserving network integrity and reliability.

On-Off Attack: The proposed dynamic trust model is specifically designed to counter on-off attacks, in which malicious nodes alternate between correct and malicious behavior to avoid detection. The adjustable parameter k governs the steepness

of trust degradation in response to misbehavior, ensuring that even a single misbehavior results in a substantial trust penalty. This mechanism prevents adversaries from regaining trust through temporary honest behavior and ensures that reputation scores are reduced to a detectable level, limiting the attack's effectiveness.

Sybil Attack: The architecture is inherently resilient to Sybil attacks, where a malicious actor creates multiple pseudonymous identities to manipulate the network and reputation system. This risk is significantly reduced in our setting, as the solution is designed for private interoperability networks involving semi-trusted, pre-approved participants. Verifiers are known to one another prior to deployment, and Gateways receive verified information through a trusted channel established by the bootstrap Verifier. This tightly controlled environment renders the creation and use of fake identities impractical and ineffective.

7.2 Performance

To address the research question regarding the impact of post-quantum cryptography on blockchain interoperability, we observed that adopting post-quantum cryptographic methods can introduce performance overhead. Lattice-based cryptography is generally more computationally demanding than traditional schemes like elliptic curve cryptography. For instance, operations such as polynomial multiplication—especially when using Number Theoretic Transforms (NTT)—can be quite costly [Tan et al., 2021; Stelzer et al., 2025]. Another contributing factor is the typically large key sizes associated with lattice-based cryptography, which are significantly larger than those in conventional systems. These larger keys can increase bandwidth usage and

introduce delays in communication and processing.

However, this performance degradation can be mitigated through hybrid approaches. For example, symmetric cryptography can be employed for encryption tasks, and digital signatures can be replaced with more efficient mechanisms like HMAC when non-repudiation is not required. Importantly, post-quantum cryptography offers substantial security benefits, as it is based on hard mathematical problems such as Learning With Errors. These problems are considered secure even against quantum-capable adversaries, providing enhanced protection and contributing to the overall robustness of the system.

In conclusion, while post-quantum cryptography does introduce performance costs, it offers significant security advantages. Furthermore, with a carefully designed architecture and hybrid cryptographic techniques, the impact on performance can be effectively minimized.

7.3 Trust

As discussed in previous sections, we designed and evaluated a novel trust model in which the discount factor (α) is dynamically adjusted using a parameter denoted as k. This design allows the system to adapt in real time, thereby answering our second research question. By defining the parameter k, the trust model gains the ability to regulate the sensitivity of trust adjustments in response to both malicious and benign behavior. A higher value of k results in more rapid trust degradation following malicious activity, as well as faster trust restoration when nodes behave correctly. Conversely, a lower value of k leads to smoother, more gradual changes in

trust levels.

Since the discount factor is defined as a function of k (as shown in Equation 5.3), the system can modify k at runtime. This flexibility enables system designers to develop policies or algorithms for dynamically tuning k, allowing the trust model to adapt to diverse operational conditions and evolving threat landscapes. The evaluation results confirm the dynamic responsiveness of our proposed model and highlight its advantages over the baseline model, which is based on the EigenTrust algorithm.

These findings directly address our final research question, which concerns the comparative effectiveness of the proposed trust model versus a traditional, fixed-discount-factor model. The results show that the proposed model introduces greater adaptability and is better suited for long-term deployment in dynamic environments. As illustrated in Figure 6.9, it performs especially well in detecting adversaries who attempt to conceal malicious behavior within intermittent honest activity.

However, this enhanced adaptability also comes with increased complexity. Unlike the baseline model, which employs a constant discount factor, the proposed approach defines α as a function of k, introducing additional computational overhead. Nevertheless, the trade-off is justified by the model's superior ability to adapt to varied and evolving trust scenarios.

Chapter 8

Conclusion

This thesis investigated the role of cryptographic primitives in enabling secure and efficient blockchain interoperability, particularly within notary-scheme-based architectures. Specifically, it evaluated the impact of employing different cryptographic approaches—lattice-based post-quantum schemes (e.g., ML-KEM, ML-DSA) and elliptic curve-based counterparts—across three critical phases: key encapsulation, digital signatures, and cross-chain communication. To address emerging trust challenges in distributed environments, the thesis further introduced a novel trust model that dynamically adjusts its parameters based on the observed behavior of a Verifier. This model defines the discount factor as a function of a tunable variable, k, enabling real-time responsiveness to both malicious and cooperative activity. The remainder of this chapter summarizes the core contributions of this work and outlines directions for future research in post-quantum-secure and trust-aware blockchain interoperability.

8.1 Contributions

This thesis addresses two primary challenges in blockchain interoperability: ensuring post-quantum cryptographic resilience in cross-chain communication, and enabling adaptive, trust-aware coordination among decentralized notary nodes. To this end, it contributes (1) the design and empirical evaluation of a quantum-resistant interoperability framework, and (2) a dynamic trust model that enhances resilience against both persistent and intermittent adversarial behavior.

8.1.1 Design, Implementation, and Evaluation of a Decentralized Quantum-Resistant Interoperability Protocol

As our research component, we designed and developed a notary-scheme-based interoperability framework to enable communication between heterogeneous blockchains. This platform served as a benchmark for comparing the performance of lattice-based cryptographic algorithms with elliptic curve cryptography (ECC), with the aim of assessing the impact of different cryptographic choices on blockchain interoperability. Specifically, we utilized various versions of ML-KEM for key encapsulation (ML-KEM-512, ML-KEM-768, and ML-KEM-1024), and ML-DSA for digital signatures (ML-DSA-44, ML-DSA-65, and ML-DSA-87).

To evaluate performance, we compared different configurations of ML-KEM and ML-DSA against ECC-based counterparts. Results indicate that although post-quantum cryptographic algorithms introduce some performance overhead, the use of hybrid models—combining asymmetric and symmetric cryptography—can help de-

sign systems with competitive efficiency. This contribution thus provides the design, implementation, and evaluation of a notary-scheme-based blockchain interoperability protocol that supports quantum-resistant security while maintaining practical performance.

8.1.2 Dynamic Trust Model for Notary-Based Blockchain Interoperability

Our second contribution is the introduction of a novel dynamic trust model. In this model, the discount factor is no longer static but dynamically adjusted based on a tunable variable k. This enables the trust mechanism to adapt in real-time, allowing system architects to fine-tune the impact of recent behavior versus historical actions when evaluating node trustworthiness.

To assess the effectiveness of this model, we analyzed how the dynamic discount factor influences trust computation. Our evaluation demonstrated that system designers can calibrate the sensitivity of trust degradation or restoration depending on the operational context. We then compared our model to a baseline built on the EigenTrust algorithm. The results showed that our dynamic trust model outperforms the baseline in both responsiveness and accuracy. In particular, it is more effective at detecting malicious behavior that is hidden among otherwise honest actions. These findings validate the design of a flexible trust mechanism for blockchain interoperability, offering system designers the ability to adapt trust scoring and node behavior dynamically at runtime.

8.2 Future Work

This thesis presents contributions in secure, quantum-resistant blockchain interoperability and adaptive trust management. Building on these contributions, several research directions remain open for future exploration:

- **Privacy-Preserving Interoperability:** While the current framework ensures message confidentiality through encryption, it does not provide sender anonymity or unlinkability. Future work could integrate privacy-preserving primitives such as ring signatures or source obfuscation techniques (e.g., Clover [Franzoni and Daza, 2022]). Evaluating their performance and compatibility with post-quantum cryptography in interoperability settings would advance both privacy and security guarantees.

- **Interoperability with Public and Semi-Public Blockchains:** This work targets interoperability in permissioned environments. Extending the protocol to support interactions with public or semi-public blockchains introduces new challenges, including trust establishment in open networks, Sybil resistance, and reputation bootstrapping. Adapting the trust model to accommodate such environments—potentially through stake-based or identity-based trust anchors—is a promising direction.

- **Evaluation of Emerging Post-Quantum Cryptographic Schemes:** The protocol currently uses standardized lattice-based schemes (ML-KEM, ML-DSA), but post-quantum cryptography remains an evolving field. Future research could assess alternative primitives (e.g., isogeny-based or multivariate

schemes) and evaluate their implications for cross-chain performance, key management complexity, and resistance to emerging quantum attacks.

- **Dynamic Notary Discovery and Reconfiguration:** The current model assumes a static set of trusted notaries defined at system initialization. To improve scalability and fault tolerance, future work could explore dynamic notary discovery mechanisms—such as decentralized identifiers (DIDs), or on-chain behavior auditing. Extending the trust model to support notary admission, revocation, and rotation based on observed behavior would enable more robust and decentralized system configurations, especially in environments where notaries can be dynamically added or removed.

Bibliography

E. Abebe, D. Behl, C. Govindarajan, Y. Hu, D. Karunamoorthy, P. Novotny, V. Pandit, V. Ramakrishna, and C. Vecchiola. Enabling enterprise blockchain interoperability with trusted data transfer (industry track). In *Proceedings of the 20th international middleware conference industrial track*, pages 29–35, 2019.

M. Adalier and A. Teknik. Efficient and secure elliptic curve cryptography implementation of curve p-256. In *Workshop on elliptic curve cryptography standards*, volume 66, pages 2014–2017. NIST, 2015.

K. M. Alonso et al. Zero to monero. *Zero to monero*, 2020.

E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

A. Augusto, R. Belchior, M. Correia, A. Vasconcelos, L. Zhang, and T. Hardjono. Sok: Security and privacy of blockchain interoperability. *Authorea Preprints*, 2023.

A. Augusto, R. Belchior, M. Correia, A. Vasconcelos, L. Zhang, and T. Hardjono.

Sok: Security and privacy of blockchain interoperability. In *2024 IEEE Symposium on Security and Privacy (SP)*, pages 3840–3865, 2024. doi: 10.1109/SP54263.2024.00255.

R. Bacho and J. Loss. On the adaptive security of the threshold bls signature scheme. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 193–207, 2022.

A. Back, M. Corallo, L. Dashjr, M. Friedenbach, G. Maxwell, A. Miller, A. Poelstra, J. Timón, and P. Wuille. Enabling blockchain innovations with pegged sidechains. *URL: http://www. opensciencereview. com/papers/123/enablingblockchain-innovations-with-pegged-sidechains*, 72:201–224, 2014.

F. Barbàra and C. Schifanella. Mp-htlc: Enabling blockchain interoperability through a multiparty implementation of the hash time-lock contract. *Concurrency and Computation: Practice and Experience*, 35(9):e7656, 2023.

E. Barker. Recommendation for key management: Part 1 – general. NIST Special Publication 800-57 Part 1, Revision 5, National Institute of Standards and Technology (NIST), 2020. URL `https://doi.org/10.6028/NIST.SP.800-57pt1r5`.

A. Batwa and A. Norrman. Blockchain technology and trust in supply chain management: A literature review and research agenda. *Operations and Supply Chain Management: An International Journal*, 14(2):203–220, 2021.

R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia. A survey on blockchain in-

teroperability: Past, present, and future trends. *ACM Computing Surveys (CSUR)*, 54(8):1–41, 2021.

M. Bellés-Muñoz, M. Isabel, J. L. Muñoz-Tapia, A. Rubio, and J. Baylina. Circom: A circuit description language for building zero-knowledge applications. *IEEE Transactions on Dependable and Secure Computing*, 20(6):4733–4751, 2022.

D. J. Bernstein. Curve25519: new diffie-hellman speed records. In *Public Key Cryptography-PKC 2006: 9th International Conference on Theory and Practice in Public-Key Cryptography, New York, NY, USA, April 24-26, 2006. Proceedings 9*, pages 207–228. Springer, 2006.

M. Borkowski, M. Sigwart, P. Frauenthaler, T. Hukkinen, and S. Schulte. Dextt: Deterministic cross-blockchain token transfers. *IEEE access*, 7:111030–111042, 2019.

V. Buterin. Chain interoperability. *R3 research paper*, 9:1–25, 2016.

V. Buterin et al. Ethereum white paper. *GitHub repository*, 1:22–23, 2013.

L. Cao and H. Yang. Two-phase election algorithm for cross-chain notaries. In *2023 3rd International Conference on Computer Science and Blockchain (CCSB)*, pages 53–58. IEEE, 2023.

L. Cao, S. Zhao, Z. Gao, and X. Du. Cross-chain data traceability mechanism for cross-domain access. *The Journal of Supercomputing*, 79(5):4944–4961, 2023.

G. Caronni. Walking the web of trust. In *Proceedings IEEE 9th International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WET ICE 2000)*, pages 153–158. IEEE, 2000.

M. Castro, B. Liskov, et al. Practical byzantine fault tolerance. In *OsDI*, volume 99, pages 173–186, 1999.

L. Chen, Y. Chen, C. Tan, Y. Luo, H. Dou, and Y. Yang. Cross-chain asset trading scheme for notaries based on edge cloud storage. *Journal of Cloud Computing*, 13 (1):90, 2024.

M. Egozcue and L. F. García. The variance upper bound for a mixed random variable. *Communications in Statistics-Theory and Methods*, 47(22):5391–9395, 2018.

L. Ertaul, M. Kaur, and V. A. K. R. Gudise. Implementation and performance analysis of pbkdf2, bcrypt, scrypt algorithms. In *Proceedings of the international conference on wireless networks (ICWN)*, page 66. The Steering Committee of The World Congress in Computer Science, Computer . . . , 2016.

Y. Fan and J. Li. Cross-chain transaction of notaries based on two-stage improved pagerank algorithm. *International Journal of Network Security*, 27(1):95–103, 2025.

N. FIPS. 203: Module-lattice-based key-encapsulation mechanism standard. *National Institute of Standards and Technology: Gaithersburg, MD, USA*, 2024a.

N. FIPS. 204: Module-lattice-based digital signature standard. *National Institute of Standards and Technology: Gaithersburg, MD, USA*, 2024b.

F. Franzoni and V. Daza. Clover: An anonymous transaction relay protocol for the bitcoin p2p network. *Peer-to-Peer Networking and Applications*, pages 1–14, 2022.

P. Frauenthaler, M. Sigwart, C. Spanring, and S. Schulte. Testimonium: A cost-efficient blockchain relay. *arXiv preprint arXiv:2002.12837*, 2020.

A. Geraci. Ieee standard computer dictionary: a compilation of ieee standard computer glossaries. ieee std. 610, 1–217, 1991.

S. Ghaemi, S. Rouhani, R. Belchior, R. S. Cruz, H. Khazaei, and P. Musilek. A pub-sub architecture to promote blockchain interoperability. *arXiv preprint arXiv:2101.12331*, 2021.

J. Golosova and A. Romanovs. The advantages and disadvantages of the blockchain technology. In *2018 IEEE 6th workshop on advances in information, electronic and electrical engineering (AIEEE)*, pages 1–6. IEEE, 2018.

J. Groth. On the size of pairing-based non-interactive arguments. In *Advances in Cryptology–EUROCRYPT 2016: 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II 35*, pages 305–326. Springer, 2016.

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

H. Guo and X. Yu. A survey on blockchain technology and its security. *Blockchain: research and applications*, 3(2):100067, 2022.

Z. Guo and X. Hu. Calculation and selection scheme of node reputation values for notary mechanism in cross-chain. *The Journal of Supercomputing*, pages 1–22, 2024.

C. G. Harris. Cross-chain technologies: Challenges and opportunties for blockchain

interoperability. In *2023 IEEE International Conference on Omni-layer Intelligent Systems (COINS)*, pages 1–6. IEEE, 2023.

T. Haugum, B. Hoff, M. Alsadi, and J. Li. Security and privacy challenges in blockchain interoperability-a multivocal literature review. In *Proceedings of the 26th International Conference on Evaluation and Assessment in Software Engineering*, pages 347–356, 2022.

A. Holcomb, G. Pereira, B. Das, and M. Mosca. Pqfabric: a permissioned blockchain secure from both classical and quantum attacks. In *2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9. IEEE, 2021.

X. Hu, Y. Ling, J. Hua, Z. Dong, Y. Sun, and J. Qi. A blockchain cross-chain transaction method based on decentralized dynamic reputation value assessment. *IEEE Transactions on Network and Service Management*, 2024.

Iden3. Snarkjs: Javascript library for zk-snarks, 2019. URL `https://github.com/iden3/snarkjs`. Accessed: 2025-04-28.

S. D. Kamvar, M. T. Schlosser, and H. Garcia-Molina. The eigentrust algorithm for reputation management in p2p networks. In *Proceedings of the 12th international conference on World Wide Web*, pages 640–651, 2003.

N. Kannengießer, M. Pfister, M. Greulich, S. Lins, and A. Sunyaev. *Bridges between islands: Cross-chain technology for distributed ledger technology.* 2020.

S. Karumba, R. Jurdak, S. S. Kanhere, and S. Sethuvenkatraman. Bailif: A blockchain agnostic interoperability framework. In *2023 IEEE International Conference on*

Blockchain and Cryptocurrency (ICBC), pages 1–9, 2023. doi: 10.1109/ICBC56567.2023.10174967.

S. Khan, M. B. Amin, A. T. Azar, and S. Aslam. Towards interoperable blockchains: A survey on the role of smart contracts in blockchain interoperability. *IEEE Access*, 9:116672–116691, 2021.

K. E. Khorasani, S. Rouhani, R. Pan, and V. Pourheidari. Automated gateways: A smart contract-powered solution for interoperability across blockchains. In *2024 IEEE International Conference on Blockchain (Blockchain)*, pages 611–618, 2024. doi: 10.1109/Blockchain62396.2024.00090.

N. Koblitz. Elliptic curve cryptosystems. *Mathematics of computation*, 48(177):203–209, 1987.

S.-S. Lee, A. Murashkin, M. Derka, and J. Gorzny. Sok: Not quite water under the bridge: Review of cross-chain bridge hacks. In *2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–14. IEEE, 2023.

S. D. Lerner. Rsk white paper. `https://rootstock.io/rsk-white-paper-updated.pdf`, 2019. Accessed: March 2024.

W. Li, A. Sforzin, S. Fedorov, and G. O. Karame. Towards scalable and private industrial blockchains. In *Proceedings of the ACM workshop on blockchain, cryptocurrencies and contracts*, pages 9–14, 2017.

Z. Liu, Y. Xiang, J. Shi, P. Gao, H. Wang, X. Xiao, B. Wen, and Y.-C. Hu. Hyperservice: Interoperability and programmability across heterogeneous blockchains. In

Proceedings of the 2019 ACM SIGSAC conference on computer and communications security, pages 549–566, 2019.

A. Lohachab, S. Garg, B. Kang, M. B. Amin, J. Lee, S. Chen, and X. Xu. Towards interconnected blockchains: A comprehensive review of the role of interoperability among disparate blockchains. *ACM Computing Surveys (CSUR)*, 54(7):1–39, 2021.

Loom Team. Intro to Loom Network. `https://loomx.io/developers/en/intro-to-loom.html`, 2022. Accessed: March 2024.

V. Lyubashevsky. Basic lattice cryptography: The concepts behind kyber (ml-kem) and dilithium (ml-dsa). *Cryptology ePrint Archive*, 2024.

M. A. Manolache, S. Manolache, and N. Tapus. Decision making using the blockchain proof of authority consensus. *Procedia Computer Science*, 199:580–588, 2022.

Monika and R. Bhatia. Interoperability solutions for blockchain. In *2020 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE)*, pages 381–385, 2020. doi: 10.1109/ICSTCEE49637.2020.9277054.

S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Decentralized business review*, 2008.

R. M. Needham and M. D. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21(12):993–999, 1978.

B. C. Neuman and T. Ts'o. Kerberos: An authentication service for computer networks. *IEEE Communications magazine*, 32(9):33–38, 1994.

C. T. Nguyen, D. T. Hoang, D. N. Nguyen, D. Niyato, H. T. Nguyen, and E. Dutkiewicz. Proof-of-stake consensus mechanisms for future blockchain networks: Fundamentals, applications and opportunities. *IEEE Access*, 7:85727–85745, 2019. doi: 10.1109/ACCESS.2019.2925010.

J. Nick, A. Poelstra, and G. Sanders. Liquid: A bitcoin sidechain. `https://blockstream.com/assets/downloads/pdf/liquid-whitepaper.pdf`, 2020.

C. Paar, J. Pelzl, and T. Güneysu. Post-quantum cryptography. In *Understanding Cryptography: From Established Symmetric and Asymmetric Ciphers to Post-Quantum Algorithms*, pages 379–463. Springer, 2024.

L. Page. The pagerank citation ranking: Bringing order to the web. Technical report, Technical Report, 1999.

M. Petkus. Why and how zk-snark works. *arXiv preprint arXiv:1906.07221*, 2019.

G. D. Putra, V. Dedeoglu, S. S. Kanhere, R. Jurdak, and A. Ignjatovic. Trust-based blockchain authorization for iot. *IEEE Transactions on Network and Service Management*, 18(2):1646–1658, 2021.

G. D. Putra, V. Dedeoglu, S. S. Kanhere, and R. Jurdak. Privacy-preserving trust management for blockchain-based resource sharing in 6g-iot. In *2023 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, pages 1–9. IEEE, 2023.

S. Rao, D. Mahto, D. K. Yadav, and D. Khan. The aes-256 cryptosystem resists quantum attacks. *Int. J. Adv. Res. Comput. Sci*, 8(3):404–408, 2017.

K. Ren, N.-M. Ho, D. Loghin, T.-T. Nguyen, B. C. Ooi, Q.-T. Ta, and F. Zhu. Interoperability in blockchain: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(12):12750–12769, 2023.

R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

E. J. Scheid, T. Hegnauer, B. Rodrigues, and B. Stiller. Bifröst: a modular blockchain interoperability api. In *2019 IEEE 44th Conference on Local Computer Networks (LCN)*, pages 332–339, 2019. doi: 10.1109/LCN44214.2019.8990860.

L. Sharma and A. Mishra. Analysis of crystals-dilithium for blockchain security. In *2021 2nd International Conference on Secure Cyber Computing and Communications (ICSCCC)*, pages 160–165. IEEE, 2021.

P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM review*, 41(2):303–332, 1999.

A. Singh, K. Click, R. M. Parizi, Q. Zhang, A. Dehghantanha, and K.-K. R. Choo. Sidechain technologies in blockchain networks: An examination and state-of-the-art review. *Journal of Network and Computer Applications*, 149:102471, 2020.

T. Stelzer, F. Oberhansl, J. Schupp, P. Karl, and H. Turcuman. Extended version: enabling lattice-based post-quantum cryptography on the opentitan platform. *Journal of Cryptographic Engineering*, 15(2):1–23, 2025.

W. Tan, B. M. Case, A. Wang, S. Gao, and Y. Lao. High-speed modular multiplier

for lattice-based cryptosystems. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 68(8):2927–2931, 2021.

Z. S. Tan, C. H. Chen, X. B. Chen, and Y. Xin. A study on privacy protection of cross-chain transactions based on improved notary mechanisms. *IEEE Access*, 2024.

A. Team. Axelar network: connecting applications with blockchain ecosystems, 2021.

J. M. Turner. The keyed-hash message authentication code (hmac). *Federal Information Processing Standards Publication*, 198(1):1–13, 2008.

F. Victor and B. K. Lüders. Measuring ethereum-based erc20 token networks. In *Financial Cryptography and Data Security: 23rd International Conference, FC 2019, Frigate Bay, St. Kitts and Nevis, February 18–22, 2019, Revised Selected Papers 23*, pages 113–129. Springer, 2019.

V. Vishnumurthy, S. Chandrakumar, and E. G. Sirer. Karma: A secure economic framework for peer-to-peer resource sharing. In *Workshop on Economics of Peer-to-peer Systems*, volume 35, 2003.

F. Wang, S. Cohney, and J. Bonneau. Sok: Trusted setups for powers-of-tau strings. *Cryptology ePrint Archive*, 2025.

G. Wang, Q. Wang, and S. Chen. Exploring blockchains interoperability: A systematic survey. *ACM Computing Surveys*, 55(13s):1–38, 2023.

H. Wang, Y. Cen, and X. Li. Blockchain router: A cross-chain communication proto-

col. In *Proceedings of the 6th international conference on informatics, environment, energy and applications*, pages 94–97, 2017.

Z. Wang, J. Li, X.-B. Chen, and C. Li. A secure cross-chain transaction model based on quantum multi-signature. *Quantum Information Processing*, 21(8):279, 2022.

G. Wood. Polkadot: Vision for a heterogeneous multi-chain framework. *White paper*, 21(2327):4662, 2016.

X. Wu, T. Zhang, J. Wang, J. Cheng, and Z. Wang. A distributed cross-chain mechanism based on notary schemes and group signatures. *Journal of King Saud University-Computer and Information Sciences*, 35(10):101862, 2023.

D. Yaga, P. Mell, N. Roby, and K. Scarfone. Nistir 8202 blockchain technology overview. *National Institute of Standards and Technology, US Department of Commerce, Washington, USA*, 2018.

Z. Yang, K. Yang, L. Lei, K. Zheng, and V. C. Leung. Blockchain-based decentralized trust management in vehicular networks. *IEEE internet of things journal*, 6(2): 1495–1505, 2018.

H. Yi. A post-quantum blockchain notary scheme for cross-blockchain exchange. *Computers and Electrical Engineering*, 110:108832, 2023.

A. Zamyatin, D. Harz, J. Lind, P. Panayiotou, A. Gervais, and W. Knottenbelt. Xclaim: Trustless, interoperable, cryptocurrency-backed assets. In *2019 IEEE symposium on security and privacy (SP)*, pages 193–210. IEEE, 2019.

M. Zhang, X. Zhang, J. Barbee, Y. Zhang, and Z. Lin. Sok: Security of cross-chain bridges: Attack surfaces, defenses, and open problems. *arXiv preprint arXiv:2312.12573*, 2023.

Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang. An overview of blockchain technology: Architecture, consensus, and future trends. In *2017 IEEE international congress on big data (BigData congress)*, pages 557–564. Ieee, 2017.

Y. Zuo, M. Yang, Z. Qiang, D. Fei, X. Shao, S. Su, Q. Mo, and Z. Liang. A review of cross-blockchain solutions. In *International Conference on Cloud Computing*, pages 221–233. Springer, 2021.