Towards an Internet of Blockchains: Decentralized Discovery Services with Layer 2 Ceremonies

by

Khalid Hassan

A Thesis submitted to the
Faculty of Graduate and Postdoctoral 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
October 2025

Thesis advisor

Sara Rouhani

Author

Khalid Hassan

Towards an Internet of Blockchains: Decentralized Discovery Services with Layer 2 Ceremonies

Abstract

The rise of heterogeneous blockchain platforms that are utilized by various organizations depending on their needs has created an interoperability problem. While significant progress has been made in enabling interoperability between blockchain networks, most works in this area assume that these networks are already aware of each other. This thesis makes two key contributions to fill this gap. First, it introduces a decentralized architecture for inter-blockchain network and asset discovery that allows domain registration and resolution without dependence on a central authority. The system is self-sufficient, scalable, and enables dynamic discovery across blockchain networks. Second, it presents a decentralized mechanism for generating trust anchors through multi-party cryptography ceremonies. By increasing the number of participants constructing Common Reference Strings (CRS) and periodically regenerating it, the mechanism enhances the security of zero-knowledge systems while maintaining their decentralization. The evaluation of both implementations shows that the discovery architecture achieves high throughput under heavy load and scales efficiently with the number of participants. Similarly, the trust anchor generation mechanism performs resilient CRS generation ceremonies within bounded durations.

Preface

The research work presented in this thesis has been conducted in the Trustworthy Computing and Distributed Technologies (TCDT) Lab lead by Dr. Sara Rouhani. This thesis is an original work by Khalid Hassan.

Portions of the work have appeared in peer-reviewed venues. The contributions on decentralized inter-blockchain discovery were accepted for publication as Khalid Hassan, Amirreza Sokhankhosh and Sara Rouhani. "Enabling Blockchain Interoperability Through Network Discovery Services," to appear in 2025 IEEE 7th International Conference on Decentralized Applications and Infrastructures (DAPPS), where the paper received the **IEEE DAPPS Best Paper Award**.

The contributions on decentralized trust anchor generation have been submitted for review as Khalid Hassan and Sara Rouhani. "Decentralized Common Reference String Generation via Layer 2 Rollups," to the 2025 IEEE/WIC International Conference on Web Intelligence and Intelligent Agent Technology.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The rise of blockchain technology over the past decade has been driven by advancements in distributed systems, cryptographic techniques, and decentralized consensus mechanisms. Blockchain was introduced as the underlying infrastructure for Bitcoin[2]. Its security, decentralization, and ability to coordinate among nodes through robust consensus mechanisms have been empirically proven. Since its initial application in cryptocurrency, blockchain has seen growing adoption in other sectors—such as finance [3], healthcare [4], agriculture [5], and supply chain management—driven by the development of smart contracts [6]. The appeal of blockchain across various fields stems from its unique features, including network-specific protocols for data exchange, governance models for decision-making, consensus mechanisms for achieving agreement among nodes, and transaction finality models that ensure the permanence of validated transactions. However, as blockchain adoption expands, new challenges emerge; most notably, the difficulty of enabling communication between different blockchains. Although these features offer substantial benefits, they are

implemented differently across networks, contributing to the challenge of blockchain interoperability and often resulting in fragmented data silos [7].

Blockchain interoperability has been the focus of extensive research, with numerous solutions proposed to address the challenges of enabling communication across different blockchain networks. This thesis aims to advance the vision of an Internet of Blockchains (IoB), where networks can communicate seamlessly. While most existing solutions emphasize cross-chain transactions, realizing the IoB vision also requires functionalities such as data sharing and service discovery across chains [8]. Projects like Polkadot [9] and Cosmos [10] have made significant strides in facilitating cross-chain communication. However, they overlook a critical requirement: dynamic and decentralized network discovery. Current interoperability approaches rely on static, pre-configured lists of known networks [11], typically maintained by centralized authorities or updated manually [12, 13, 14]. The absence of a decentralized and dynamic discovery service remains a major obstacle to achieving blockchain interoperability at a scale that truly enables the IoB.

The discovery mechanism required to achieve IoB must incorporate a trust anchor to validate the records it stores. In Web2 DNS systems, trust anchors typically take the form of centralized certificate authorities that verify domain ownership. This centralization can be avoided by using zero-knowledge proofs (ZKPs), generated through multi-party computation (MPC). While blockchains can facilitate this process, they continue to face the inherent trade-offs among scalability, security, and decentralization; commonly referred to as the Blockchain Trilemma [15]. In this context, security ensures protection against malicious actors, and decentralization enables trustless,

censorship-resistant interactions. However, scalability remains the most persistent challenge. For example, Ethereum [16] supports a maximum throughput of approximately 15 transactions per second (TPS), and Bitcoin handles only around 7 TPS, despite its widespread adoption. This limited throughput often results in high gas fees and long confirmation times during peak usage. In contrast, centralized payment systems like Visa regularly process about 1,700 TPS, with a theoretical capacity exceeding 24,000 TPS [17], underscoring the significant scalability gap between blockchain-based and conventional infrastructures.

To address the scalability constraints of blockchain networks—commonly referred to as Layer 1 (L1) networks—Layer 2 (L2) solutions, particularly rollups, have emerged as a promising approach. Rollups execute transactions off-chain, batch them, and periodically commit the results to the underlying L1 network. There are two main types of rollups: 1. Optimistic rollups which rely on fraud proofs 2. Zero knowledge rollups which leverage ZKPs. However, both approaches suffer from a critical flaw: the presence of centralized sequencers that control transaction ordering. This centralization introduces single points of failure and requires trusting a single entity—contrary to blockchain's core principle of minimizing trust. Such centralization is particularly unsuitable for protocols that require distributed trust, such as decentralized Common Reference String (CRS) generation ceremonies like Powers of Tau [18], which are essential for ZKP generation.

1.1 Research Questions

This thesis revolves around answering the following research questions:

1. How can a scalable and decentralized discovery mechanism be designed to support dynamic inter-blockchain discovery at scale?

2. What incentive mechanisms can effectively discourage free-riding and ensure active participation in maintaining the discovery infrastructure?

3. How can trust anchors for domain validation be decentralized and secured against manipulation or central points of failure?

1.2 Methodology

This thesis addresses the challenges with inter-blockchain network discovery and secure yet decentralized trust anchors through a systematic approach. First, the design goals are established, followed by an outline of the system's general workflow. Next, the assumed threat model is described, and finally, the evaluation criteria are discussed.

1.2.1 Design Goals

To appropriately address the aforementioned challenges while accommodating the requirements imposed by the inherent characteristics of blockchain networks, the system must achieve the following:

1. **Decentralization:** The discovery mechanism and trust anchor scheme must be decentralized to eliminate single points of failure, governance bias, and censorship.

2. **Scalability:** The discovery mechanism must support a large volume of concurrent requests from diverse sources while managing the storage of discovery records. Similarly, trust anchor generation must be efficient to avoid adverse effects on the network's performance.

3. **Dynamicity:** The discovery system must support dynamic updates of network and service records stored within, while trust anchor generation must tolerate participant rotation, maintaining flexibility when they connect and disconnect.

4. **Collusion Resistance:** The system must prevent the compromise of trust anchors through long-term coordination between participants to undermine its security.

5. **Self-sufficiency:** The system must incentivize sustained participation in network maintenance and participant honesty in trust anchor generation. Correspondingly, it must disincentivize malicious actions of participants.

1.2.2 System Workflow

Discovery Mechanism

The discovery mechanism functions as follows:

1. **Network Registration:** The system implements a voluntary opt-in model where blockchain networks can register to be discoverable by the architecture and can deregister without restrictions. The networks will send requests to the mechanism with the identifier they are attempting to claim and a list of participants that will be used to maintain the discovery mechanism.

2. **Asset Registration:** A network can register its assets for discovery by sending a request to the discovery mechanism. The system then ensures that the identifier associated with the network registering asset is active, then proceeds with asset registration.

3. **Network and Asset Discovery:** The system can be queried using network identifiers. It then returns the network information associated with the identifier and allows users to query registered assets associated with it.

4. **Incentive Mechanism:** The networks are incentivized to maintain the discovery mechanism through transaction fees associated with the actions taken on it. Moreover, the mechanism only allows networks to be discoverable if they provide participants who actively connect to and maintain the discovery architecture.

Trust Anchor Generation

The process of generating secure trust anchors is as follows:

1. **Offloading and Generation:** The system employs a decentralized set of participants that are not part of the discovery mechanism to generate the trust anchor while avoiding performance degradation.

2. **Rotation and Regeneration:** A new trust anchor is periodically generated, rotating the set of participants. The rotation of the participants is collectively decided by the system and is random to disallow prediction of the chosen participants.

1.2.3 Threat Model

The system assumes that networks work in their own interest. This means that networks ideally register themselves for discovery without contributing back to the architecture. This behavior is discouraged through transaction fee incentives and mitigated by mandating resource contribution through maintainers. Furthermore, the system assumes that adversaries will collude in the long-term to compromise the security of the trust anchor. This is mitigated by decentralizing its generation and periodically regenerating it through a decentralized, randomly rotating committee.

1.2.4 Evaluation Criteria

Since the discovery mechanism is expected to receive a large volume of requests, it will be evaluated on its throughput. Specifically, the mechanism will be evaluated on how its resolution latency varies as the concurrent request load increases.

Similarly, the trust anchor generation scheme will be evaluated on the throughput that its infrastructure can handle. Furthermore, the resilience of the system against failures and malicious actors, along with trust anchor regeneration latency, will be assessed.

1.3 Contributions

The contributions of this thesis are as follows:

1. Proposes and implements the first functional DNS-like infrastructure for inter-blockchain discovery, addressing a critical gap in decentralized blockchain in-

teroperability.

2. Introduces a novel scheme for both blockchain network and service discovery, improving the accessibility and utility of decentralized networks.

3. Designs and implements a robust incentive mechanism that promotes active participation in maintaining the discovery infrastructure, discouraging free-riding behavior.

4. Develops a custom ZK-Rollup architecture featuring a fully decentralized sequencer network coordinated through Practical Byzantine Fault Tolerance (PBFT), mitigating centralization risks in existing rollup solutions.

5. Proposes and integrates a decentralized CRS generation scheme within the Layer 2 rollup. The scheme supports both smart contract–based and peer-to-peer coordination, enabling periodic regeneration of compact CRS instances through rotating multiparty sequencer contributions—eliminating long-term trust assumptions and enhancing collusion resistance.

6. Conducts extensive empirical evaluations of both systems, measuring discovery resolution latency, query throughput, rollup transaction capacity, and CRS generation reliability under adversarial scenarios.

Chapter 2

Related Works

Although still evolving, blockchain interoperability has already attracted extensive research attention due to its transformative potential. This chapter reviews foundational works in blockchain systems, interoperability mechanisms, decentralized discovery services, Layer 2 scalability solutions, and zero-knowledge proof techniques.

2.1 Blockchain

Since its introduction by Satoshi Nakamoto in the form of Bitcoin [2], blockchain technology has been the subject of extensive research, resulting in numerous evolutions. Following the success of Bitcoin, other blockchain networks were launched with distinct focus areas, such as Ethereum [16] and Cardano [19] which both support smart contracts using their own scripting languages, Solana [20] which has gained popularity for non-fungible token (NFT) applications; and Polkadot [9] and Cosmos [10], which both focus on enabling interoperability between different blockchains.

Although the focus of each of the aforementioned blockchains is different, they all share a common characteristic: open participation. These networks are referred to as *public or permissionless blockchain*. In contrast, *private or permissioned* and *consortium* blockchains [21, 22] restrict participation to authorized entities. Since public blockchains allow open participation, they must enforce stronger consensus mechanisms to deter malicious behavior.

This can be done by setting an artificial barrier-to-entry as is the case in Proof of Work (PoW) [2], where nodes must compute a hash of the block header—adjusting the transaction order and nonce—until the resulting hash meets a target difficulty. The result is then verified by other participants in the network. Proof of Stake (PoS) variants, on the other hand, require participants to stake a portion of their tokens as collateral [23, 24]. These tokens can be staked either by the node itself or via delegation, as in Delegated Proof of Stake (DPoS). Malicious behavior results in *slashing* of the staked tokens, creating a strong economic disincentive. The greater the stake, the higher the chance of being selected to propose a block, thereby increasing the network's implicit trust in that node. While these mechanisms enhance security in public blockchains, they come with drawbacks—such as the energy inefficiency of PoW and the centralization risks of PoS due to large stakeholders.

Permissioned and consortium blockchains differ significantly from public blockchains. Both incorporate identity management to restrict participation to authorized nodes. Consortium blockchains extend this by allowing multiple organizations to govern the network, introducing a degree of decentralization—albeit not full [25]. Several platforms platforms support the development of permissioned blockchains, including Hy-

perledger Fabric [26], Substrate [27], Corda [28], Quorum [29], and Tendermint [10]. These platforms cater to different use cases through their unique design choices—for instance, Fabric's channels for private communication, Substrate's modular pallets for customizable runtimes, Corda and Quorum's transaction-level privacy, and Tendermint's instant finality and interoperability support.

Because participants in permissioned blockchains are known and vetted, these systems often adopt lighter consensus protocols. Fabric, Corda, and Quorum, for example, support RAFT [30], which is a leader-follower consensus algorithm where nodes trust the leader. Substrate uses Authority Round (AURA) [31] or Blind Assignment for Blockchain Extension (BABE) [32] for its block production and GHOST-based Recursive ANcestor Deriving Prefix Agreement (GRANDPA) [33] for finality. Both AURA and BABE require a predetermined set of nodes to be selected as validators, where the former rotates through all nodes in the set giving each a turn in block production, and the latter grants nodes block production rights if they win the current slot based on a Verifiable Random Function (VRF). GRANDPA is a deterministic finality gadget which utilizes validators voting for the best chain, and finalizes chains rather than single blocks, allowing for high throughput. Finally, Tendermint [10] utilizes a custom implementation of DPoS named Tendermint BFT, where nodes stake tokens to participate in consensus and allows for instant finality.

2.2 Blockchain Interoperability

Although the diversity of blockchain platforms enables networks to be tailored to specific use cases, it also introduces significant interoperability challenges. Each

platform may differ in its consensus mechanism, governance model, and security assumptions, making cross-chain communication complex. As a result, blockchain interoperability is best understood as a multi-layered problem. Inspired by the OSI model [34], several solutions have been proposed with a layered approach to address interoperability. For example, the Overledger project [35] defines four key layers: transaction (ledger storage), messaging (data exchange), filtering and ordering (message validation), and application. Another model proposed by Lohachab et al. [36] propose a more granular seven-layer model, introducing layers such as virtual-chain, access, and gateway layers. While these models differ structurally, they all acknowledge interoperability as a hierarchical challenge requiring solutions across various levels of the stack.

This layered framing allows researchers to adopt a divide-and-conquer strategy, where interoperability mechanisms can be designed to address individual layers rather than attempting an all-encompassing solution. However, despite these efforts, no universal layered model has emerged. Existing solutions remain largely multi-layered and highly use-case specific [37]. A notable example is the Interledger Protocol (ILP) [38], a blockchain agnostic protocol that works with different ledgers [22], which operates across the application, messaging, and networking layers. While it initially depended on conditional channels with Hashed Time-Lock Contracts [39], ILP has since evolved into a stateless, packet-switched protocol which enables greater scalability and flexibility.

The layered approach to interoperability has also been referenced in the context of security. In a comprehensive review, Augusto et al. [40] decomposed the security of

cross-chain systems into four layers: 1. the *network layer* focusing on the underlying decentralized ledger 2. the *protocol layer* focusing on the architecture of cross-chain protocols 3. the *implementation layer* which encompasses all components that enable interoperability whether they be on-chain or off-chain 4. and the *operational layer* which is concerned with deployment, upgradability, and monitoring. The paper further discusses different solutions for the security of interoperable systems including trusted third parties which can be a centralized trusted notary or could be in the form of a trusted execution environment (TEE), distributed trust which can be achieved through aforementioned consensus algorithms like PoW and PoS, and native state validation which can either be optimistic by assuming that all transactions are valid and waiting for third parties to challenge their validity, or by including precomputed proofs with the transactions such as zero knowledge proofs.

Several works have analyzed the security aspects of the different interoperability solutions according to their categorization [41, 42, 43]. In fact, according to Belchoir et al. [44], blockchain interoperability solutions can be organized into three major categories:

1. **Public Connectors** which deal with allowing interoperation between public blockchains. They are further subdivided into (a) *Relays* where a blockchain tracks the state of another (e.g., Polkadot's Parachains that connect to a central Relay Chain for shared security and consensus) [9] (b) *Notary schemes* which are trusted intermediaries that handle cross-chain transactions and can be centralized or decentralized (c) and *Hashed time-locked contracts* which require recipient acknowledgment within a set period or refund the sender, enabling

atomic swaps as is the case in Bitcoin's Lightning Network [45, 46, 39].

2. **Blockchain of Blockchains** which connect multiple application-specific blockchains via shared protocols. Examples include Cosmos' Inter-Blockchain Communication Protocol (IBC) [47] and Polkadot's XCMP [48], with smart contracts playing a major role in defining the behavioral logic of these solutions [44, 49, 50].

3. **Hybrid Connectors** which allow interoperation between both public and permissioned blockchains, offering more flexibility compared to public connectors. They are subdivided into (a) *Trusted Relays* which are trusted validators such as Hyperledger Cactus [51] that facilitate cross-chain transactions (b) *Blockchain-agnostic Protocols* such as the previously mentioned ILP to allow for cross-chain communication (c) and *Blockchain Migrators* which are tools utilized to move data and smart contracts from across blockchains.

2.3 Domain Name Services

Despite the substantial progress made in enabling cross-chain interaction, many proposed solutions overlook a crucial first step: blockchain network discovery. In practice, current systems typically rely on pre-defined lists of known networks [11]. These lists are either 1. managed by centralized authorities—raising concerns about centralization and governance bias—or 2. statically configured and manually updated by the clients [12, 13, 14].

While achieving full decentralization in blockchain interoperability remains challenging [52], specific aspects such as discovery services can be decentralized to mean-

ingfully reduce centralization risks. Moreover, the broader vision of the Internet of Blockchains (IoB)—where blockchains form a decentralized internet infrastructure—demands such decentralized discovery mechanisms [11, 53]. To achieve this, a decentralized solution should support automatic, trustless discovery of networks in a manner akin to Web2's Domain Name System (DNS) [54].

Continuing the theme of layered interoperability, IEEE Standard 3205 [55] formalizes this structure by defining five layers for cross-chain communication: 1. an application layer 2. a cross-chain transaction layer 3. a transport layer 4. a secure authentication layer, and 5. a data link layer. It introduces the Unified Cross-chain Packet (UCP) structure and highlights the need for a Blockchain DNS (BCDNS) to assign globally unique, verifiable identities to blockchain networks—drawing inspiration from conventional DNS, but implemented with certificate authorities and decentralized trust anchors.

Despite this recognition, the idea of a general-purpose, decentralized blockchain discovery service remains underexplored. One of the closest real-world systems is the Ethereum Name Service (ENS) [56] which provides human-readable names that resolve to Ethereum wallet addresses. Although ENS supports multi-chain address resolution via custom records, it remains Ethereum-centric, relying solely on Ethereum's infrastructure for storage and resolution. This dependence not only introduces scalability issues—especially given Ethereum's limited throughput—but also reintroduces single-platform reliance, which contradicts the goal of cross-chain neutrality.

Even though its potential for inter-blockchain discovery remains largely unexplored, blockchain technology has been used to implement alternative Web2 DNS

architectures in a more secure and decentralized way. Namecoin [57], a Bitcoin fork, introduced the *.bit* top level domain (TLD) to allow users to claim domains and record them using transactions on the blockchain in a trustless and decentralized way. However, scalability issues like domain name length limits quickly surfaced. A limit of 64 characters is imposed on domain names because the registration data is stored in the transactions, which must fit within the size limits inherited from the Bitcoin transaction structure. Moreover, it faced issues with name squatting because the fees required to claim domains were minimal [58]. To address these issues, Blockstack [59] improves on Namecoin by adding an offchain layer called the *virtual chain*, and offloads all the naming logic to that layer while only anchoring important data to the main network. Furthermore, it supports smart contracts for more advanced logic to be implemented, and attempts to prevent name squatting by adding renewal fees to keep claimed domains. EmerDNS [60] further improves on Blockstack by implementing a more rigid approach to mitigate name squatiing. It does so by introducing variable expiry dates to separate between disposable domains and long-term leases, and also gradually increases renewal fees over time to make squatting unsustainable.

Since both Namecoin and Blockstack are built on top of Bitcoin, they use PoW for their consensus which limits their performance. This is addressed by B-DNS [61] which instead uses the less-heavy PoS consensus along with building a domain index tree to improve query times. Furthermore, DNS-BC [62] leverages a consortium blockchain to implement a DNS caching system with high accuracy, security, and real-time performance. It improves latencies by introducing DNS over KCP (DoK) protocol, and manages security by utilizing a credibility score that influences the

probability of a participating node being selected for a query. Zhu et al. [63] also propose a solution that uses a consortium blockchain to replace traditional DNS. The main chain acts as a relay network for side chains within the consortium, which acts as root networks for each region. However, these side chains must be manually registered into the consortium, limiting the scalability of the solution. Moreover, the utilization of a consortium chain in both this solution and DNS-BC introduces a centralization risk, since the members of the consortium hold all the power when it comes to the chain's governance, and the system's upkeep depends solely on the members' goodwill. A similar problem is found in D-DNS [64] which utilizes a permissioned network built on Quorum, introducing an element of centralization since a set of trusted validators must be present, and utilizes PoS for its consensus. Moreover, since it utilizes a single blockchain for its solution, it potentially runs into scalability issues since all domain data is stored on a single chain.

2.3.1 Sequencer Centralization in Rollups

Rollups have emerged as the dominant Layer 2 (L2) scaling solution for Ethereum, offering substantial improvements in throughput and cost-efficiency. Implemented through a combination of smart contracts and off-chain components, rollups fall into two main categories:

1. **Optimistic Rollups** which execute transactions and bundle them into batches, but have the optimistic assumption that all transactions are valid, and thus do not provide a proof for their validity. Instead, a time window is allowed for nodes to submit disputes about invalid transactions before they are finalized

on the L1 chain. While this approach is lightweight, it relies heavily on honest behavior from participants [65].

2. **Zero-knowledge Rollups (ZKRollups)** which by contrast, use zero-knowledge proofs to generate cryptographic guarantees of transaction correctness. These proofs are attached to batches and verified on-chain, eliminating the need for a dispute period. Although more secure, ZKRollups incur higher computation costs due to proof generation [65].

Regardless of type, current rollup implementations rely on centralized sequencers—entities responsible for ordering transactions and submitting them to the L1 chain. This introduces a single point of failure and creates opportunities for censorship [66]. The centralization of sequencers opens the system to a multitude of other attacks such as delay attacks where adversaries try to prevent the rollup from making progress, denial of service attacks through expensive proof flooding in ZKRollups or fraud proof denial in optimistic rollups, and soft finality attacks where the sequencer invalidates a batch by reordering transactions differently from what was reported to users before finalizing them on L1 [67].

Several promising architectures have been proposed to decentralize sequencers due to the challenges faced by their centralized counterparts. Capretto et al. introduce *arrangers* as a new concept, combining sequencing with data availability responsibilities [68]. Their proposal offers both semi-decentralized and fully decentralized implementations. The semi-decentralized implementation utilizes centralized sequencers with a separate data availability committee, whereas the filly decentralized one leverages Set Byzantine Consensus (SBC) to achieve a decentralized arranger without a

single point of failure. Meanwhile, zkSync have proposed a consensus mechanism named ChonkyBFT [69] for their rollup. However, it is not fully integrated with their solution, making their current implementation still centralized. ChonkyBFT is a Byzantine Fault Tolerant (BFT) consensus protocol aimed at enhancing the scalability of BFT-based protocols in a distributed setting. This is achieved by introducing optimizations like signature aggregation, which utilizes threshold signatures to enable validators to collectively sign a message without revealing their contributions, decreasing the number of messages needed to be exchanged. This decreases the communication complexity of the protocol from the usual $O(n^2)$ in PBFT to a linear complexity, decreasing the overall network latency and computational costs. Furthermore, future plans for the protocol include implementing token staking and slashing for validators to increase the security of the decentralized system against malicious actors.

2.3.2 ZKP and Multiparty Setups

Zero-knowledge proofs (ZKPs) have become a foundational cryptographic technique for enhancing privacy and security in blockchain systems. Blockchains are transparent, decentralized, and immutable ledgers that contain all historical transaction data and are made publicly accessible to network participants While blockchains are valued for their transparency and immutability, this openness can introduce privacy concerns, as sensitive transaction data is publicly accessible to all participants [70].

ZKPs are interactive verification protocols with three core properties: complete-

ness (honest provers convince honest verifiers), soundness (cheating provers cannot convince verifiers of false statements), and zero-knowledge (no additional information is leaked to the verifier). These properties make ZKPs especially useful for privacy-preserving applications, such as confidential payments, private voting, and selective disclosure in identity systems. A widely adopted class of ZKPs in blockchain is zkSNARKs (Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge), which are favored for their small proof sizes and fast verification times—both crucial for on-chain efficiency [71].

While zkSNARKs can facilitate trustless operation within a blockchain ecosystem, they themselves require a trusted setup. This setup is required to generate a Common Reference String (CRS), which can be referred to as the system parameters used to generate and verify ZKPs. Given that the CRS is used throughout the entire system, if the setup is compromised, the security of the system can be undermined by potentially allowing for the creation of fraudulent proofs [72].

To mitigate this risk, multiparty computation (MPC) protocols have been introduced to decentralize CRS generation. The most well-known of these is Powers of Tau [18], in which multiple independent participants contribute randomness to the setup process. The core principle is that the final CRS remains secure as long as at least one participant behaves honestly and keeps their secret contribution undisclosed.

The MPC process involves sequential transformations of elliptic curve points by each participant.

Given curve points $G = \{G_1, G_2, \ldots, G_n\}$ and $H = \{H_1, H_2, \ldots, H_m\}$, each participant multiplies them by their secret scalar r_i, producing new points:

$$G'_i = r_i \cdot G_i, \quad H'_j = r_i \cdot H_j$$

After all participants apply their contributions, the final CRS is a layered composition of all transformations.

The security of the CRS depends on the hardness of the *Elliptic Curve Discrete Logarithm Problem (ECDLP)*, which ensures that no participant—nor any colluding group—can reverse or extract others' contributions.

While projects like Powers of Tau have successfully demonstrated secure CRS generation on Layer 1, they introduce significant computational and storage overhead, especially in resource-constrained or high-throughput settings. This motivates offloading CRS generation to Layer 2 environments, provided decentralization and trust-minimization can be preserved. This idea is expanded upon in the subsequent chapters of the thesis.

Chapter 3

System Design

The systems presented in this thesis have one overarching goal, that is to enable discovery between isolated blockchain networks. The achievement of this goal hinges on the creation of a solution that meets certain requirements. These imposed requirements are governed by the inherent characteristics of blockchains and by the functional demands that the system must meet. To this end, this thesis decomposes the primary objective of inter-blockchain network discovery into two categories of sub-goals:

1. **Imperative Goals** which satisfy characteristics that are inherently required by blockchain networks, without which the solution would not be viable regardless of its performance. The two imperative goals of this thesis are (a) Decentralization, (b) and Collusion Resistance.

2. **Functional Goals** which satisfy the practical requirements imposed on the solution to provide services with minimal failures and appropriate performance.

The three functional goals of this thesis are (a) Scalability (b) Dynamicity, (c) and Self-Sufficiency.

3.1 Designing the Discovery Architecture

The design of the discovery architecture must achieve the imperative and functional goals discussed earlier, of which the first and most important is *decentralization*. The main objective behind decentralizing the architecture beyond mitigating single points of failure is to eliminate the risk of censorship or governance bias which comes with centralization. Since the maintaining entity holds absolute power over a centralized system, it can selectively choose to stop providing services to certain networks they deem as competitors or simply for political reasons. Furthermore, the governing entity can be biased towards specific beneficiaries due to ties between them or because they were offered external incentives, providing them higher priority regarding identifier registration or query resolution.

To decentralize the architecture, the simplest approach is to utilize a blockchain network. This is because blockchains are built on a distributed network of nodes, eliminating the single points of failure. Moreover, the inherent security provided by blockchain infrastructure because of the immutability of its ledger, providing permanent and reliable provenance. Furthermore, given that the blockchain network is appropriately configured with a consensus mechanism that encourages decentralization, such as PoW, PoS, and Proof of Elapsed Time [73].

This design satisfies the decentralization criterion and is relatively simple, making maintenance straightforward. However, if a subset of nodes colludes, they can

influence the entire system, indicating that it lacks resistance to collusion. Furthermore, this design does not fulfill the functional goals required by the system. While a case can be made for meeting dynamicity requirements since records can be modified through transactions on the chain, the design quickly encounters scalability challenges. Since all network records are stored on-chain, the storage burden on participating nodes grows rapidly. In addition, depending on the consensus mechanism employed, validator caps may be imposed to ensure consensus efficiency and timely finality. Such limits concentrate decision-making power in a subset of nodes, thereby undermining decentralization.

3.1.1 Architecture

The problems described above must be solved for the design to be viable. Resistance to collusion can be achieved by dividing the system into several subsystems such that, even if coordination occurs among participating nodes, they cannot assume control over the entire system. The growth of storage demands as the number of records increases can likewise be mitigated by distributing data across multiple subsystems. The validator cap issue can be addressed in a similar way: although validator limits cannot be raised within a single network, the overall effective system capacity can be increased if it is comprised of multiple blockchain networks.

The analysis of the aforementioned problems leads to a unified conclusion: the system should be divided into multiple blockchain networks. However, the organization of these networks remains an open question. Ideally, the networks comprising the system must be logically structured so that their roles are easily understood by

users. The existing Web 2.0 DNS architecture achieves this by arranging servers hierarchically: root servers route queries to the appropriate Top-Level Domain (TLD) servers, which in turn direct requests to the authoritative name servers that store DNS records and return the IP address associated with the queried domain.

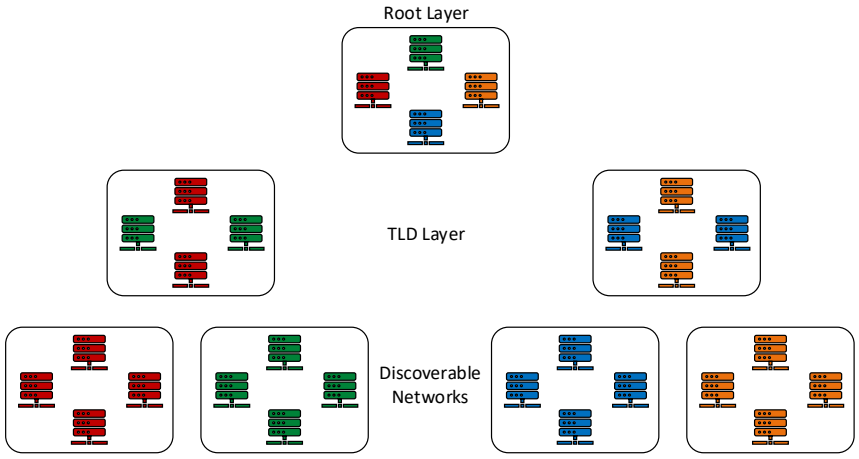

Figure 3.1: DNS Architecture Diagram [1]

Leveraging the well-established design principles of Web 2.0 architecture, this thesis proposes a multi-layered architecture for a blockchain-based discovery mechanism. Inspired by traditional DNS, blockchain network domains are organized into top-level domains (TLDs), as shown in Figure 3.1. The root layer consists of a single blockchain network that serves as the entry point to the architecture. The TLD layer is composed of several blockchain networks, each responsible for a specific TLD and for storing the connection information associated with the domains under its jurisdiction. Moreover, the nodes participating in all layers of the architecture are provided by the discoverable networks themselves, which are incentivized to contribute to the system through mechanisms discussed later in this thesis.

3.1.2 Operational Workflow

The architecture described above addresses the challenges of collusion resistance and scalability. However, another functional requirement of the mechanism is *dynamism*, which eliminates reliance on static lists or centralized repositories. To achieve this, discoverable networks must have the ability to freely opt in or out of discovery. Responsibility for registration, updates, and transfers is therefore delegated to the networks themselves, removing the need for a centralized authority to manage lists of known networks. For this to be effective, the connection information of blockchain networks stored within the architecture must be easily modifiable, transferable, or revocable at the request of the associated network. The design achieves this by defining explicit protocols for each operation.

Since the primary objective of a discovery mechanism is to enable interconnection between networks through identifiers, these identifiers must be resolved into raw connection information that can be used for communication. Following the Web 2.0 inspiration, the resolution protocol is analogous to how domain names are resolved in a hierarchical manner. In traditional DNS, domain names are resolved by a recursive resolver that queries root servers, TLD servers, and authoritative name servers in sequence to retrieve the IP address associated with the requested domain.

Correspondingly, the resolution protocol outlined in Figure 3.2 illustrates how blockchain domains are resolved recursively. To resolve a domain, the light client or arbitrary node first connects to the root network, whose connection details are static and well known, as it serves as the entry point to the architecture. Once connected, the client queries the root network to obtain the connection details of

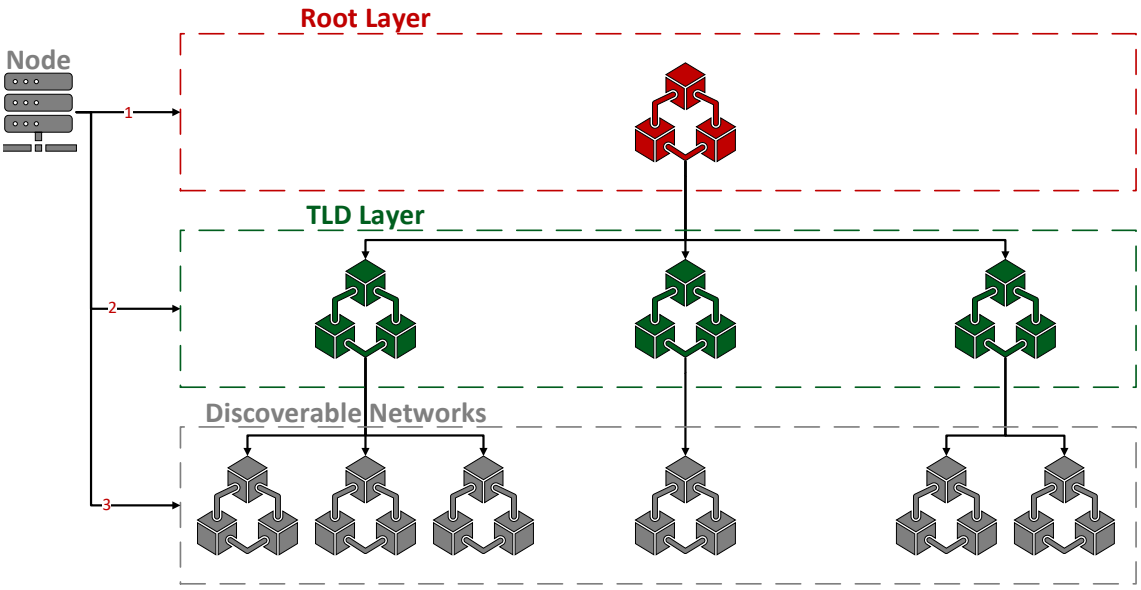

Figure 3.2: Domain Resolution Protocol [1]

the TLD network associated with the target domain. Using these details, it then connects to the TLD network and queries for the records associated with the domain's underlying network. Finally, once the domain is resolved, the client uses the obtained information to establish a direct connection with the target network and caches the result locally to improve efficiency.

Before domains can be resolved by the architecture, they must first be registered by the corresponding networks. In traditional DNS, domain registrars manage reserved domains and sell them at prices determined by their popularity or rarity. This arrangement places trust in centralized entities that may impose restrictions or censor domains at will. In contrast, the decentralized design presented in this thesis allows blockchain networks to claim domains on a first-come, first-served basis. To register, networks must pay the transaction fees associated with the process and contribute resources to the architecture, as enforced by the incentive mechanism discussed in

Section 3.1.4.

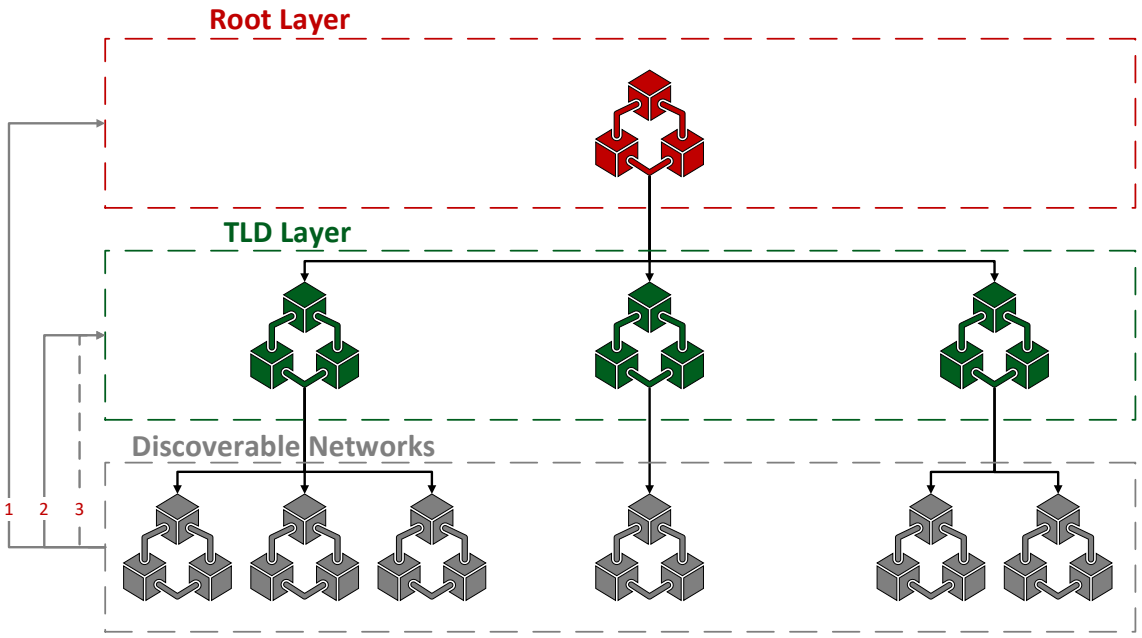

Figure 3.3: Domain Registration Protocol [1]

Before the registration process can take place, the blockchain network that wishes to be discovered must internally agree on a domain to claim through its consensus mechanism. Concretely, networks must adhere to the registration protocol shown in Figure 3.3, which is described as follows:

1. A node from the network, selected by its consensus mechanism, queries the root network to obtain the connection details of the TLD network associated with the desired domain.

2. The selected node submits a transaction to the TLD network to register the domain.

3. Optionally, if the desired domain is already taken, the network may select an

alternative domain and repeat the process by attempting to register it with the TLD network.

3.1.3 Asset Discovery

So far, the system design section has outlined the infrastructure required to support blockchain network discovery and achieve the stated design goals. However, while network discovery is necessary for realizing the Internet of Blockchains, it is not sufficient on its own.

When blockchain networks interoperate, it is common for a contract on one network to require an asset provided by another. In this thesis, the term *asset* refers to resources, tokens, or services that a blockchain network can offer. For seamless interoperability, smart contracts and blockchain networks must therefore be able to query other networks for the assets they provide. This is enabled through a mechanism that allows networks to locate providers of a desired asset, facilitating efficient cross-chain interactions while ensuring that asset identification and access occur in a decentralized manner.

Leveraging the decentralization achieved by the system design, the architecture is extended to support asset discovery in addition to network discovery. The objective of this extension is to allow inquirers to locate specific assets, provided they are already aware of their existence. This constraint mitigates potential security risks in which malicious clients could scrape information stored on-chain and exploit registered assets. By restricting discovery to assets that clients already know, the design discourages malicious actors from creating counterfeit or harmful assets. Fur-

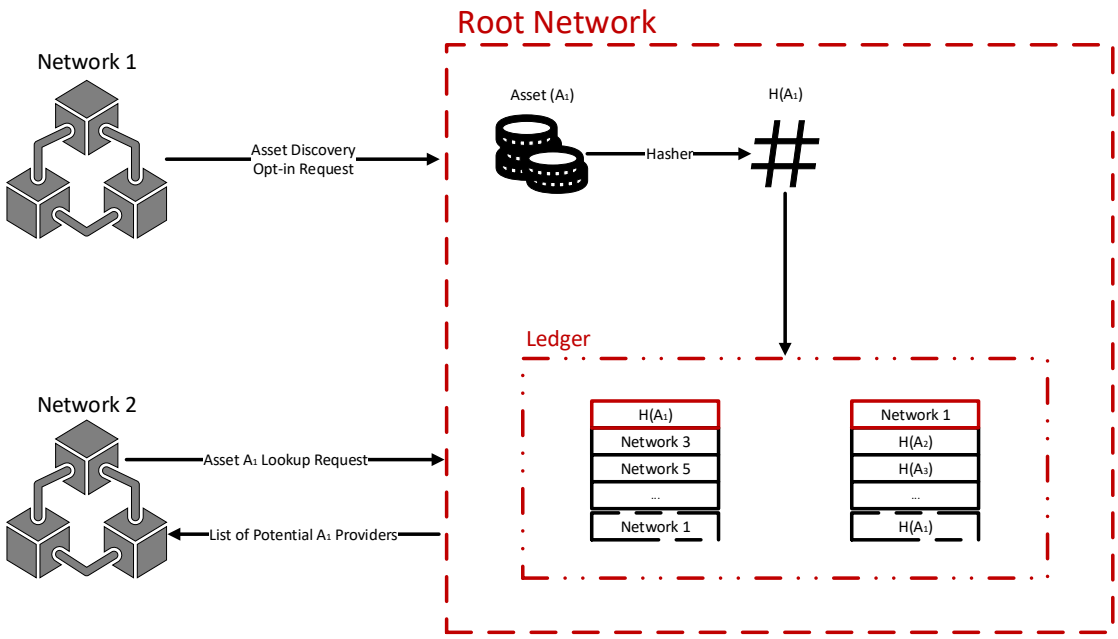

Figure 3.4: Asset Discovery Protocol [1]

thermore, this approach follows the precedent set by traditional DNS (and by the decentralized network discovery system presented here), where users must know of a network's existence and its associated domain to interact with it.

Taking into account the characteristics discussed above, Figure 3.4 illustrates the protocol that outlines the process for a network to opt into asset discovery and to query the architecture for locating a specific asset. To preserve anonymity, the architecture stores hashes of asset identifiers rather than the raw identifiers themselves. This ensures that any information scraped by malicious clients from the chain is effectively useless, as a hash can only be linked to an asset if the attacker already knows of its existence and possesses the original identifier.

However, for this strategy to be effective, the identifiers must be hashed by one of the parties. If hashing were performed on-chain, the registering network would

need to submit the raw identifiers to the architecture, causing them to be permanently recorded in the transaction history. This would expose the identifiers publicly and undermine the design's goal. To avoid this, the hashing process is delegated to the registering network. Each network must hash its identifier using a previously agreed-upon algorithm and then submit a transaction containing only the hash to the architecture.

Following the hashing of identifiers, assets can be registered for discovery through the architecture in the root network, as shown in Figure 3.5. The process begins when a client or node submits an *asset registration request* in the form of a transaction to the root network. Upon receipt, the system records the request in a queue on-chain rather than registering the asset immediately. Processing is deferred for two reasons:

1. Due to the decentralized and dynamic nature of the architecture, asset registration requests can be submitted by any client or node without restrictions. As a result, requests may target invalid or non-existent domains, or they may be submitted for a valid domain by an unrelated malicious party. To prevent this, the domain's legitimacy must first be validated, and the requester must be verified as a maintainer (Section 3.1.4) associated with the network requesting the asset registration.

2. All transactions submitted to the blockchain must be deterministic, meaning they must always produce the same result given identical inputs. However, validating domain and requester legitimacy requires external network queries, which are inherently non-deterministic and therefore cannot be performed as part of the transaction.

To address these requirements and circumvent the issues associated with determinism, the system employs off-chain workers to validate and process incoming registration requests. Since asset registration transactions simply create internal requests that are queued on the chain, the workers periodically poll the chain state to detect requests newly appended to the queue. Once identified, a worker retrieves and processes the request by first validating the domain's legitimacy. This involves fetching the TLD connection details associated with the domain from the chain and then using the TLD network's Remote Procedure Call (RPC) interface to query its state for the corresponding domain information. If the domain exists, the worker verifies that it is currently active. The list of maintainers associated with the domain is then checked for the presence of the identifier of the node that originally submitted the asset registration transaction. If both conditions are met, the worker executes a transaction to register the asset on-chain.

The transaction submitted by the off-chain worker to register an asset follows the structure shown in Figure 3.5. Asset identifiers are leveraged as keys that map to lists of providers (domains) offering the corresponding asset. This design allows inquirers to efficiently obtain a set of providers they can interact with. In addition, the system maintains an inverse mapping on-chain, where providers act as keys that map to the assets they offer. This inverse mapping supports asset discovery revocation: it enables the system to quickly identify all assets associated with a given provider without scanning every asset record, since a provider may appear in multiple provider lists. This structure is later leveraged by the incentive mechanism to revoke asset discovery for assets associated with revoked domains.

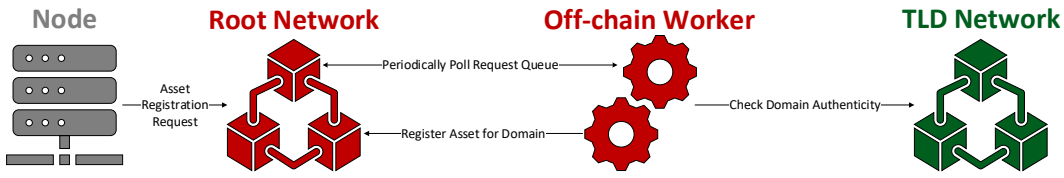

Figure 3.5: Asset Registration Protocol [1]

3.1.4 Incentive Strategy

So far, the design has outlined the strategies and protocols employed to achieve the *imperative goals* and *functional goals* of the system, with one exception: self-sufficiency. The realization of both decentralization and dynamism required the registration process to be delegated to the discoverable networks. While this enables networks to freely decide whether to participate in discovery and to update their information without relying on a centralized authority, it introduces a critical issue: **ensuring sustained participation in the architecture**.

For the architecture to maximize decentralization, nodes from diverse networks must contribute to its maintenance to prevent the consolidation of power within a single entity. However, from the perspective of discoverable networks, the most natural course of action is to utilize the architecture without maintaining it, thereby saving resources that would otherwise be required. To address this challenge, a robust incentive mechanism must be designed such that participation by all discoverable networks is guaranteed, mitigating all forms of free-riding.

Following this principle, the architecture incentivizes participation through transaction fees. The design supports actions such as network domain registration, asset registration, and domain transfer. Since these actions interact with the chain by ap-

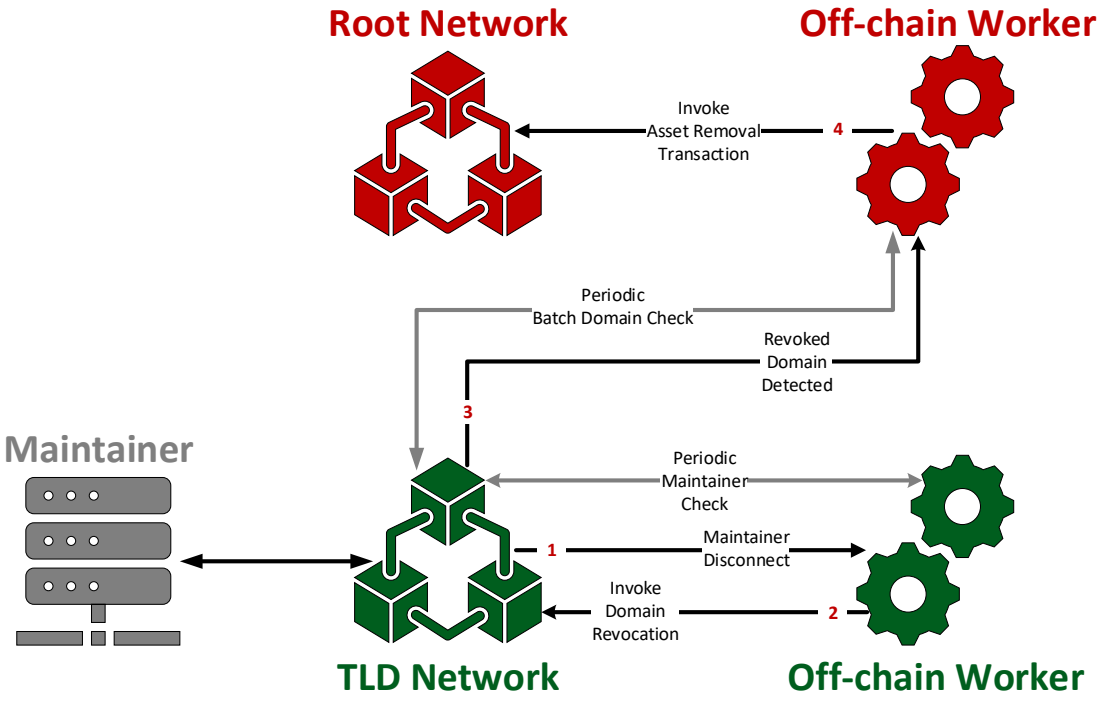

Figure 3.6: Mandatory Participation Incentive Mechanism [1]

pending or modifying information, they must be implemented as transactions. Once submitted, transactions are processed by the nodes maintaining the architecture, consuming their resources. To compensate these nodes, each transaction includes a fee, providing a direct financial incentive to sustain the architecture. Fees vary according to the computational cost of the transaction, which can be determined in advance due to their deterministic nature.

While nodes might be encouraged to participate in architecture maintenance by transaction fees, it alone is insufficient to ensure participation and maintain self-sufficiency. In addition to transaction fees, a more robust incentive mechanism is required to ensure the architecture remains operational in perpetuity. This is achieved through the mandatory participation incentive mechanism shown in Figure 3.6. The

objective of this mechanism is to establish a mutually reinforcing relationship between the discoverable networks and the architecture such that all parties benefit from one another. Specifically, for a network to claim a domain it must provide a minimum number of nodes that contribute to the maintenance of the TLD network that the domain belongs to and the root network of the architecture.

The strategy employed by the mechanism involves mandating that networks provide a set of node identifiers during registration, which are treated as maintainers of the architecture. Additionally, off-chain workers periodically poll the network to continuously monitor the peers connected to it. If a node is detected to have left the network, the off-chain worker initiates the process of revoking the domain associated with the node that ceased to maintain the network. This mechanism incentivizes networks to ensure that networks have a strong incentive to contribute resources, as the availability of the architecture directly depends on their continued participation.

3.2 Designing a System for Trust Anchor Generation

So far, the design has focused primarily on designing the architecture needed to support network and asset discovery. The design was completed such that both the *imperative* and *functional* goals of the system were satisfied while providing the required services. However, when comparing traditional Web 2.0 discovery services to the architecture presented in this thesis so far, a critical difference is revealed: trust guarantees.

The Web 2.0 architecture relies on traditional DNS for discovery through domain resolution. This system utilizes servers owned by different organizations at different levels of its hierarchical architecture to assist with the recursive resolution protocol that it follows. However, since man-in-the-middle, phishing, and replay attacks are potential risks during the resolution process, the traditional system employs trust anchors. Specifically, to verify the ownership of a domain, a Transport Layer Security (TLS) certificate is typically presented to the client by the server. This certificate is cryptographically generated by a designated Certificate Authority (CA) which are trusted entities that oversee the authenticity of domains, hence they are trust anchors.

A similar system is required to enhance security and increase trust in the decentralized discovery architecture presented in this thesis. However, utilizing the CA system employed by traditional DNS completely undermines the goals of the design, therefore the core problem of centralized trust anchors must be solved before taking any additional steps. Since utilizing the same infrastructure used for current certificate generation is infeasible due to its centralization, a natural approach would be simply to create a decentralized alternative that follows the same methodology. However, this would still encounter privacy problems. Creating such certificates requires networks not only to announce that they own a domain but also reveal metadata about themselves that they otherwise might not want to. Fortunately, a cryptographic scheme fits the aforementioned criteria of being decentralizable while also allowing for selective privacy, which is the essence of a zero knowledge proof (ZKP).

The usage of ZKPs allows a party to prove the validity of their statements without revealing any underlying information related to the statement. This directly coincides

with the decentralized trust and selective privacy characteristics that blockchain networks desire. A common system used for ZKP generation and verification is Zero Knowledge Succinct Non-interactive Arguments of Knowledge (zkSNARKs). This system is most commonly used in blockchain applications due to its small proof size and low computation costs associated with the proof verification. However, zkSNARKs rely on a trusted setup phase for the generation of a Common Reference String (CRS) which is a public, shared parameter required when generating or verifying proofs, making it the system's trust anchor. Consequently, if the CRS is compromised, then proofs can be forged, undermining the security of the system. Since centralized CRS generation results in a single point of failure and can lead to its abuse, decentralized ceremonies like the Powers of Tau have been proposed to mitigate that risk.

However, these ceremonies are often held on Layer 1 (L1) networks, diverting resources that could otherwise be used for user-facing applications. Furthermore, they fail to address long-term collusion risks associated with CRS generation. While a critical characteristic of multi-party CRS generation is that it remains secure as long as at least one participating node is honest, the risk of collusion increases if the selected nodes are always from the same pool. Moreover, if the CRS remains static (i.e. the CRS has been generated once and is being used for a long time thereafter), the risk of all nodes colluding with one another overtime remains.

Solving the aforementioned problems first requires that the CRS generation ceremony be delegated to a secondary network instead of performing it on L1 directly. This arrangement ensures that the performance of the L1 network is independent

of whether the ceremony is active. Layer 2 (L2) solutions, particularly rollups, are most suited for this purpose because they directly inherit the security models of their L1 counterparts, making their interoperability simple. Furthermore, to mitigate the possibility of long-term collusion between participants, the generated CRS must not be static. Specifically, proofs can use different CRS values generated at different points in time, generated by various participants, drastically reducing the risks of CRS compromise through collusion.

Accordingly, the trust anchor generation system proposed in this thesis employs an architecture that comprises two key components. The first is the rollup that will be utilized to offload the ceremony. However, the sequencer — a core part of all rollup architectures which manages the ordering of transactions in batches before they are anchored to L1 — is strictly centralized in contemporary commercial solutions, undermining the decentralization of the process and directly clashing with the *imperative goals* of the system design presented in this thesis. Thus, the system employs a decentralized sequencer network for offloading the ceremony, forming the foundation of the trust anchor generation system and enabling the coordination and execution of subsequent components. Leveraging PBFT, the nodes participating in this network coordinate to collaboratively order transactions, mitigating the single points of failure present in centralized rollups.

Following on from this, the design introduces a decentralized CRS generation ceremony that uses the Powers of Tau protocol, operating periodically over the sequencer network. Each round of CRS generation, a randomized subset of sequencer nodes are selected for participation in a multiparty computation process to produce the CRS

instance for the current ceremony. The design presents two variations of the CRS generation protocol: a semi-decentralized approach where a smart contract is used to coordinate the ceremony, and a fully decentralized one where participating nodes directly communicate with one another in a peer-to-peer fashion. In both cases, the resulting CRS is anchored to the L1 blockchain to ensure public verifiability and state integrity.

3.2.1 Decentralized Sequencer Model

As mentioned above, the sequencer lies at the core of all rollup solutions. It is mainly responsible for ordering the transactions submitted to the rollup and aggregating them into batches, although it often also executes them. Since current solutions utilize centralized sequencers, they suffer from single points of failure, as shown in Figure 3.7, compromising the ever so desirable decentralization characteristics sought after by blockchain systems.

The typical life cycle of an L2 transaction is as follows: users submit their transactions to the rollup — specifically a *node* component that exposes an interactive RPC interface — which are then passed to the sequencer. The sequencer can potentially execute the transactions to obtain the state transitions caused by them depending on its implementation. Regardless, the transactions are then ordered based on the rules enforced by the sequencer, and their results are then passed to the solution's proof utility. In the case of ZKRollups, this utility is used to generate zero knowledge proofs for the transactions which can later be verified, after which they are posted to the L1 network as a batch. Conversely, optimistic rollups directly post the ordered

transactions as a batch to the underlying L1 network, maintaining a rebuttal window where provers can potentially dispute fraudulent transactions.

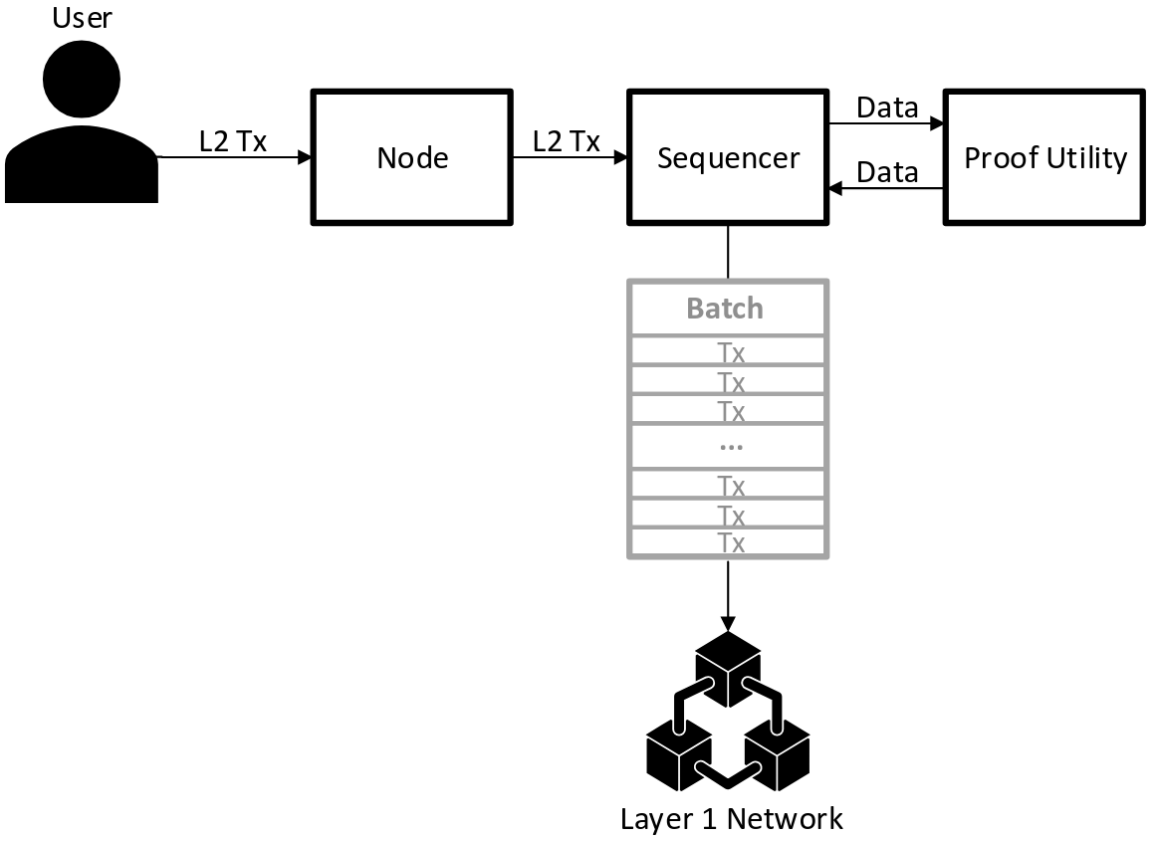

Figure 3.7: Centralized Sequencer Architecture [1]

In a decentralized sequencer model, the rollup no longer depends on a single centralized sequencer. Instead, its reliance on a single centralized sequencer is eliminated in favor of a network of nodes that are part of a P2P network as Figure 3.8 illustrates. Rather than letting a single node dictate the order of transactions included in each batch, the set of participating nodes collectively agree on an order through a consensus mechanism. This distributes the authority over multiple participants, eliminating single points of failure and in turn enhances the resistance to censorship and the se-

curity of the overall system, since the order of transactions cannot be dictated by a single entity.

The recurring CRS regeneration process presented in this design requires that the result be immediately usable by zkSNARK systems that are dependent on it. Optimistic rollups operate under the assumption of transaction validity and delegate the validation process to provers who may require up to a week. This makes them incompatible with the trust anchor generation system, making zero knowledge rollups the preferable choice since they include proofs with the submitted batches, resulting in the batches being immediately validated and finalized.

This thesis utilizes the high-level decentralized sequencer model discussed above to design a ZKRollup to be used to enable the CRS generation scheme. The decentralized sequencer network supporting this rollup consists of sequencer nodes that operate independently of one another, but constantly communicate to ensure that transaction and batch information is agreed upon and relayed to all nodes. To achieve this, the network supports three proprietary protocols:

1. A *transaction protocol* for relaying incoming transactions.

2. A *batch protocol* for broadcasting the proposed transaction batches.

3. A *consensus protocol* for coordinating agreement among sequencers.

This P2P layer is what enables the consensus mechanism to function. In this design, Practical Byzantine Fault Tolerance (PBFT) is the consensus utilized, chosen for its simplicity of implementation while still allowing nodes to reach an agreement on transaction order in a decentralized manner. It consists of three phases:

1. **Pre-prepare** where a leader orders transactions into a batch and proposes the batch to the network.

2. **Prepare** where after having received the proposed batch, the followers validate the batch and broadcast a message indicating their agreement to it.

3. **Commit** where nodes will broadcast their confirmation indicating their acceptance of a batch once enough prepare messages are received.

For both the *Prepare* and *Commit* phases, nodes progress to the next stage only when a *supermajority* (at least two-thirds of the total number of nodes) is achieved. Consequently, once a node collects enough *Commit* messages, the batch is finalized.

To maximize the decentralization of the scheme, any node must be allowed to participate. However, the resulting outcomes of consensus operations heavily depend on the honesty of participating nodes. Thus, incentivizing nodes is critical to encourage them to maintain honest behavior. This design incorporates a reputation mechanism to acheive this purpose. Malicious nodes that misbehave resulting in network disruptions or fraudulent transactions are penalized in proportion to the severity of their actions. Minor incidents like occasional disconnections that occur due to unstable network conditions are lightly penalized. However, if disconnections become regular and consecutive, the node is temporarily excluded from participating in consensus. Furthermore, severe violations like submitting invalid contributions are considered malicious and result in the node's reputation being heavily reprimanded. Regardless of how a node loses its reputation score, falling below a pre-defined threshold results in it being permanently excluded from further consensus operations.

Once transactions are ordered by the sequencer network, they are aggregated into a batch. However, before anchoring it to L1, the batch must have a proof generated for it since this is a ZKRollup. The proof is attached to the batch of finalized transactions and is used to prove that all state transitions such as smart contract calls, account balance updates, and nonce increments are valid. The rollup maintains a Merkle tree which records all state transitions, having the Merkle root represent the current system state, where the transition between the old and new state is proven by the ZKP. The proof is constructed using a zkSNARK circuit, which enforces transaction validity constraints, including correct nonce ordering, where each transaction must use a nonce greater than that of the sender's previous submission.

Once generated, the proof can be attached to the batch, after which both are submitted to L1, enabling efficient verification of batch validity. This is facilitated by a smart contract deployed on L1 which is responsible for receiving and processing finalized L2 batches. The contract verifies the validity of the proof attached to the batch and then stores it on the chain. Each submitted batch includes metadata such as its batch number, Merkle root, and transaction hashes, allowing direct access of batch contents through L1 for further verification if required.

3.2.2 Decentralized CRS Generation

So far, the design has focused on decentralizing the sequencer component of ZKRollups. This rollup design is utilized as the underlying infrastructure for the trust anchor generation scheme. As previously mentioned, zkSNARKs are the dominating class of ZKPs for blockchain applications due to their compact proof sizes and

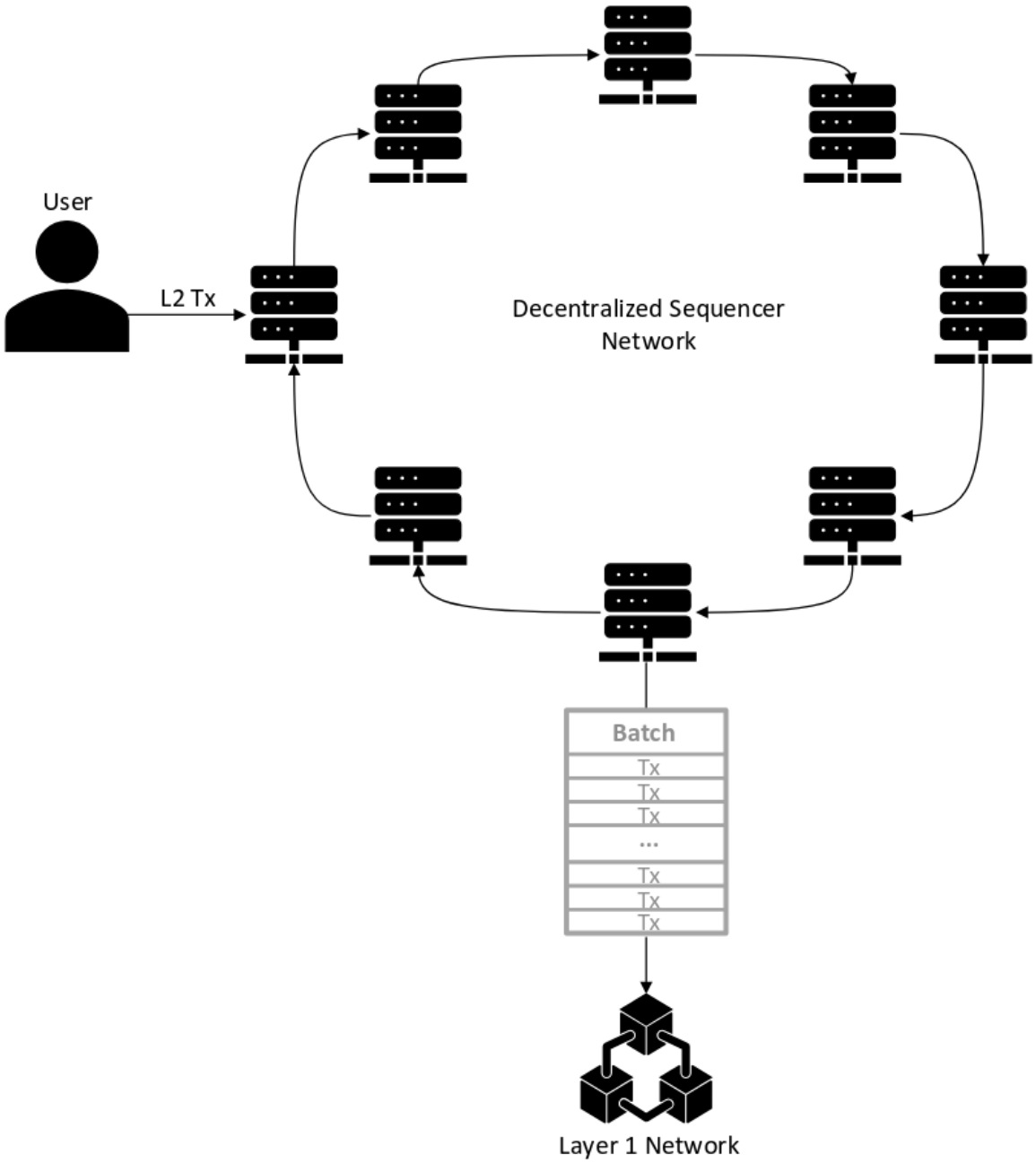

Figure 3.8: Decentralized Sequencer Architecture [1]

efficient verification. However, they rely on the CRS being securely generated prior to their operation, making it their trust anchor.

Formally, the CRS can be represented as a tuple (G, H), where G and H are sets

of points on elliptic curves. Specifically:

$$G = \{G_1, G_2, ..., G_n\}, H = \{H_1, H_2, ..., H_m\}$$

where G_i and H_j are elements of an elliptic curve group, and the values of n and m depend on the size and complexity of the zkSNARK circuit.

Since the CRS is the trust anchor of zkSNARKs, their security is directly dependent on its integrity. If the CRS is compromised, invalid proofs can be made to appear valid, allowing forgery by adversaries. However, the chances of CRS compromise can be reduced through distributing its generation. This is typically achieved through MPC protocols, where independent participants take turns to contribute randomness to construct a CRS instead of allowing a single entity to control the process.

In a multi-party CRS generation ceremony, each participant applies a transformation to the CRS using a secret random scalar r_i Mathematically, the transformation is expressed as:

$$G_i^{'} = r_i \cdot G_i, H_j^{'} = r_i \cdot H_j$$

where $G_i^{'}$ and $H_j^{'}$ represent transformed points, and r_i is a scalar randomly sampled from the underlying elliptic curve group. The final CRS is computed by sequentially applying each participant's transformation in turn:

$$G_{final} = r_i \cdot (r_{i-1} \cdot (...(r_1 \cdot G)...)), H_{final} = r_i \cdot (r_{i-1} \cdot (...(r_1 \cdot H)...))$$

Since the transformations apply contributions from many different participants, this increases the security of the generated CRS. A key security property is that the CRS remains cryptographically secure as long as at least one participant behaves

honestly and keeps their scalar r_i secret. In essence, this security is derived from the difficulty of the discrete logarithm problem. This is adopted into elliptic curve computations, making it computationally infeasible to calculate a scalar r such that $B = r \cdot A$ where A and B are both points on the curve.

While CRS generation has been implemented in real-world applications such as Powers of Tau, existing approaches primarily rely on Layer 1 networks and incur significant computational and storage overhead. This thesis addresses this limitation by leveraging the decentralized sequencer network proposed to offload the CRS generation ceremonies to L2, while preserving its decentralized nature. Two variations are proposed: 1. Semi-decentralized CRS generation, and 2. Fully decentralized CRS generation.

Semi-decentralized CRS generation

This scheme utilizes the multiparty CRS generation logic discussed above, but employs a smart contract deployed on L2 to manage the ceremony state. The main purpose behind this design is to enable the ceremony to be held on L2 without modifying the sequencer implementation. However, since the ceremony state is managed by a smart contract, all information is publicly accessible including participant secrets.

To circumvent this, a commit-reveal strategy is employed where each participant first commits their random scalar r_i by publishing a hash $h_i = H(r_i)$, where H is a hash function. Once all participants have published their commitments, they reveal their scalar values which are then used to construct the CRS, preventing participants from changing their contributions maliciously based on other contributions.

CRS generation rounds occur periodically at a predetermined interval agreed upon by the participating nodes. At the start of each round, nodes are allowed to register themselves in the pool of eligible participants. A subset of participants is then selected to make commitments and contribute to the generation of the CRS. After all contributions have been made, the CRS is finalized and made available for zkSNARK operations. The protocol can be more concretely summarized as follows:

1. *Registration*: Nodes register themselves in a pool of eligible participants, from which a subset $P = \{P_1, P_2, ..., P_k\}$ is selected in each round to participate in the CRS generation ceremony.

2. *Commitment*: Each participant P_i from the set of participants P generates a random scalar r_i, computes a cryptographic commitment $h = H(r_i)$, and submits to the system to prevent premature disclosure.

3. *Contribution*: Once all commitments have been submitted, contributions are applied sequentially. The CRS is updated as follows: $G_i = r_i \cdot G_{i-1}, H_i = r_i \cdot H_{i-1}$.

4. *Finalization*: Once all contributions have been applied, the final CRS (G_{final}, H_{final}) is stored and commitments are revealed to ensure the integrity of the final generated CRS.

Fully decentralized CRS generation

This scheme, unlike the previous, requires the modification of the sequencer implementation. This is because instead of relying on a smart contract to manage ceremony

state, sequencer nodes manage the state amongst each other through consensus. Each participating node stores a copy of the state which indicates the status of the ceremony including the round number, list of participants, current step in generation sequence, and the intermediate CRS.

Since a consensus mechanism is required, the ceremony is integrated with the existing PBFT mechanism used by the sequencer network, ensuring that all sequencer nodes maintain a consistent view of the CRS state throughout the process. New CRS generation rounds are periodically scheduled, each with a random subset of participants selected. Once initialized, each participant contributes to the CRS ceremony by applying their transformation to the intermediate CRS. The new state is then propagated to other nodes and is agreed on through PBFT before allowing the next participant to contribute. Once all participants have made their contributions, the CRS is finalized to be used by zkSNARK systems, and is anchored to the L1 chain.

Chapter 4

Implementation

4.1 Discovery Architecture Implementation

The first phase of implementation was the discovery architecture[1]. This is to satisfy the main objective of this thesis: enabling decentralized inter-blockchain discovery to achieve an Internet of Blockchains. The implementation utilizes the Substrate blockchain framework [27] which has proven its reliability by being the core technology used in the Polkadot network. This is achieved in part due to it being written in the Rust programming language since it allows intricate resource optimization, providing the nodes with high performance. Furthermore, the ability to develop light-weight clients due to their support for WebAssembly (WASM) allows for seamless interaction with Substrate-based blockchains using minimal resources, enabling even browsers to function as clients that can query the chain state and submit transactions without being full nodes.

[1]The implementation of the architecture and its evaluation scripts can be accessed on: https://github.com/icdcs248/enabling-blockchain-interoperability-through-discovery

The aforementioned characteristics make Substrate an appealing platform for the implementation of the architecture, but its main attractive trait is modularity. Through its *pallets*[2] system, complex modifications can be made to the blockchain runtime at ease. This is because the runtime is comprised of pallets, where each pallet provides specific functionality. For example, in Substrate's template node implementation[3], the block authoring mechanism is handled by *pallet_aura*, while the block finalization logic is managed by *pallet_grandpa*, together forming the network's consensus mechanism. Following this pattern, the discovery architecture is implemented across three new pallets: 1. the root pallet 2. the TLD pallet 3. and the asset discovery pallet.

4.1.1 The Root Pallet

The root pallet is utilized by the root network of the architecture. It acts as a central registry, primarily responsible for the management of registered TLD networks. The pallet accomplishes this by mapping the names of various TLDs to the connection details required to connect to the specific TLD network. This connection information is represented by *chain specifications* in the Substrate ecosystem as outlined in Table 4.1.

Despite Substrate utilizing the chain specification structure, the discovery architecture design is flexible. This is because it allows for the storage of any type of connection details for any blockchain network built using any arbitrary framework. This is possible because the objective of the architecture is to facilitate inter-blockchain

[2]https://docs.substrate.io/reference/frame-pallets/
[3]https://github.com/paritytech/substrate/blob/master/bin/node-template

Table 4.1: Structure of a Substrate Chain Specification

Section	**Description**
`name`	The human-readable name of the chain.
`id`	A unique identifier for the chain.
`chainType`	Declares the type of the chain (*Local*, *Development*, or *Live*).
`bootNodes`	A list of addresses of the boot nodes of the network.
`genesis`	The genesis information containing the initial state of the network.

network discovery, but the connection process is delegated to the client, so no special logic is required to be implemented for the connection details of different blockchain platforms.

The pallet performs the mapping between TLD names and their connection details through *extrinsics* — the terminology used by Substrate for blockchain transactions. Specifically, a *register_tld* extrinsic is defined for the registration of TLD networks. It contains logic for verifying whether the specified TLD is already managed by a network. If a TLD is found to be available after a registration request for it is submitted, the pallet marks the requesting network as the TLD manager and maps them together in the on-chain storage.

4.1.2 The TLD Pallet

The networks registered in the root network as managers for a specific TLD must run the TLD pallet, which provides all the functionality needed by a TLD network within the architecture. This pallet manages the registration of domains, mapping them to the connection information of their associated networks, and ensures that they actively maintain the network through the incentive mechanism.

For each domain registered within a TLD network, the pallet stores a `Domain`

`Information` structure outlined in Table 4.2. The structure contains information for connecting to the discoverable network — such as the chain specifications in the case of a substrate network, it also contains an identifier for the account that registered the domain for the network, a boolean value indicating whether the domain name is still active or has been relinquished by the network, and a list of identifiers for nodes from the discoverable network referred as *maintainers*. These nodes are utilized as resources for the maintenance and upkeep of the discovery architecture. Each registered network is required to provide a list of maintainer nodes in exchange for benefiting from the services of the architecture.

Table 4.2: Structure of `Domain Information`

Field Name	**Description**
`Creator`	Contains the identifier of the account that claimed the domain.
`Chain Specifications`	The network connection information.
`Maintainers`	The list of identifiers of maintainer nodes supplied by the network.
`Available`	A boolean value indicating the domain's availability.

While the storage of a maintainer list establishes their relationship with a specific domain, it is not enough for efficiently implementing the mandatory participation incentive mechanism described in Chapter 3. This is because the mechanism will often find that a maintainer disconnected and then proceed to retrieve its associated domain for further processing. However, with the current implementation, this will require the iteration over all domains stored within the network and their maintainer lists to verify whether the disconnected maintainer is part of the list or not. To circumvent this, the pallet maintains an inverse map of maintainers mapped to the domain they are associated with. Leveraging this, a node identifier can be more efficiently queried as the key of this map with the associated domain as its value

in constant time. Conversely, the time complexity of iterating through the domains would scale linearly with the number of domains registered in the TLD network.

Algorithm 1 Mandatory Participation

Require: *cached_nodes*, *current_participants*, *maintainer_map*, *revoke_domain_tx*

```
disconnected_nodes ← ∅
for node_id in cached_nodes do
    if node_id ∉ current_participants then
        disconnected_nodes ← disconnected_nodes ∪ node_id
    end if
end for
for node_id in disconnected_nodes do
    domain ← maintainer_map[node_id]
    if domain ≠ ∅ then
        revoke_domain_tx(domain)
    end if
end for
cached_nodes ← current_participants
```

Using the maintainer mappings implemented above, the mandatory participation mechanism (shown in Algorithm 1) enforces that they stay connected to the network and contribute to its maintenance. This is done through a combination of transactions and off-chain workers for the following reasons:

1. The network must be polled at regular intervals to verify the presence of nodes, ensuring compliance with the mandatory participation requirements.

2. Blockchain transactions must be deterministic. Since polling peers for their presence involves network calls that are inherently non-deterministic in execution time, such operations cannot be included within a transaction.

Furthermore, a polling strategy is employed instead of an event-driven one. This is because the latter relies heavily on node honesty. Nodes have complete control over the events they emit which enables them to send fraudulent events or withhold them. This means that a node can potentially disconnect from the network without notifying it — either maliciously or unintentionally due to network conditions — by broadcasting a disconnection event. Considering all previous justification, off-chain workers poll the network at regular intervals and execute pre-defined logic to verify maintainer participation.

Concretely, off-chain workers keep track of active nodes by maintaining a local cache that stores a list of all their identifiers. As the worker periodically polls the network at a pre-defined period of a few blocks, it compares the list of currently connected nodes to the one stored in the cache. As it iterates through the list, it appends new identifiers to the cache to ensure they can be tracked later. Similarly, if an identifier is present in the cache but not the active list, this indicates that the node has disconnected from the network. The worker then queries the maintainer-domain inverse map to fetch the domain associated with the disconnected node if it is present. If the query successfully returns a domain, that serves as a confirmation that the disconnected node was a maintainer associated with the domain, and the associated network's ownership of the domain is revoked.

Domain ownership revocation is entirely handled by the *revoke_domain* extrinsic

which is called by off-chain workers for domains that fail to contribute to the architecture. This is because the revocation process requires the modification of the domain information entry which is stored on the chain and can only be updated through transactions. The transaction submitted by the off-chain worker revokes the domain ownership by setting its status to be available so that other networks can claim it. Additionally, the connection information is overwritten to an empty field such that the record cannot be used for the discovery of the excluded network.

In addition to the domain revocation functionality, this pallet exposes three further extrinsics for domain management: *register_domain*, *amend_chainspec*, and *transfer_domain*. The *register_domain* transaction allows networks to acquire available domains. Once a request is received, the system verifies the domain availability, and if confirmed, it proceeds by mapping the domain to the connection details of the network supplied in the transaction. To accommodate changes in network configuration, the *amend_chainspec* transaction allows domain owners to update their stored connection information. Once submitted, the transaction first verifies that the submitter is the owner of the domain by ensuring that their account identifier matches with the one that registered the domain. Once ownership is verified, the connection information on the chain is overwritten by the one supplied in the transaction. Finally, if a network decides to transfer its domain to another network, the transfer is initiated by submitting a *transfer_domain* transaction. Once received, an internal request is stored on chain for the transfer after ownership is verified. However, the transfer process is not executed unless the target network formally accepts the transfer by submitting a *transfer_domain_accept* transaction specifying the specific domain name

to be accepted.

4.1.3 The Asset Discovery Pallet

Building on the implementation of the network discovery infrastructure discussed so far in this section, this pallet extends the available functionality by enabling the cross-network discovery of assets. The objective of this pallet is to fetch the domains associated with the assets that the inquiring clients are interested in. So far, the architecture design achieved its scalability goals by logically distributing the storage of network information across multiple blockchain networks. However, since this pallet aims to only fetch domain names, it is deployed directly to the root network to mitigate unnecessary queries that add latency to the request. While this might initially contradict the scalability-oriented design choices outlined in Chapter 3, the following justifications support this approach:

1. Storing a mapping between domains and the assets they provide requires significantly less storage compared to storing connection information directly.

2. The number of networks interested in exclusively making themselves discoverable is expected to be far larger than the number of networks making their assets discoverable.

Accordingly, this pallet manages the the required information for asset discovery directly on the chain of the root network.

Each asset discovery request provides a list of *asset providers*, which are domains that provide that specific asset. To become asset providers, networks must voluntarily

submit a request specifying the assets they wish to make discoverable through the architecture. This registration is a two-phase process: 1. on-chain request submission 2. off-chain validation and finalization

On-chain Request Submission

Asset providers initiate the registration process by submitting a *register_asset_for_domain* extrinsic which is exposed by the pallet. This transaction contains the hashed asset identifier (hashed using a previously agreed upon algorithm) and the domain that provides the asset. Using this information, the transaction constructs a `Pending Request` whose structure is shown in Table 4.3, and stores it on the chain. The request is appended to a request queue and is stamped with a configurable *lifetime* period so that requests are regularly removed to prevent cluttering on-chain storage. A submitted request is deemed expired if the current block number exceeds its lifetime.

Off-chain Validation and Finalization

Similarly to the incentive mechanism discussed earlier, asset registration requires the validation of information that is not directly available on the chain. Specifically, the following two pieces of information must be retrieved from a provider's TLD network and verified when an asset registration request is processed:

1. The entity submitting the transaction must be an authorized representative of the provider network. This is verified by matching the submitting account's identifier with the one that registered the network domain.

2. The domain specified in the request must exist in the respective TLD registry and be active.

Since the registration request must first be initiated by submitting a transaction that creates a request and appends it to an on-chain queue, there must be a common link that can access both on-chain and off-chain information.

Table 4.3: Structure of `Pending Request`

Field Name	**Description**
`Requester`	Contains the ID of the request initiator.
`Domain`	The domain name specified in the transaction initiating the request.
`Asset Hash`	The hash (using an arbitrary agreed upon algorithm) of the asset identifier.
`timestamp`	The block number at which the request was submitted.

To process the pending requests, off-chain workers continuously poll the queue for new requests being appended. Upon retrieving a request, the worker first parses the domain to discern the associated TLD. It then queries the local chain state for the connection information of the TLD network. Using this information, the worker connects to the TLD network, interacting with it through its RPC interface. For an external client to access information on the chain over RPC, it must provide a valid *storage key*. In substrate, this key is calculated by first encoding the domain using the Simple Concatenated Aggregate Little-endian (SCALE) encoding, and then hashing it using the Blake128 Concat algorithm [74].

Once calculated, this key is used by the worker as a part of an RPC request to query the TLD network for the domain's information which is represented by the `Domain Information` structure shown in Table 4.2. If the domain exists, the TLD network responds with this structure containing all relevant data. The worker then

utilizes this data to verify that: 1. The account identifier stored in the pending request matches with the one in the fetched domain information 2. The domain information indicates the status of the domain as available. Once the domain is verified, the worker submits a *submit_verified_domain* transaction, which updates the chain to list the domain as one of the asset's providers. The complete asset registration process is outlined in Algorithm 2.

Finally, this pallet must integrate a part of the incentive mechanism so that its full effect is achieved. Since domains are stored in their respective TLD networks, they are not directly accessible by the asset discovery pallet. Consequently, when the incentive mechanism updates the domain status to inactive, the root network is not notified. This means that if a domain had already registered itself as a provider for a certain asset, the record will remain functional even if the domain ownership has been revoked.

To address this, off-chain workers periodically poll the TLD networks associated with all registered providers. However, iterating through registered assets to find their list of providers is inefficient and leads to redundant availability checks for the same provider unless a tracking mechanism is employed. To resolve this, as shown in Algorithm 3, an inverse mapping between providers and assets is maintained on the chain. Furthermore, since regularly iterating through all providers at once is inefficient, the workers perform the polling in batches. This is done by maintaining an index on-chain which represents the latest verified provider. In this manner, off-chain workers incrementally iterate through domain batches to verify their active status. If a domain is found to be inactive, the workers remove their asset provider

Algorithm 2 Asset Registration

Require: *request_queue*, *asset_providers*

Input: Transaction with *domain* and *asset_hash*

pending_request ← {*requester_id*, *domain*, *asset_hash*, *timestamp*}

request_queue ← *request_queue* ∪ *pending_request*

while processing blocks **do**

Remove expired requests from *request_queue*

for *request* in *request_queue* **do**

tld ← Extract TLD from *request.domain*

tld_info ← Query chain for *TLD* connection info

storage_key ← Blake128Concat(SCALEencode(*request.domain*))

domain_info ← Query *tld_info* with *storage_key*

if *requester_id* == *domain_info.owner* **and** domain is active **then**

submit_verified_domain(request.domain, request.asset_hash)

asset_providers[request.asset_hash] ← *request.dom- ain*

end if

end for

end while

records from the chain.

Algorithm 3 Provider Verification

Require: *asset_providers*, *provider_assets*, *last_processed_provider*

```
Initialize last_processed_provider on-chain if unset
provider_batch ← Fetch providers starting from last_processed_provider
for provider in provider_batch do
    Verify provider availability via TLD network
    if provider is inactive then
        assets ← provider_assets[provider]
        for asset in assets do
            Remove provider from asset_providers[asset]
        end for
    end if
    last_processed_provider ← provider
end for
```

4.2 Decentralized Trust Anchor Generation Implementation

As discussed in the system design in Chapter 3, the discovery architecture needs a decentralized trust anchor that can be used for data verification or certificate generation. To this end, the practical viability of offloading the generation of such trust anchors while maintaining their decentralized nature is shown through this implementation through achieving a decentralized sequencer network for zero knowledge rollups and integrating it with CRS generation ceremonies.

The core implementation[4] of the ZKRollup and the CRS generation schemes is written in Golang due to its simplicity while providing high performance and strong concurrency support. Furthermore, the implementation utilizes well-established libraries like `libp2p` for peer-to-peer communication support and `go-ethereum` to simulate the Ethereum Virtual Machine (EVM) and integrate with L1. Overall, the system is separated into modular components that interoperate with one another including the P2P, consensus, state management, and EVM execution modules. The detailed implementation of these modules is discussed in the rest of this section.

4.2.1 Sequencer Decentralization

At the core of any decentralized network lies peer-to-peer networking to facilitate communication between all the independent nodes. The P2P layer of the rollup is implemented using the `libp2p` library [75], which provides common functionality like peer discovery, connection management, and message broadcasting. When a rollup node is initiated, it is assigned a random unique identifier and is bound to a port on the host system. Furthermore, for the nodes to establish a sequencer network, they must first discover all other nodes participating in it. Instead of relying on a centralized registry to manage this, the network leverages the Kademlia Distributed Hash Table protocol [76], allowing autonomous node discovery without the need for a central server to initiate nodes into the network. Since nodes send and receive different types of messages, the implementation includes three distinct protocols to handle transactions, batch information, and consensus.

[4]https://github.com/wiiatcrs/wiiatzk

The consensus of the decentralized sequencer network is implemented on top of the P2P layer. Specifically, the system implements a Practical Byzantine Fault Tolerance (PBFT) consensus mechanism to enable nodes to reach agreement on the ordering of transactions within each batch. The consensus module manages the consensus state, leader rotation, and consensus message processing. Since PBFT has three distinct stages which require different information, a unified structure, shown in Table 4.4, is used for consensus messages to ensure the flexibility of the implementation, where the `MessageType` field specifies the stage of the consensus and only the fields relevant to that stage are populated. Concretely, the `PrePrepare` message type is used by the leader of the consensus round to propose a batch. Follower nodes then respond with `Prepare` messages to indicate the validity of the message, and then follow with a `Commit` message to finalize the batch once enough `Prepare` messages have been collected from other nodes.

Table 4.4: Structure of `ConsensusMessage`

Field Name	**Description**
`Type`	Type of the message (Preprepare, Prepare, or Commit).
`View`	Current view number.
`Sequence`	Sequence number for the consensus round.
`BatchHash`	The hash of the proposed batch.
`NodeID`	The identifier of the node from which the message originates.
`Timestamp`	The timestamp at which the message was sent.
`Batch`	The proposed batch (only included in Preprepare).

After each successful round of consensus, a new batch containing transactions whose order is agreed on by the nodes is produced. The production of these batches causes a transition in the rollup's state, which is managed by the state management

module. At its core, the module organizes the state as a Merkle Patricia Trie, which is a data structure commonly used in blockchains — such as Ethereum, for efficient state verifications. This is because any modification to the rollups state will be reflected in the root of the tree, known as the Merkle root, enabling immediate detection of state tampering. Furthermore, it integrates with a `StateDB` interface, which acts as an adapter to translate EVM operations into concrete state changes in the rollup.

The translation of EVM operations is required because smart contracts are deployable on the rollup. This is achieved through the EVM execution layer, which provides a simplified yet functional environment for executing Solidity smart contracts. It allows contracts to be deployed to a unique address calculated based on the deployer's address and nonce. The contract's bytecode is then stored in the rollup's state where its functionality can be utilized through transactions. Executed transactions result in state updates such as token transfers between accounts which are reflected in the rollup through the previously discussed `StateDB`.

The sequencer logic integrates the P2P networking layer, the consensus mechanism, the state management module, and the EVM execution layer into a cohesive workflow. Its primary function is to process user transactions, participate in batch finalization, and ensure the resulting state is executed and anchored to Layer 1. Each sequencer maintains a local transaction pool where incoming transactions undergo validation before being included in a batch. This includes:

1. **Signature validation** to confirm that the transaction has been submitted by the authorized account for the operation.

2. **Nonce validation** to ensure that the submitted nonce value is greater than

the current one.

3. **Balance validation** to verify that the sender's balance can cover the transaction value.

Periodically, the sequencer attempts to create a new batch by collecting transactions from its pool. Its subsequent actions depend on its role in the current round of consensus:

1. If the node is the *leader*, it constructs the batch and proposes it by broadcasting a `PrePrepare` message.

2. If the node is a *follower*, it simply waits for the leader's proposal. Once received, it validates the batch and responds with a `Prepare` message and eventually a `Commit` message to finalize the batch.

Once a batch is finalized through consensus, it undergoes further processing. A zkSNARK proof is generated with an option for disabling proof generation, in which case dummy proofs are generated, and the batch is added to the state history and broadcast to discovered peers. Finally, the batch is submitted to the verifying smart contract on Layer 1 to anchor the batch.

4.2.2 Decentralized CRS Generation

The CRS generation ceremony presented in this thesis has two distinct implementations that correspond to the two variants discussed in Chapter 3. Although both variations utilize the decentralized sequencer network as their underlying infrastructure, they differ in the degree of decentralization achieved. Despite their differences,

both schemes employ the Powers of Tau protocol implemented by Circom for CRS generation. This is preferred to implementing generation from scratch to ensure the security of the protocol.

Semi-decentralized Generation

This scheme employs a smart contract, the `CRSManager`, that is deployed on the L2 rollup and acts as the trustless ceremony coordinator. It orchestrates the CRS ceremony by managing participant registration, maintaining a queue for contribution order, and storing the intermediate CRS as contributions are applied. Moreover, it prevents participants from changing their contributions based on others' by implementing a commit-reveal scheme. Contributions are sequentially applied with each participant transforming the transient CRS using Circom's Powers of Tau implementation and submitting the result to the contract. Once all contributions have been made, the last participant finalizes the round by calling the `finalizeCRS` function, storing the final CRS on the ledger.

Both the smart contract and the sequencer nodes complement one another in this scheme. This is because sequencers have the necessary cryptographic implementations for generating the random scalar contributions r_i and hashing it before submitting it to the smart contract. Furthermore, all interactions with the contract are facilitated by a client running on the node. This client is used to monitor the ceremony state by querying the contract and to submit the hashed commitments of the sequencers.

The workflow followed in this CRS generation scheme begins with initialization. This includes deploying the smart contract to the L2 network with paramters that

dictate ceremony characteristics like `roundDuration` dictating the duration of each CRS generation round and `maxParticipants` specifying the number of participants that can take part in the ceremony. The nodes then register themselves as participants for the current round by sending a registration transaction within a specified deadline. Once registered, participants submit their random scalars as hashed commitments to the smart contract. When all commitments have been made, participants start applying their contributions to the CRS sequentially in a predetermined order. After applying their contributions, participants submit the transformed CRS along with the revealed scalar r_i so that the contract can verify that it matches the commitment previously submitted. Finally, the contract finalizes the round, storing the final CRS and preparing for the next round of generation.

Fully Decentralized Generation

Instead of relying on a smart contract to coordinate the ceremony, this scheme leverages the pre-existing sequencer consensus. Using the PBFT implementation discussed earlier, the participants maintain the state of the ceremony directly between one another. The ceremony state, `CRSCeremonyState` as shown in Table 4.5, is an object added to the consensus state forcing all sequencers to agree on it. This state includes the current epoch, the list of participants, the intermediate CRS, and the ceremony status.

Since the ceremony state is integrated in the consensus state, any contribution made by a participant is treated as a state transition and must be agreed on through consensus. During a node's turn to contribute, it generates a random scalar r_i and

applies its transformation to the intermediate CRS. The node then proposes a new ceremony state which must be verified by other participants to ensure that the intermediate is not replaced by a different one by a malicious participant. Once the validity of the intermediate CRS is verified by a supermajority of the nodes, the new state is accepted and the ceremony advances to the next participant. After all contributions have been made, the CRS is finalized and a new round is initiated for the generation of another.

Table 4.5: Structure of `CRSCeremonyState`

Field Name	**Description**
`EpochNumber`	The current epoch of the CRS generation ceremony.
`Participants`	The nodes participating in the current ceremony.
`CurrentStep`	The index indicating whose turn it is to contribute to the CRS.
`IntermediateCRS`	The incomplete CRS to which participants must contribute.
`Completed`	Indicates the completion status of the ceremony.

Chapter 5

Evaluation

This chapter presents the evaluation results of the two core components of this thesis: the decentralized network discovery architecture and the decentralized trust anchor generation scheme. In this evaluation, the performance under heavy load, scalability, and resilience under adversarial conditions are all assessed according to the functional expectations of each component. Furthermore, all experiments were conducted on a machine equipped with an Intel Xeon Silver 4216 processor with 64 CPUs, 128 gigabytes of RAM, and 500 gigabytes of NVMe storage.

5.1 Methodology

The evaluation of the decentralized discovery architecture was conducted using two Golang tools that complement each other. The first is a client application created to facilitate the resolution of requested domains. This client implements the registration and resolution protocols detailed in Chapter 3 for both network and asset discovery,

utilizing them to interact with the architecture's RPC interface through a Golang library [77] to simplify the process. The client also implements a benchmarking feature that allows for the generation of a high volume of concurrent requests to be sent to the architecture and measure their latencies. Furthermore, a separate script was implemented to orchestrate the evaluation process by deploying the architecture including the root network, the TLD networks, and the test networks, which all run within Docker containers. After deployment, the script populates the architecture with data, executes stress tests, and then prepares another evaluation configuration to be executed.

Similarly, the decentralized trust anchor generation scheme was evaluated using a Golang script. The script initializes testbeds with different configurations to measure the transaction throughput of the sequencer and the duration of the CRS generation ceremonies. Furthermore, the scheme's robustness against malicious actors was tested by implementing a probabilistic attack model. This allows nodes to be assigned probabilities for randomly exhibiting malicious behavior by disconnecting from the network and submitting invalid transactions, or simply maintaining honest behavior without interrupting service.

5.2 Decentralized Network Discovery Architecture

5.2.1 Performance Evaluation

The complete evaluation of the architecture was fully managed by the aforementioned scripts to ensure the reproducibility of the results. The topology employed

consists of a single network in the root layer, which by design never changes regardless of the topology implemented. Additionally, two networks were deployed in the TLD layer, and two networks were part of the discoverable layer. During the experiment, the system was subjected to concurrent request loads to study their effect on the response latency experienced by clients as the number of participating nodes in each network varies. This was achieved by varying the node configurations of all networks such that the networks operate with 4, 8, 10, and 15 nodes in separate scenarios. For each configuration, a total of 10,000,000 requests were sent to the architecture, with concurrent request loads varying from 50,000 requests per second (rps) to 130,000 rps.

The results of the evaluation experiments are presented in Figure 5.1. The performance of the system under the lowest request load of 50,000 rps demonstrates extremely low latencies for responses, as shown in Figure 5.1(a), with a median delay of 1 millisecond. Furthermore, as the request rate increased to 100,000 rps and 110,000 rps, the median latency, as shown in Figures 5.1(b) and 5.1(c), also increased across all configurations. Despite this increase, the highest median latency observed was 4 milliseconds, with the highest impact on the 4-node and 8-node configurations. Unsurprisingly, this shows that the architecture experiences higher latencies, with higher request rates distributed across a lower number of nodes. However, the 130,000 rps load overwhelmed the 4-node configuration, having a median response time of 227 milliseconds as shown in Figure 5.1(d). This follows the trend demonstrated across the experiments, where smaller networks are more strongly affected by increasing concurrent request load. Unlike the 4-node configuration, all other config-

urations handled the increased load without issue, experiencing a maximum median latency of 5.5 milliseconds. This uncovers the scaling potential of the architecture which can be achieved by minimally increasing the number of participating nodes.

5.2.2 Storage Evaluation

The hierarchical design of the architecture was chosen to increase its scalability. Since storing connection records is a critical function of the architecture, storage costs must be considered. Naturally, instead of storing all records in a single network, they are logically distributed across networks in different layers. When considering a single network, assuming the number of stored domains grows at a rate `domain_growth_rate` per month, with an average size of `string_size` megabytes for each connection string, the storage growth rate can be modeled as:

$$storage\_growth\_rate = domain\_growth\_rate * string\_size$$

If a growth rate similar to the current Internet infrastructure is assumed, the number of registered domains would increase by approximately 244,000 daily [78]. This paired a connection string of size 767 kilobytes—which is the smallest size for Substrate connection strings measured during the experiments—means that a monolithic architecture comprised of a single blockchain network would require over 187 gigabytes of new storage daily.

However, since the architecture presented in this thesis has a hierarchical design, the storage requirements are distributed across the TLD networks, which are independent of each other. In the best case, the domains would be equally distributed across all TLD networks. This would mean that, assuming the number of TLDs deployed

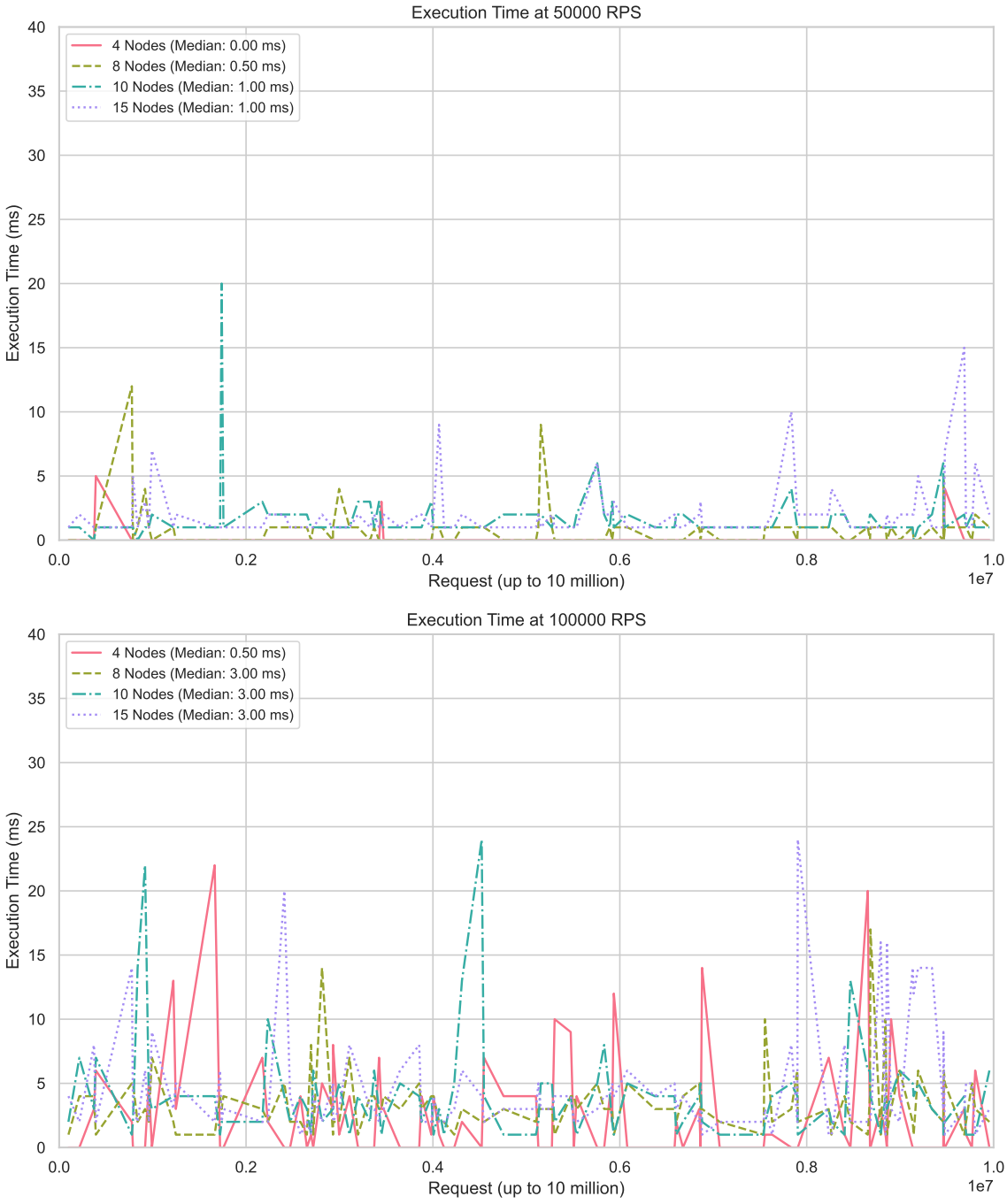

Figure 5.1: Discovery Architecture Performance Evaluation Results [1]

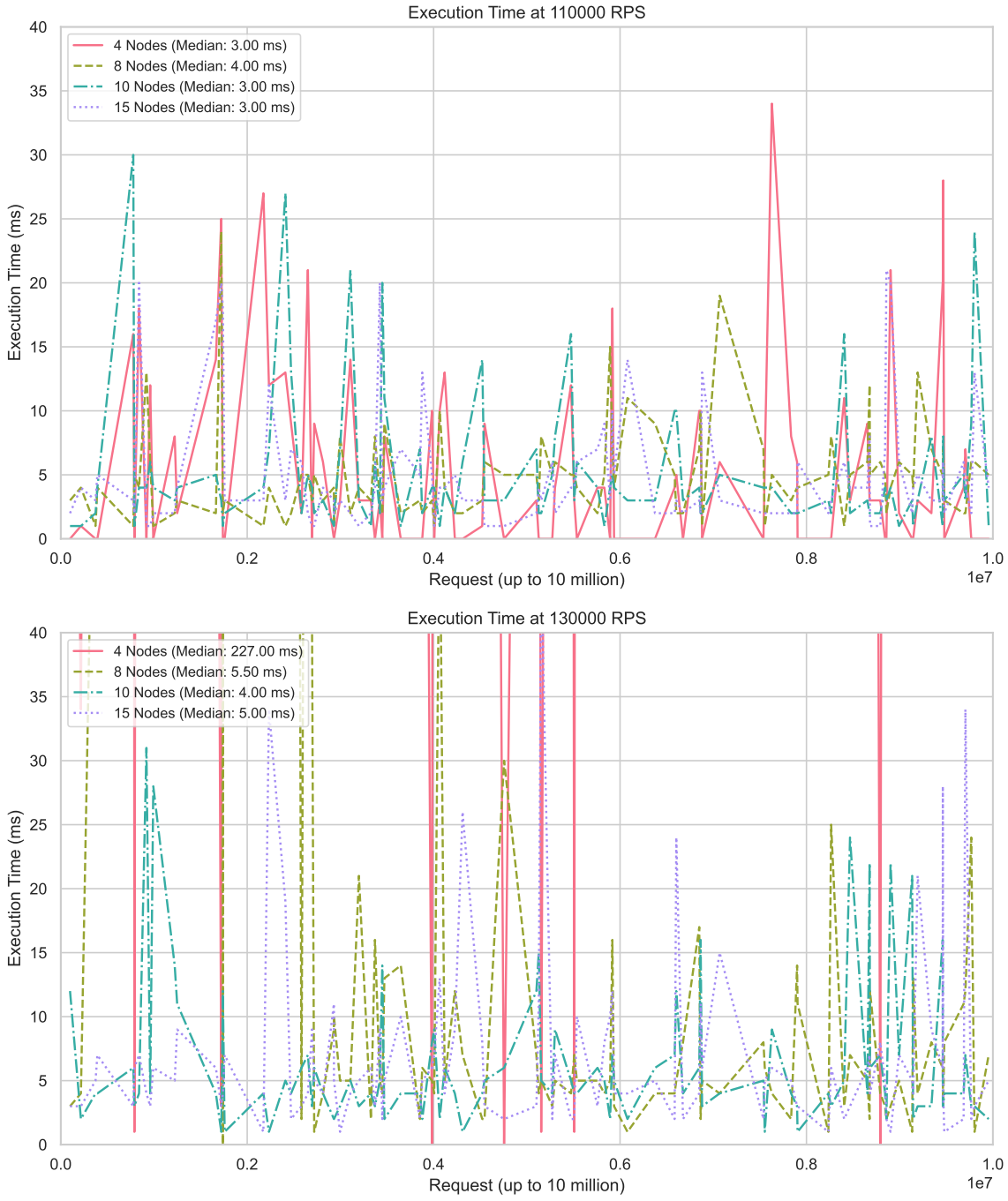

Figure 5.1: Discovery Architecture Performance Evaluation Results [1]

for the architecture is n, and assuming an equal distribution of domains across these TLD networks, the storage growth rate per TLD network will be reduced by a factor of n compared to the single-network solution. In practice however, the distribution is often skewed towards a specific network. If the architecture exhibits trends similar to current Web 2.0 DNS with the largest TLD holding 36.48% [79] of all domains, its storage requirements would grow at 68.4 gigabytes per day. This achieves a reduction of 63.52% for the most utilized network, reducing the load even further for the rest of the TLD networks.

5.3 Decentralized Trust Anchor Generation

5.3.1 Decentralized Sequencer Network Throughput

The transaction throughput of the decentralized sequencer network is evaluated by first launching a testbed that consists of three nodes on the host machine. The network was then subjected to batches that had sizes ranging from 20,000 to 200,000 transactions. For each batch, the transactions were distributed across multiple senders with sequentially incrementing nonces to simulate real-world conditions. Since the transaction content can affect the throughput results obtained if simpler transactions were submitted in different runs, seeded randomness is utilized for transaction generation. This means that for each run, the generated transactions and their order will be the same, ensuring that the obtained throughput values are purely dependent on the underlying infrastructure.

The results obtained from the evaluation are shown in Figure 5.2. As the number

of transactions increased, the throughput initially peaked to 42,500 transactions per second (tx/sec), stabilizing later at around 41,000 tx/sec as batch sizes increased. This behavior indicates that the sequencer network started to be overloaded as the batch sizes increased to 200,000 transactions. However, although the network was overload, the stable throughput achieved at 41,000 tx/sec is still close to the peak throughput. Furthermore, the throughput achieved by the sequencer network exceeds the average throughput of 7,000 tx/sec handled by Visa and its theoretical limit of 24,000 tx/sec [17], showing that it can easily accommodate real-world workloads.

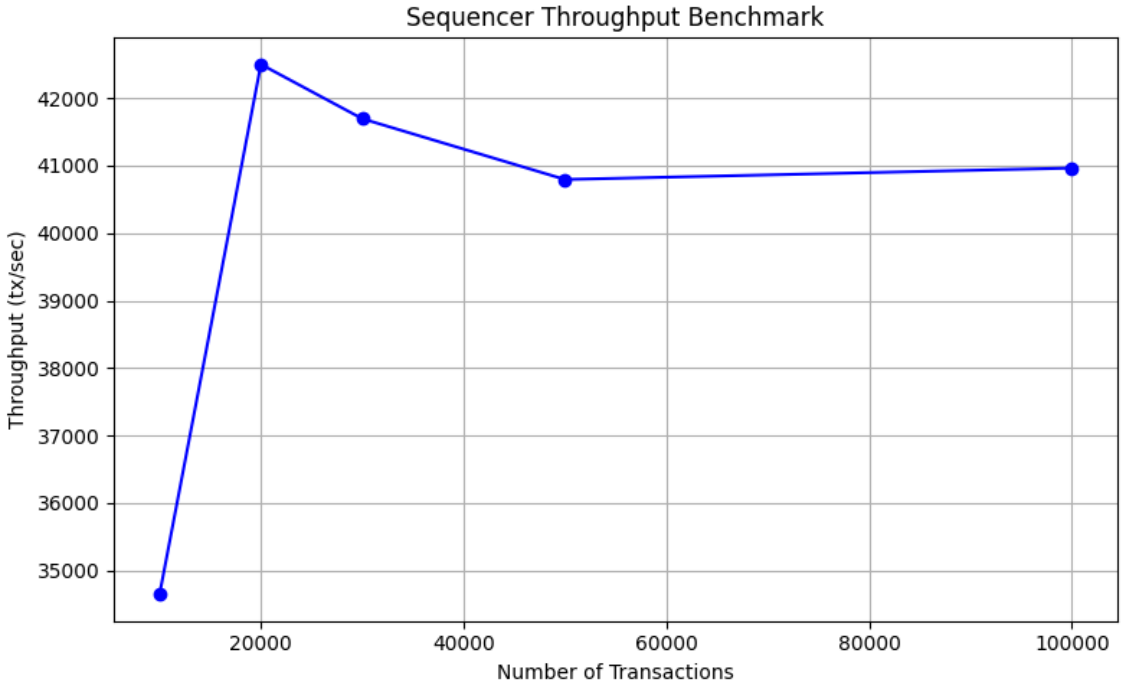

Figure 5.2: Decentralized Sequencer Network Transaction Throughput

5.3.2 Decentralized CRS Generation

The CRS generation was evaluated based on the ceremony duration, resilience against attacks, and the final CRS size. The duration was measured under both normal and adversarial conditions using a testbed with two network configurations: one with 6 participants and another with 12. For each configuration, the CRS was generated at three different power sizes (8, 10, and 12). Furthermore, to ensure the reliability of the results, 10 independent iterations of the ceremony were executed per configuration to provide an average duration under varying network conditions.

In addition to the duration, the resilience of the system against adversarial behavior was evaluated using a probabilistic attack model. For each iteration, a random subset of nodes were marked as adversaries. During their turn, each adversarial node randomly decides whether to act malicious during the round or not. If malicious behavior is decided by the node, it can choose one of the following two attack vectors:

1. Random disconnection: The node simply disconnects during its turn to simulate a network failure or the node intentionally exhibiting timeouts as a denial of service attack.

2. Invalid contribution: The node adds an invalid contribution to the CRS during its turn. This can be done by either replacing the CRS completely to disregard previous contributions or by the contribution getting corrupted.

The evaluation results are shown in Figure 5.3. As expected, the mean ceremony duration increases with higher power sizes, rising from 47 seconds to 98.7 seconds for the 6 participant configuration and from 113.2 seconds to 275.4 seconds for the 12

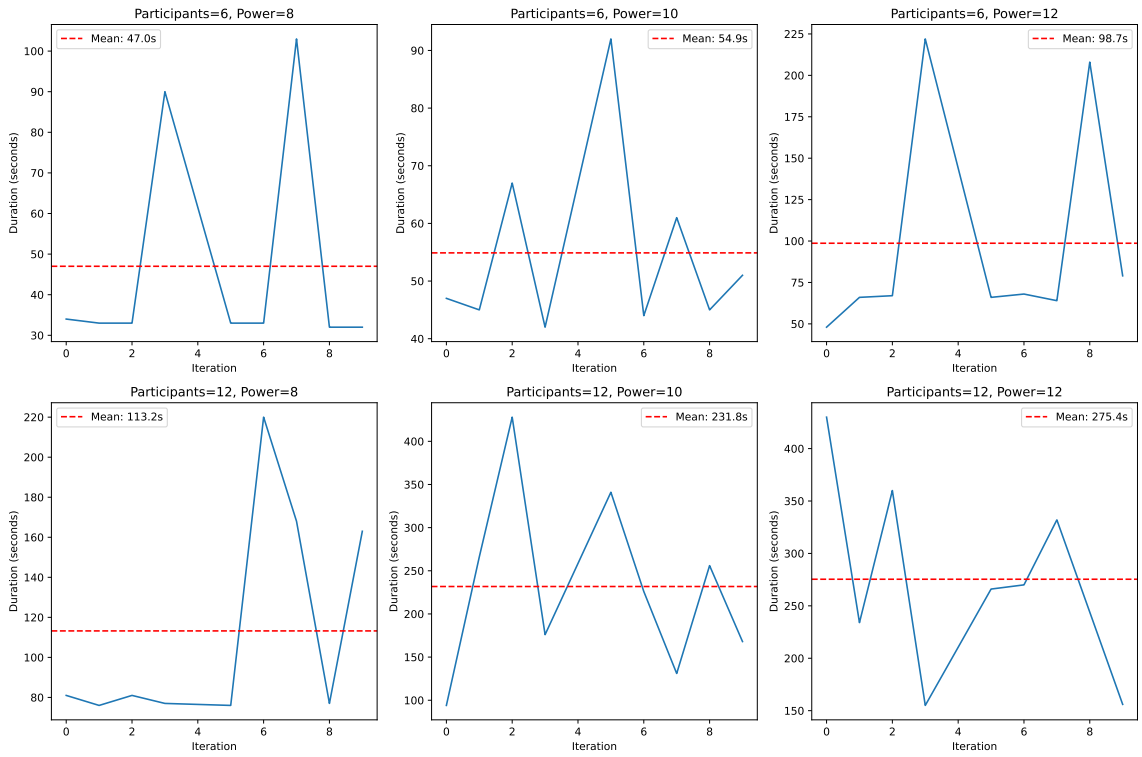

Figure 5.3: Decentralized CRS Generation Ceremony Durations

participant configuration. This is because generating CRS strings with higher power sizes requires more computation, translating into longer ceremonies. Furthermore, the results show that the mean duration is higher for the 12-node configuration than the 4-node one. This is because as the number of participants increases, the time required to achieve consensus will increase, subsequently increasing the ceremony's duration.

For each participant count and power size combination, no consistent trend was observed between individual iterations, since malicious behavior was randomly determined per run. Depending on the chosen attack vector and how persistent the adversarial node is, it can take some iterations longer to recover than others. Never-

theless, the ceremonies succeeded in all cases because the system either temporarily excludes nodes experiencing failures, or permanently bans a participant from future ceremonies depending on their reputation scores. This shows that while adversarial behavior may increase the ceremony duration, it does not compromise its integrity or its completion.

5.3.3 Generated CRS Size

The size of the generated CRS was evaluated as a function of both the power size and the number of participants in the ceremony. As shown in Figure 5.4, while the size of the generated CRS shows a slight linear growth with an increasing number of participants, it is more strongly dependent on the power size. This means that systems can achieve higher security through larger participant sets without substantially increasing CRS size, provided that the associated increase in consensus time remains within acceptable bounds for the target application.

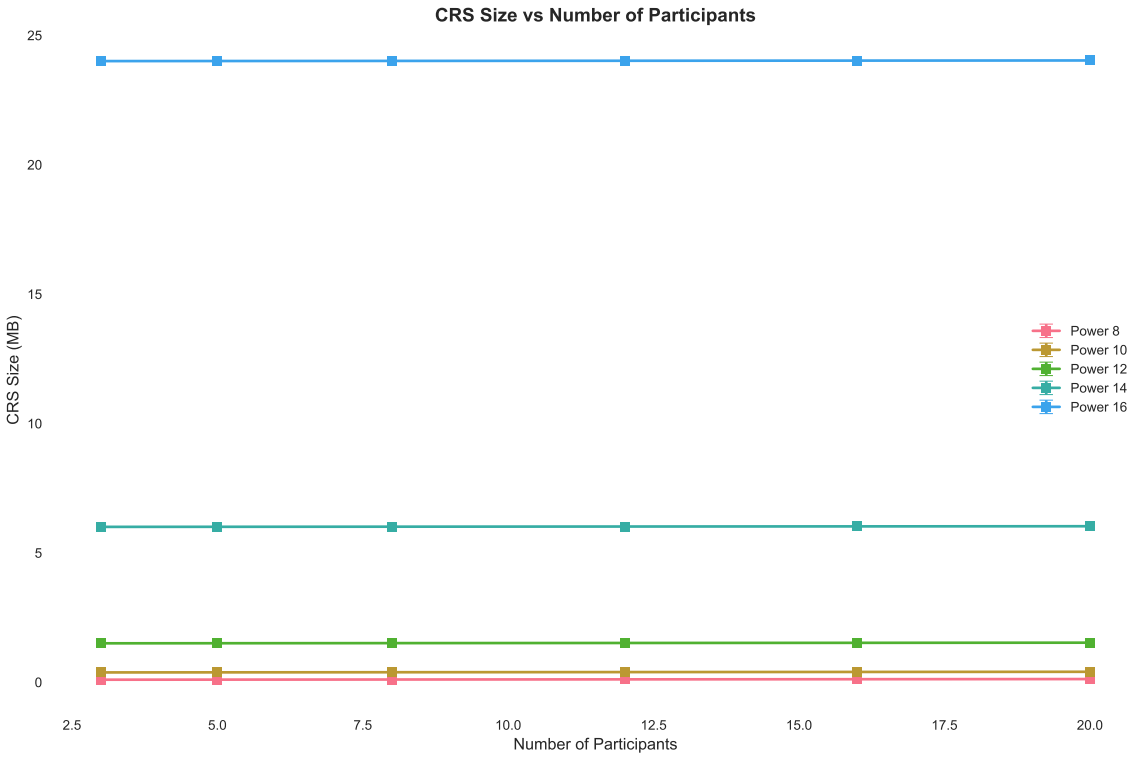

Figure 5.4: CRS Size Correlation with Number of Participants

Chapter 6

Discussion

6.1 Decentralization

One of the main goals of the components presented in this thesis is their decentralization. This is because blockchain networks are inherently decentralized at different levels, so the solutions presented in the thesis must also be decentralized to not undermine the decentralization of the communication chain. The discovery architecture achieves this through the combination of the layered architecture and the incentive mechanism leading to self-sufficiency while ensuring all participating nodes are supplied by beneficiary networks.

Although the presented design promotes decentralization, there may be pitfalls when it comes to the implementation. The employed consensus algorithm is critical in ensuring that a blockchain network is not controlled by a central authority. The algorithm utilized in this thesis randomly assigns a node for block production every round. This ensures no bias in node selection, making the scheme completely decen-

tralized. However, it is unclear how other popular algorithms will affect the overall decentralization of the system. For example, if a TLD network uses Proof of Stake for its consensus, the stake distribution across the participating nodes will heavily skew the node selection process, effectively decreasing the decentralization of the system.

Similarly, the decentralization of the trust anchor generation schemes heavily depends on the decentralization of the sequencers they are offloaded to. To avoid the aforementioned potential problems with staking, the implementation used a simple PBFT consensus to achieve a decentralized proof of concept. The CRS generation schemes presented in the thesis have different purposes. The semi-decentralized scheme was presented as a solution that can be implemented even without a decentralized sequencer. This is enabled by having a smart contract manage the ceremony, allowing for any participant to register for the ceremony. However, a critical tradeoff for this scheme is that participants must reveal their secret scalars r_i after committing them to the ceremony. While this does not compromise the resulting CRS, it is more secure to destroy and never reveal them, which is done in the case of the fully decentralized scheme. Because it directly uses the decentralized sequencer network, the consensus manages the ceremony state and nodes are not required to reveal their scalars.

6.1.1 Scalability

The layered discovery architecture presented in this thesis allows for the TLD networks to scale independently of each other. This is crucial because as discussed in Chapter 5, the distribution of domains across TLDs is skewed, which requires different

resource allocations depending other their utilization. Furthermore, having multiple networks participating in the architecture allows for the increase of the architecture's effective validator cap by having their local caps enforced. However, distributing the caps across multiple networks greatly increases the complexity of the governance process.

The scalability is also greatly improved by the CRS generation schemes presented in this thesis. If the ceremonies were to be held by the discovery architecture, it would have to be distributed across the participating networks. However, coordinating the ceremony across networks introduces complexity into the design. This complexity paired with the computation needed for the generation process increases the resource requirements of the architecture. Instead, offloading the generation to Layer 2 solutions reduces the complexity while freeing up resources to be utilized for discovery rather than trust anchor generation.

6.1.2 Dynamic Discovery

The design enables dynamic discovery by allowing networks to voluntarily register themselves into the architecture. The necessary validations are then handled by the nodes in the architecture participating in its consensus. This eliminates the need for a central entity for accepting or rejecting potential registrations, resulting in a faster and more seamless process.

Chapter 7

Conclusion

This thesis introduced two complementary contributions toward advancing blockchain interoperability and achieving the Internet of Blockchains:

1. **Inter-blockchain Discovery Architecture** that is inspired the traditional DNS architecture. The integration of the root and TLD networks into the hierarchical architecture presented in the thesis, the system distributes functionality logically, effectively separating concerns to create a decentralized, scalable, and robust discovery framework. Furthermore, a robust incentive mechanism was designed to ensure the self-sustainability of the architecture. This mechanism requires the beneficiary networks to contribute back to the architecture by providing nodes, fostering a symbiotic relationship between participating networks and the discovery infrastructure. Using this design, the architecture was implemented in Substrate for its evaluation. Consequently, the architecture demonstrated the ability to handle high volumes of concurrent domain resolution queries, achieving extremely low response times while scaling effectively

with the architecture's size.

2. **Decentralized Trust Anchor Generation** through offloading the trusted setup process of zero-knowledge systems to Layer 2 solutions. Specifically, a proprietary zero-knowledge setup with a decentralized sequencer network was designed and implemented to achieve the decentralized trust anchor generation ceremonies. Leveraging a reputation-based incentive scheme, the system mitigates risks of participant unresponsiveness or malicious contributions by excluding them from participation. The ceremonies aim to generate Common Reference Strings in both a semi-decentralized and a fully decentralized manner to accommodate for different modes of operation. Experimental results demonstrate that the rollup maintains high transaction throughput under decentralized operation. Additionally, the CRS generation ceremonies consistently complete within bounded time, even in adversarial settings, and successfully produce secure CRS instances.

7.1 Future Work

Together, both contributions lay the foundation for the establishment of an Internet of Blockchains. The discovery architecture allows for dynamic and decentralized inter-blockchain discovery while the CRS generation schemes provide a decentralized regenerative trust anchor whose generation is offloaded, allowing for the attestation of discovery records. However, while these contributions have been validated by evaluating their implementations, several avenues remain open for further research:

7.1.1 Trust Enhancement and Certification

The current state of the discovery architecture directly sends responses back to the querying clients. While these responses inherit trust from the consensus since the chain is only updated through it, certifying them with verifiable proofs further reinforces integrity. This thesis already laid the foundation for this through the decentralized trust anchor generation schemes that have been discussed. However, using this trust anchor for proof generation is not straight forward. A promising direction is to explore decentralized certificate generation and management schemes to strengthen trust within the system.

7.1.2 Node Accountability and Reputation

Both components of this thesis rely on the participation of independent nodes. Although this helps to increase their decentralization, it can increase the rate of malicious node participation. However, this participation of disparate nodes provides data about their behavior, which can be leveraged as an opportunity to strengthen accountability at a broader scale. Future research should focus on building robust accountability frameworks that maintain a global reputation metric for individual participants, allowing networks to evaluate nodes before on-boarding them.

7.1.3 Enhanced Asset Discovery

The current asset discovery model supports simple queries. However, achieving interoperability at the scale of an Internet of Blockchains will require more complex queries. Incorporating more complex asset characteristics into the queries to enable

more granular filtering of assets is an interesting direction to explore.

7.1.4 Domain Squatting Prevention

As with traditional DNS, domain squatting poses a potential issue. Future directions include designing mechanisms for the verification of network authenticity before they are allowed to claim a domain. This can be done through heuristics to ensure that the network is actively utilizing the domain and is not simply a test network whose sole purpose is squatting.

7.1.5 Scalability of Trust Anchor Generation

The decentralized trust anchor generation schemes presented in this thesis demonstrated resilience against malicious attacks with a bounded ceremony duration. However, their dependence on the consensus mechanism used by the sequencer network can drastically increase the duration as the number of participants increases. Future work should focus on achieving low ceremony durations with a large number of participants while maintaining the resilience of the ceremony and the integrity of the produced anchor.

Bibliography

[1] K. Hassan, A. Sokhankhosh, and S. Rouhani, "Enabling blockchain interoperability through network discovery services," in *2025 IEEE 7th International Conference on Decentralized Applications and Infrastructures (DAPPS)*, Jul. 2025. [Online]. Available: https://doi.org/10.48550/arXiv.2506.16611

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

[3] N. Dashkevich, S. Counsell, and G. Destefanis, "Blockchain application for central banks: A systematic mapping study," *IEEE Access*, vol. 8, pp. 139 918–139 952, 2020.

[4] R. Jabbar, N. Fetais, M. Krichen, and K. Barkaoui, "Blockchain technology for healthcare: Enhancing shared electronic health record interoperability and integrity," in *2020 IEEE International Conference on Informatics, IoT, and Enabling Technologies (ICIoT)*, 2020, pp. 310–317.

[5] H. Xiong, T. Dalhaus, P. Wang, and J. Huang, "Blockchain technology for agriculture: Applications and rationale," *Frontiers in Blockchain*, vol. 3,

2020. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fbloc.2020.00007/full

[6] P. Dutta, T.-M. Choi, S. Somani, and R. Butala, "Blockchain technology in supply chain operations: Applications, challenges and research opportunities," *Transportation Research Part E: Logistics and Transportation Review*, vol. 142, p. 102067, 2020.

[7] 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*, 2019, p. 29–35.

[8] W. Ou, S. Huang, J. Zheng, Q. Zhang, G. Zeng, and W. Han, "An overview on cross-chain: Mechanism, platforms, challenges and advances," *Computer Networks*, vol. 218, p. 109378, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128622004121

[9] G. Wood, "Polkadot: Vision for a heterogeneous multi-chain framework," *White paper*, vol. 21, no. 2327, p. 4662, 2016.

[10] J. Kwon and E. Buchman, "Cosmos: A network of distributed ledgers," https://github.com/cosmos/cosmos/blob/master/WHITEPAPER.md, 2018.

[11] H. Tam Vo, Z. Wang, D. Karunamoorthy, J. Wagner, E. Abebe, and M. Mohania, "Internet of blockchains: Techniques and challenges ahead," in *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Com-*

puting and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), pp. 1574–1581.

[12] 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)*, 2024, pp. 611–618.

[13] G. Wang and M. Nixon, " InterTrust: Towards an Efficient Blockchain Interoperability Architecture with Trusted Services ," in *2021 IEEE International Conference on Blockchain (Blockchain)*. Los Alamitos, CA, USA: IEEE Computer Society, Dec. 2021, pp. 150–159. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/Blockchain53845.2021.00029

[14] Hyperledger Labs. Weaver: Dlt interoperability. (Accessed 5 November 2024). [Online]. Available: https://hyperledger-labs.github.io/weaver-dlt-interoperability/docs/external/introduction/

[15] J. Werth, M. H. Berenjestanaki, H. R. Barzegar, N. E. Ioini, and C. Pahl, "A review of blockchain platforms based on the scalability, security and decentralization trilemma," in *International Conference on Enterprise Information Systems*, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258369909

[16] V. Buterin, "Ethereum white paper: A next generation smart contract & decentralized application platform," 2013. [Online]. Available: https://github.com/ethereum/wiki/wiki/White-Paper

[17] G. Tripathi, M. A. Ahad, and G. Casalino, "A comprehensive review of blockchain technology: Underlying principles and historical background with future challenges," *Decision Analytics Journal*, vol. 9, p. 100344, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772662223001844

[18] V. Nikolaenko, S. Ragsdale, J. Bonneau, and D. Boneh, "Powers-of-tau to the people: Decentralizing setup ceremonies," in *Applied Cryptography and Network Security*, C. Pöpper and L. Batina, Eds. Cham: Springer Nature Switzerland, 2024, pp. 105–134.

[19] IOHK. (2025) Cardano docs. Accessed: 2025-08-03. [Online]. Available: https://docs.cardano.org/about-cardano/introduction

[20] Anatoly Yakovenko. (2018) Solana: A new architecture for a high performance blockchain v0.8.13. Accessed: 2025-08-03. [Online]. Available: https://solana.com/solana-whitepaper.pdf

[21] G. Wang, "Sok: Exploring blockchains interoperability," *Cryptology ePrint Archive*, 2021.

[22] G. Wang, Q. Wang, and S. Chen, "Exploring blockchains interoperability: A systematic survey," *ACM Comput. Surv.*, vol. 55, no. 13s, Jul. 2023.

[23] F. Saleh, "Blockchain without Waste: Proof-of-Stake," *The Review of Financial Studies*, vol. 34, no. 3, pp. 1156–1190, 07 2020.

[24] L. Bach, B. Mihaljevic, and M. Zagar, "Comparative analysis of blockchain consensus algorithms," 05 2018, pp. 1545–1550.

[25] M. Belotti, N. Božić, G. Pujolle, and S. Secci, "A vademecum on blockchain technologies: When, which, and how," *IEEE Communications Surveys and Tutorials*, vol. 21, pp. 3796–3838, 10 2019.

[26] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, S. Muralidharan, C. Murthy, B. Nguyen, M. Sethi, G. Singh, K. Smith, A. Sorniotti, C. Stathakopoulou, M. Vukolić, S. W. Cocco, and J. Yellick, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *Proceedings of the Thirteenth EuroSys Conference*, ser. EuroSys '18, 2018, pp. 1–15.

[27] Parity Technologies. (2024) Substrate blockchain technology | substrate_. [Online]. Available: https://substrate.io

[28] C. Shaw. (2018) The corda platform: An introductory white paper. [Online]. Available: https://r3.com/the-corda-platform-an-introduction-whitepaper/

[29] J. Morgan. (2018) quorum/docs/quorum whitepaper v0.2.pdf at master · consensys/quorum. [Online]. Available: https://github.com/Consensys/quorum/blob/master/docs/Quorum%20Whitepaper%20v0.2.pdf

[30] D. Ongaro and J. Ousterhout, "In search of an understandable consensus algorithm," in *Proceedings of the 2014 USENIX Conference on USENIX Annual*

Technical Conference, ser. USENIX ATC'14. USA: USENIX Association, 2014, p. 305–320.

[31] OpenEthereum. (2021) Aura - authority round. Accessed: July 1, 2025. [Online]. Available: https://openethereum.github.io/Aura

[32] Handan Kilinc Alper. Babe. Accessed: July 1, 2025. [Online]. Available: https://research.web3.foundation/Polkadot/protocols/block-production/Babe

[33] A. Stewart and E. Kokoris-Kogia. (2020) Grandpa: a byzantine finality gadget. Accessed: July 1, 2025. [Online]. Available: https://github.com/w3f/consensus/blob/master/pdf/grandpa.pdf

[34] H. Zimmermann, "OSI reference model-the ISO model of architecture for open systems interconnection," *IEEE Trans. Communication (USA)*, vol. COM-28, no. 4, pp. 425–432, Apr. 1980, iRIA/Lab., Rocquencourt, France.

[35] G. Verdian, P. Tasca, C. Paterson, and G. Mondelli, "Overledger whitepaper," Quant Foundation, Tech. Rep., 2018.

[36] 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," vol. 54, no. 7, Jul. 2021. [Online]. Available: https://doi-org.uml.idm.oclc.org/10.1145/3460287

[37] A. Zamyatin, M. Al-Bassam, D. Zindros, E. Kokoris-Kogias, P. Moreno-Sanchez, A. Kiayias, and W. J. Knottenbelt, "SoK: Communication across distributed ledgers," 2019, publication info: Published elsewhere. Major

revision. Financial Cryptography and Data Security 2021. [Online]. Available: https://eprint.iacr.org/2019/1128

[38] Interledger. (2020) Interledger protocol v4 (ilpv4). Accessed: October 30, 2024. [Online]. Available: https://interledger.org/developers/rfcs/interledger-protocol/

[39] J. Poon and T. Dryja, "The bitcoin lightning network:," Lightning Network, Tech. Rep., 2016.

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

[41] 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.

[42] 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*, 2022, pp. 347–356.

[43] 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)*. IEEE, 2023, pp. 1–14.

[44] R. Belchior, A. Vasconcelos, S. Guerreiro, and M. Correia, "A survey on

blockchain interoperability: Past, present, and future trends," *ACM Computing Surveys (CSUR)*, vol. 54, no. 8, pp. 1–41, 2021.

[45] Bitcoin Wiki. (2016) Hashlock. Accessed: October 30, 2024. [Online]. Available: https://en.bitcoin.it/wiki/Hashlock

[46] ——. (2016) Timelock. Accessed: October 30, 2024. [Online]. Available: https://en.bitcoin.it/wiki/Timelock

[47] IBC Ecosystem Working Group, "Inter-blockchain communication protocol (ibc)," https://github.com/cosmos/ibc, 2020.

[48] Polkadot Wiki, "Cross-chain message passing (xcmp)," https://wiki.polkadot.network/docs/learn-xcm-transport, 2019.

[49] N. Kannengiesser, M. Pfister, M. Greulich, S. Lins, and A. Sunyaev, "Bridges between islands: Cross-chain technology for distributed ledger technology," 01 2020.

[50] 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*, vol. 9, pp. 116 672–116 691, 2021.

[51] H. Montgomery, H. Borne-Pons, J. Hamilton, M. Bowman, P. Somogyvari, S. Fujimoto, T. Takeuchi, T. Kuhrt, and R. Belchior. (2022) Hyperledger cactus whitepaper. Accessed: July 1, 2025. [Online]. Available: https://github.com/hyperledger-cacti/cacti/blob/7bb39576080592919bea0ac89646b32105e1748e/whitepaper/whitepaper.md

[52] P. Lafourcade and M. Lombard-Platet, "About blockchain interoperability," *Information Processing Letters*, vol. 161, p. 105976, 05 2020.

[53] J. Zarrin, H. Wen Phang, L. Babu Saheer, and B. Zarrin, "Blockchain for decentralization of internet: prospects, trends, and challenges," vol. 24, no. 4, pp. 2841–2866.

[54] P. Mockapetris, "Domain names - concepts and facilities," 1987, num Pages: 55. [Online]. Available: https://datatracker.ietf.org/doc/rfc1034

[55] IEEE, "IEEE standard for blockchain interoperability data authentication and communication protocol," 2023, ISBN: 9781504496209. [Online]. Available: https://ieeexplore.ieee.org/document/10108929/

[56] T. E. Foundation. (2025) Ethereum name service. Accessed: July 1, 2025. [Online]. Available: https://docs.ens.domains/web/quickstart/

[57] Namecoin. (2011) Namecoin. Accessed: October 30, 2024. [Online]. Available: https://www.namecoin.org/

[58] C. Patsakis, F. Casino, N. Lykousas, and V. Katos, "Unravelling ariadne's thread: Exploring the threats of decentralised dns," *IEEE Access*, vol. 8, pp. 118 559–118 571, 2020.

[59] M. Ali, J. Nelson, R. Shea, and M. J. Freedman, "Blockstack: A global naming and storage system secured by blockchains," in *2016 USENIX Annual Technical Conference (USENIX ATC 16)*. Denver, CO: USENIX Association, Jun. 2016, pp. 181–194.

[60] Emercoin. (2014) Emerdns. Accessed: October 30, 2024. [Online]. Available: https://emercoin.com/en/emerdns/

[61] Z. Li, S. Gao, Z. Peng, S. Guo, Y. Yang, and B. Xiao, "B-dns: A secure and efficient dns based on the blockchain technology," *IEEE Transactions on Network Science and Engineering*, vol. 8, no. 2, pp. 1674–1686, 2021.

[62] T. Gao and Q. Dong, "Dns-bc: Fast, reliable and secure domain name system caching system based on a consortium blockchain," *Sensors*, vol. 23, no. 14, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/14/6366

[63] L. Zhu, X. Zhu, L. Sun, S. Hu, S. Chen, and X. Guo, "Optimized cross-chain mechanisms for secure and reliable domain information synchronization in blockchain-driven networks," in *Communications in Computer and Information Science*, vol. 2098 CCIS. Springer Science and Business Media Deutschland GmbH, 2024, pp. 157–166.

[64] U. Divakarla and K. Chandrasekaran, "D-dns: A decentralized domain name system on the blockchain: Implementation and assessment," in *2024 IEEE International Conference on Blockchain and Distributed Systems Security, ICBDS 2024*. Institute of Electrical and Electronics Engineers Inc., 2024.

[65] L. T. Thibault, T. Sarry, and A. S. Hafid, "Blockchain scaling using rollups: A comprehensive survey," *IEEE Access*, vol. 10, pp. 93 039–93 054, 2022.

[66] S. Motepalli, L. Freitas, and B. Livshits, "Sok: Decentralized sequencers

for rollups," *arXiv preprint arXiv:2310.03616*, 2023. [Online]. Available: https://arxiv.org/abs/2310.03616

[67] A. Koegl, Z. Meghji, D. Pellegrino, J. Gorzny, and M. Derka, "Attacks on rollups," in *Proceedings of the 4th International Workshop on Distributed Infrastructure for the Common Good*, ser. DICG '23. New York, NY, USA: Association for Computing Machinery, 2024, p. 25–30. [Online]. Available: https://doi.org/10.1145/3631310.3633493

[68] M. Capretto, M. Ceresa, A. F. Anta, P. Moreno-Sánchez, and C. Sánchez, "A decentralized sequencer and data availability committee for rollups using set consensus," *arXiv preprint arXiv:2503.05451*, 2025. [Online]. Available: https://arxiv.org/abs/2503.05451

[69] B. França, D. Kolegov, I. Konnov, and G. Prusak, "Chonkybft: Consensus protocol of zksync," 2025. [Online]. Available: https://arxiv.org/abs/2503.15380

[70] S. R., "Promise of zero-knowledge proofs (zkps) for blockchain privacy and security: Opportunities, challenges, and future directions," *Security and Privacy*, vol. 8, 09 2024.

[71] X. Sun, F. R. Yu, P. Zhang, Z. Sun, W. Xie, and X. Peng, "A survey on zero-knowledge proof in blockchain," *IEEE Network*, vol. 35, no. 4, pp. 198–205, 2021.

[72] S. Bowe, A. Gabizon, and I. Miers, "Scalable multi-party computation for zk-snark parameters in the random beacon model," *IACR Cryptol. ePrint Arch.*, p. 1050, 2017. [Online]. Available: http://eprint.iacr.org/2017/1050

[73] Hyperledger, "Hyperledger Sawtooth (Archived) - LF Decentralized Trust — lf-hyperledger.atlassian.net," https://lf-hyperledger.atlassian.net/wiki/spaces/sawtooth/overview, 2015, [Accessed 30-10-2024].

[74] Parity Technologies. Substrate: State transitions and storage. (Accessed 15 November 2024). [Online]. Available: https://docs.substrate.io/learn/state-transitions-and-storage/

[75] Protocol Labs. (2025) libp2p. Accessed: 2025-06-21. [Online]. Available: https://pkg.go.dev/github.com/libp2p/go-libp2p

[76] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric," in *Peer-to-Peer Systems*, P. Druschel, F. Kaashoek, and A. Rowstron, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 53–65.

[77] Centrifuge. (2023) Go substrate rpc client (gsrpc). Accessed: November 16, 2024. [Online]. Available: https://github.com/centrifuge/go-substrate-rpc-client

[78] CircleID. (2024) Global domain activity trends seen in q3 2024. Accessed: December 4, 2024. [Online]. Available: https://circleid.com/posts/global-domain-activity-trends-seen-in-q3-2024

[79] Domain Name Stat. (2024) Domain name registration's statistics. Accessed: December 4, 2024. [Online]. Available: https://domainnamestat.com/statistics/overview