UNIVERSITÉ DU LUXEMBOURG

PhD-FSTM-2023-078
The Faculty of Science, Technology and Medicine

DISSERTATION

Defence held on 25/09/2023 in Luxembourg

to obtain the degree of

DOCTEUR DE L'UNIVERSITÉ DU LUXEMBOURG

EN INFORMATIQUE

by

Vytautas Tumas
Born on 9th of July 1993 in Kaunas, Lithuania

MULTI-LAYER SECURITY ANALYSIS OF THE XRP LEDGER

Dissertation defence committee

Dr Radu STATE, dissertation supervisor
Full Professor, Université du Luxembourg

Dr Mats BRORSSON
Research Scientist, Université du Luxembourg

Dr Gilbert FRIDGEN, Chairman
Full Professor, Université du Luxembourg

Dr Mathis BADEN
Telindus

Dr Damien MAGONI, Vice Chairman
Professor, Universite de Bordeux

S. T.

Acknowledgments

During the years of my PhD, I was incredibly fortunate to enjoy the support, encouragement and advice from many incredible people.

Above all, I would like to thank Radu State for his guidance, supervision and all the fantastic opportunities. Though, most importantly the contagious, never-fading *"Let's do it!"* attitude.

I am grateful for all past and present SEDAN members, especially Rob, Beltran, Mary & Sean and Christof. Thank you for the brilliant ideas, lunch debates and great memories from Luxembourg.

My brothers-in-arms, doctors Gergely and Guillermo, for the bemusing amusement, the puntastic discourse and, above else, company. I must mention the wonderful people from Luxembourg climbing community, who never let me down and never left me hanging.

Last but definitely not least, I would like to thank my family for pushing me to persevere and remain steadfast in my goals.

This was quite a ride, and you made it awesome!
Thank you.

Vytautas Vito Tumas
Luxembourg, September 25th, 2023

Abstract

XRP Ledger is one of the oldest blockchains. It was created to address the inefficiencies of Bitcoin. Unlike Bitcoin or Ethereum, XRP Ledger uses a voting-based consensus protocol, which allows for fast and efficient creation of new blocks. Despite its advantages and high market capitalization, it has received little academic interest. Existing literature mainly explores the consensus protocol. Whilst important, many other aspects that affect the performance, safety and robustness are not fully understood. Therefore, in this work, we perform first of its kind multi-layer analysis of the XRP Ledger to gain deeper understanding and reveal opportunities for further improvement.

In the first part of the dissertation, we outline our research goals, discuss blockchains and provide a detailed description of the XRP Ledger.

In the following part, we study the XRP Ledger peer-to-peer network topology. We begin by examining the graph-theoretic properties of the network and how they change over time. We show the existence of critical nodes that provide network connectivity. We use these results to examine the robustness of the network and the consensus protocol against random and targeted attacks. We show that an attacker has to disconnect only 9% of nodes to halt the XRP Ledger. Furthermore, we propose a mitigation strategy that increases robustness to 54%, whilst maintaining critical topological properties.

In the third part, we examine the protocol layer by exploring ways to improve message routing efficiency. To this end, we propose *pemcast* - probabilistic edge multicast routing algorithm. It uses local topology awareness to reduce the amount of redundant traffic generated whilst propagating consensus messages.

In the fourth part, we examine the decentralized exchange running on the XRP Ledger. Concretely, we study whether the pseudo-random transaction execution order provides sufficient protection against frontrunning. We show that frontrunning is not only possible but also profitable. Furthermore, we discover existing actors that perform frontrunning attacks.

Finally, we conclude the dissertation by examining the answers to the research questions we posed in the first chapter.

Contents

List of Figures

List of Tables

Part I

Prelude

1 | Introduction

"Devyni amatai, dešimtas badas."

Nine trades, the tenth one - hunger.

Lithuanian Proverb

On the 31st of October 2008, Satoshi Nakamoto revealed the *Bitcoin: A Peer-to-Peer Electronic Cash System* white paper, in which they describe a decentralised digital currency that does not rely on trusted third parties [29]. What started as a novelty used by technology enthusiasts has quickly grown into a virtual asset worth 1.2 trillion USD at its peak [97]. Although Bitcoin is not the first cryptocurrency to solve the double-spending problem [113, 95, 93], the solution of Bitcoin did not suffer from crippling trade-offs and limitations that prevented others from gaining widespread adoption.

Blockchains or Distributed Ledgers (DL) enable mutually distrusting parties to conduct financial operations and reach an agreement on an ever-growing log of transactions. In other words, they serve as decentralised payment networks. At its core, a distributed ledger is a decentralised database containing an append-only list of records. A record is composed of zero or more transactions. A transaction is a cryptographically signed message communicated to the system that modifies the ledger state. A set of servers in a decentralised distributed peer-to-peer network maintains the records. The servers follow protocol rules to accept new transactions and rely on a consensus protocol to determine which transactions to include in a record.

There exist multiple blockchain taxonomies, each creating a different blockchain classification. We opt to use two high-level groups based on the openness of the blockchain. These are *permissioned* and *permissionless* blockchains.

Permissioned, also known as a private blockchain, is access restricted. Only a specific

group of pre-authorised participants can access it. Due to strict access rules, the participants trust each other, and only a limited number of nodes are responsible for validating transactions and maintaining the ledger integrity.

In contrast, a permissionless blockchain, sometimes called a public blockchain, is a type of blockchain where anyone can participate as a node, validate transactions, and add blocks to the chain without prior authorisation or permission. These blockchains operate on a decentralised network, where all nodes have equal status and can participate in the consensus mechanism to validate transactions and maintain the ledger integrity.

The permissionless blockchain consensus protocols fall between resource-based and communication-based protocols that utilise classical distributed consensus techniques. For example, in the *Proof-of-Work* consensus protocol used in Bitcoin [29], participants must solve computationally intense cryptographic problems. Specifically, the block creator must search for a numeric value such that its hash starts with a certain number of zeroes. Once a node finds a satisfying number, it transmits it to the rest of the network. The *finder* of the block receives a reward in the form of newly minted Bitcoin coins for their effort, akin to gold mining, hence the term *Bitcoin mining*.

As the Bitcoin network continues growing, the Proof-of-Work consensus protocol encountered multiple performance bottlenecks and sustainability challenges. The blockchain community has raised the following concerns about Proof-of-Work: 1) unsustainable energy consumption, 2) low transaction capacity and poor scalability, and 3) diminishing mining rewards.

By design, the mining process requires a colossal amount of hash calculations. On the 15th of November 2022, Bitcoin performed an estimated 243 Quintillion hash operations per second [96]. At the same time, the blockchain produced around 3 transactions per second. The ratio of hash calculations to processed transactions is, at best, 81 Quintillion to 1.

In addition, Bitcoin processes between 3 and 7 transactions per second [39], whereas the Visa payment network handled around 3,900 transactions per second during the 4th quarter of 2020 [137].

Furthermore, on the Visa platform, transactions are recorded sequentially. Therefore, the time interval between submission and the transaction recording is kept to seconds, while on Bitcoin, the same latency can be between 10 minutes and several hours [39].

In response to these limitations, the blockchain community designed the Proof-of-Stake [130] as an energy-efficient alternative to Proof-of-Work. The stake refers to the coins owned by a participant that they can invest in the consensus process. Instead of brute-force computation, the size of the stake is proportional to the chance to propose a new block.

The *Proof-of* techniques utilise resource-based algorithms to reach a consensus. They are well-suited for public, dynamic blockchains where anyone can participate in the consensus process. At the other end of the spectrum are communication-based protocols, where participating entities achieve consensus by some voting mechanisms over several rounds of communication. One such public, distributed ledger incorporates these characteristics: XRP Ledger [1]

The XRP Ledger is a public, decentralised cryptocurrency created in 2011 by David Schwartz, Jed McCaleb, and Arthur Britto. The developers set out to build a new distributed ledger that improves upon the fundamental limitations of Bitcoin, such as high resource consumption and slow transaction speeds. The XRP Ledger can handle up to 1,500 transactions per second, with an approx 3-second confirmation time [33]. As of 2023, It is the 6th largest cryptocurrency by market capitalization [116].

The XRP Ledger differs from Bitcoin and Ethereum in several ways, primarily the fast and efficient XRP Ledger Consensus Protocol [124]. Unlike Bitcoin, where nodes compete against each other to produce a new block on the XRP Ledger, validating servers, called *validators*, collaborate to establish a new ledger version.

Validators share candidate transaction information by exchanging signed messages to achieve consensus. However, other nodes consider messages sent by specific validators they deem to be trustworthy. In other words, each participant of the XRP Ledger chooses which validators to trust when making decisions about the next ledger version. Each node maintains a list of trusted validators called *Unique Node List*. The node accepts a set of transactions only when a super majority of UNL members, known as a quorum, propose the same set of transactions.

Despite the widespread adoption of the XRP Ledger, the blockchain has not received much attention from the academic community. The majority of works focus on the XRP Ledger Consensus Protocol and its various security aspects, such as analysing the safety

[1] Originally named "Ripple", the XRP Ledger term was introduced to differentiate from the "Ripple Inc" company.‘

properties of the protocol [47], studying the security of the unique node list mechanism[65], and evaluating the liveness of the protocol [73]. However, there is a lack of research on the general properties and characteristics of the XRP Ledger.

1.1 Research Questions

Unlike other academic works that focus on a single aspect of the XRP Ledger, we take a multi-layer view of the system.

As a security measure, most blockchains conceal their network topology. For example, Bitcoin nodes are public. However, nodes hide their connections [123]. This security-by-obscurity approach attempts to hinder partitioning and eclipse attacks. The former attacks attempt to isolate select nodes in the network by taking control of network connections [63]. The latter attacks flood the network with enough malicious nodes to control at least 51% of the network nodes. After that, the attacker may outvote honest nodes and refuse to transmit or receive blocks. Furthermore, the attacker may modify the transaction order, reverse transactions to enable double-spending or prevent transaction confirmation [64].

Since the inception of Bitcoin, researchers have created multiple techniques to bypass the security by obscurity measure by developing novel network inference techniques [37, 81, 49, 41]. While Bitcoin developers promptly resolve these issues, it is hard to prevent attackers from using undisclosed network topology inference methods. On the other hand, obscuring the network topology impedes network analysis [37] [48]. Ultimately, these limitations hinder the efficiency and security improvement of the network.

The XRP Ledger community took a radically different approach by providing a public peer crawler API [109]. The public API reveals the peers of the called node, allowing anyone to infer the topology of the XRP Ledger. We can study the accurate XRP Ledger topology to determine structural properties such as size, connectivity, degree distribution and others that affect network robustness and message propagation delay.

Robustness, also called resilience, is the ability of the network to continue providing its services even when some nodes are absent [38]. Node outages due to updates, crash faults and network failures are common occurrences in blockchains [57]. Seminal research by Cohen et al. [10] shows that depending on network topology, a significant portion of the network nodes have to fail to disrupt the network. Furthermore, in a related work, Cohen

et al. proved that an attack against the most connected nodes could disrupt the network by disconnecting only a fraction of these nodes [12]. The core service of XRP Ledger is to append valid user transactions to the ever-growing blockchain history. XRP Ledger achieves this through a voting-based consensus protocol, in which trusted *validator* nodes work to determine which transactions and in what order should be accepted. Therefore, we submit that, to halt the XRP Ledger, an attacker does not need to fully fragment the network. It is sufficient to disconnect the validator nodes.

Message propagation delay is the time required for a message to reach each node in the network after submission. Messages propagate quickly in blockchains where the average shortest path is small. Fast message propagation causes fewer stale blocks and forks [40]. Given the high throughput and small transaction confirmation time on the XRP Ledger, we consider that message propagation is a crucial property of the XRP Ledger.

The XRP Ledger topology is still unknown. Therefore, we are the first to answer the following research question:

Research Question 1 - RQ 1

What are the topological properties and how do they affect the robustness of the XRP Ledger?

In the past, the XRP Ledger used flooding to propagate consensus protocol messages. The developers of XRP Ledger determined that consensus protocol messages accounted for 72% of traffic in the network [58]. The developers proposed a novel *squelching* algorithm to reduce the amount of redundant traffic and server load. This algorithm allows servers to select a subset of peers that act as the source of consensus messages from a specific validator. It suppresses, or squelches, messages from the rest of its peers. This new algorithm improves efficiency, making XRP Ledger more resilient and less prone to network congestion [133].

At its essence, squelching represents a gossip algorithm that leverages the message delivery speed from a specific validator as an optimisation metric. Gossip protocols minimise redundant traffic during the propagation of messages within a network. The term "gossip protocol" was introduced by Alan Demers [4] when studying methods for disseminating in-

formation in unreliable networks. Gossip protocols garnered extensive research attention at the turn of the century and have experienced a resurgence in interest in recent years, primarily due to advancements in blockchain technology [66, 77]. These protocols are highly suitable for decentralised systems like blockchains, as they enable efficient information dissemination without relying on a central authority.

In gossip protocol, a node selects several peers to send its message. The number of peers is called fanout. The fanout can be static or adjusted dynamically based on various conditions, such as network congestion or the number of messages in the network. A higher fanout value can lead to faster dissemination of information. However, it also increases the amount of network traffic and the risk of message duplication. A lower fanout value can reduce network traffic, but it can also slow down the dissemination of information. Therefore, selecting an optimal fanout value for a gossip protocol is crucial when designing and deploying a gossip-based system. Leitao *et al.* [31] showed a trade-off between the fanout and protocol reliability and an inverse correlation between fanout and messaging efficiency. Our second research question is as follows:

Research Question 2 - RQ 2

Can the routing efficiency of the XRP Ledger be further improved?

Historically, market manipulation techniques exploiting insider knowledge, i.e., non-public information, have been prevalent in traditional financial systems. These techniques have also extended to digital asset markets. Frontrunning is one such technique that affects blockchains. Attackers create new transactions based on public, pending, but uncommitted transactions for a considerable profit. These attacks are frequent on the Ethereum blockchain [88]. For the attacks to be effective, the attacker needs a way to manipulate the transaction execution order. Ethereum uses the fee to determine execution order, i.e. transactions with a higher cost execute first. Therefore, to frontrun, an attacker places a transaction with a greater fee than the victim. One proposed mitigation strategy is to process the transactions in a random order [110]. This method makes it harder for attackers to exploit their insider knowledge.

XRP Ledger users were victims of frontrunning in the past [36]. Similarly to other

blockchains, the pending transactions on XRP Ledger are public. Adversaries could generate low transaction IDs to ensure their adversarial orders are applied first. As a response, the XRP Ledger developers introduced a new transaction ordering strategy to make the transaction order difficult to predict [101]:

> *"The order transactions execute within a ledger is designed to be unpredictable to discourage frontrunning."*

The new transaction ordering strategy, similar to suggestions in previous works [110], creates a pseudo-random shuffle of the to-be-executed transactions. However, the effectiveness of such a strategy is not yet clear. Therefore, our final research question is as follows:

> **Research Question 3 - RQ 3**
>
> *Does pseudo-random transaction execution order provide sufficient protection against frontrunning attacks?*

1.2 Contributions

In answering these questions we have made the following contributions:

1. We examine the graph-theoretic properties of the XRP Ledger peer-to-peer network and its temporal characteristics. We collected 1,290 unique network snapshots. We uncover a small group of nodes that act as a networking backbone. In addition, we observe a high network churn, with a third of the nodes changing every five days. Our findings have implications for the resilience and safety of the XRP Ledger.

2. We propose a new robustness metric that captures quorum-based consensus protocol resilience to network disruptions. We show that the XRP Ledger Consensus Protocol is vulnerable to targeted network attacks. An attacker has to disconnect only 9% of the highest-degree nodes to halt the blockchain. We propose a mitigation strategy which increases the FBA resilience to approximately 55%.

3. We present *pemcast*, an application layer algorithm for efficient one-to-many message routing. The algorithm leverages limited topology awareness and application layer multicasting to deliver messages in the network. The evaluation shows that compared to flooding and gossiping algorithms, *pemcast* can maintain similar reliability whilst generating significantly less redundant traffic.

4. We examine whether pseudo-random transaction order provides sufficient protection against frontrunning. Our results show that the mechanism embedded in the XRP Ledger protocol is insufficient to prevent these attacks. We showcase two strategies to perform frontrunning attacks. The first strategy uses randomly generated accounts. The second strategy uses specially created ones to improve the probability of a successful attack. Based on our XRP Ledger historical data analysis, we estimate that attackers could generate up to approx. 1.4M USD profit over two months, provided they succeeded to frontrun every opportunity.

1.3 Overview

The remainder of the thesis consists of four parts. Each part examines one of the three research questions, and the final part concludes our work.

1.3.1 Part II: Infrastructure Layer

The second part consists of two chapters. It focuses on a graph-theoretic and robustness analysis of the XRP Ledger Network.

Chapter 3 focuses on the topology analysis of the XRP Ledger. It answers parts of *RQ1* by revealing the XRP Ledger network topology structural properties and how they change over time. This chapter is based on the paper:

- Vytautas Tumas, Sean Rivera, Damien Magoni, and Radu State. 2023. Topology Analysis of the XRP Ledger. In the 38th ACM/SIGAPP Symposium on Applied Computing, March 27 - March 31, 2023, Tallinn, Estonia.

Chapter 4 focuses on the robustness analysis of the XRP Ledger. It presents a new robustness metric for Federated Byzantine Agreement Protocols. This chapter is based on

the paper:

- Vytautas Tumas, Sean Rivera, Damien Magoni, and Radu State. 2023. Federated Byzantine Agreement Protocol Robustness to Targeted Attacks. In the 28th IEEE Symposium on Computers and Communications, July 9 - July 12, 2023, Tunis, Tunisia.

1.3.2 Part III: Network Layer

The third part examines the network layer of the XRP Ledger. Concretely, it focuses on improving message propagation on the XRP Ledger.

Chapter 5 analyses the network layer of the XRP Ledger. In this chapter, we examine the existing routing algorithm and propose *pemcast*, a probabilistic edge multicast routing algorithm to reduce the amount of redundant traffic generated in the network. This chapter is based on the paper:

- Vytautas Tumas, Sean Rivera, Damien Magoni, Radu State. 2022. Probabilistic Edge Multicast Routing for the XRP Network. In IEEE Global Communications Conference, December 4 - December 8, 2022, Rio De Janeiro, Brazil.

1.3.3 Part IV: Protocol Layer

The fourth part examines the decentralized exchange of the XRP Ledger.

Chapter 6 examines the effectiveness of the pseudo-random transaction shuffling as a frontrunning prevention strategy. As well as analyses existing occurrences of frontrunning on the XRP Ledger. This chapter is based on the paper:

- Vytautas Tumas, Beltran Borja Fiz Pontiveros, Christof Ferreira Torres, Radu State. 2023. A Ripple for Change: Analysis of Frontrunning in the XRP Ledger. In IEEE International Conference on Blockchain and Cryptocurrency, May 1 - May 5, 2023, Dubai, United Arab Emirates.

1.3.4 Part V: Conclusion

In this final part, we examine our answers to the research questions and summarise our work.

2 | Background

"Kas skaito rašo duonos neprašo."

He who reads and writes, does not beg for bread.

Lithuanian Proverb

2.1 Blockchain at a glance

Overview Blockchain is a decentralized, distributed ledger that records secure and tamper-resistant transactions maintained by a network of nodes. They validate and record transactions and use a consensus mechanism to ensure the ledger's integrity. Once in the blockchain, a transaction is immutable, creating a permanent and transparent record of all activities on the network. As a result, blockchain technology is well-suited for multiple use cases, including digital currencies, supply chain management, and decentralized applications. The decentralized and secure nature of the blockchain makes it attractive for many industries as it eliminates the need for intermediaries and provides a trusted and tamper-proof record of transactions.

Blockchains such as Bitcoin and Ethereum guarantee that the state is correct as long as more than half of the voting power, expressed as compute power in Proof-of-Work, or wealth distribution in Proof-of-Stake, is controlled by honest participants. These blockchains are *permissionless*, as arbitrary participants can join and leave the system anytime. Furthermore, any of these actors can participate in the consensus process. In contrast, a *permissioned* model restricts the participant set to only known, authorized entities.

2.1.1 Key Properties

Three key attributes, *decentralization, immutability*, and *transparency*, provide the foundation for secure, efficient, and trustworthy transactions and record-keeping on the blockchain network.

Decentralization One of the core features of a blockchain is that it operates on a decentralized network of nodes rather than relying on a single central server. All transactions are verified and validated by a consensus of participating nodes in the network. Although the blockchain infrastructure is decentralized, a single entity may control all the nodes. Therefore, a public blockchain must also exhibit decentralization of ownership. That is why a wider adoption of the blockchain increases security.

Immutability Immutability refers to the ability of the blockchain to maintain a permanent and unalterable record of all transactions that have taken place on the network. The blockchain guarantees this through cryptographic hashes by linking blocks to each other. The only one without a "parent" is the first block, called the *genesis block*. Once a block is in the chain, it is immutable, providing a permanent and auditable record of all transactions.

Transparency Transparency refers to the network participants' ability to access and view the information contained within the blockchain. Transparency creates a shared and verifiable source of truth, not controlled by any single party and allows participants to validate the authenticity and accuracy of the data on the network.

In permissioned blockchain, the level of transparency can vary depending on the specific design and implementation. The data may be accessible only to select or all participants but not the public. Or a permissioned blockchain may provide a limited level of transparency to the public but only write access to select individuals or organizations.

2.1.2 Blockchain Layers

Similarly to the OSI model, a blockchain may be split into several abstraction layers: *infrastructure, network, protocol, interface, and application layers* [94, 128]. Each layer depends on the functionality of the layers below it.

1. The **Infrastructure Layer** is the lowest level of the blockchain architecture and refers to the physical and technical components that support the blockchain network. It includes the hardware, network connectivity, and data storage.

2. The **Network Layer** refers to the software and protocols that support the communication and data transfer between nodes in the blockchain network. This layer is responsible for distributing transactions to other nodes and maintaining the integrity of the blockchain.

3. The **Protocol Layer** refers to the rules and standards that govern how transactions are processed and validated on the blockchain. This layer defines the consensus mechanism used to validate transactions, the method for adding new blocks to the chain, and the algorithms for managing and securing the blockchain.

4. The **Interface Layer** is responsible for providing a user-friendly interface for accessing and interacting with the blockchain. It can include web or mobile applications, APIs, or other tools that allow users to interact with the blockchain and perform transactions.

5. The **Application Layer** refers to the various applications and services built on top of the blockchain. They can include decentralized exchanges, supply chain management systems, voting systems, and many other applications that leverage the security and transparency of the blockchain.

These five layers form the complete blockchain architecture and define how data is processed, stored, and secured on the blockchain network.

2.2 Trust the Consensus

Distributed Ledger Technology (DLT) allows distrusting parties to exchange virtual assets without a trusted central authority. In other words, DLT shifts trust away from a central authority and places it on the collection of components, such as cryptography, consensus, communication and other protocols, that comprise a distributed ledger. The fact that benevolent actors follow an identical set of rules to reach an agreement about the state of the distributed ledger makes the whole system appear trustworthy. However, this shift of trust

presupposes a careful examination of trust assumptions made by the consensus mechanism. The implicit or explicit trust assumptions affect the correctness and stability of the blockchain.

The consensus protocols can tolerate two types of failures: *crash faults* or *Byzantine faults*. Crash faults are straightforward. The participant does not perform any operations and does not respond to messages sent.

In contrast, the *Byzantine* failures [2] are more complex. The failing participant behaves arbitrarily, deviating from the specified protocol and taking any action. It may follow the protocol, respond correctly or incorrectly or not respond at all. In the Byzantine failure model, participants' behaviour is uncertain. *Byzantine Fault Tolerance* (BFT) is the characteristic of a system that tolerates Byzantine failures. A BFT system tolerates up to one-third of its participants being malicious [2].

The quorum-based consensus relies on a set of participants reaching an agreement about the system state. The consensus participants exchange messages to reach an agreement. In 1985, Fisher, Lynch and Paterson (FLP) [3] published a seminal work which proves that it is impossible to reach a consensus in a fully asynchronous distributed system in which at least one participant can fail. Therefore, most consensus mechanisms assume an eventually synchronous communication model.

In traditional quorum-based systems, trust is *symmetric*. Participants trust other participants in the same way. In other words, participants trust each participant the same "amount". However, this does not reflect the reality of trust, as some participants may be more trustworthy than others. An alternative *asymmetric* trust model captures a more subjective notion of trust [26]. Each participant selects whom to trust. However, the trust is not bidirectional. I.e. you trusting me doesn't imply I trust you. In addition, participants share their identities and authenticate communications to prevent Sybil attacks.

2.3 XRP Ledger

The XRP Ledger is a public, distributed, open-access blockchain created in 2011. Interconnected servers running the *rippled* software process transactions, manage the ledger and apply the XRP Ledger Consensus Protocol to determine which transactions to include in a new block.

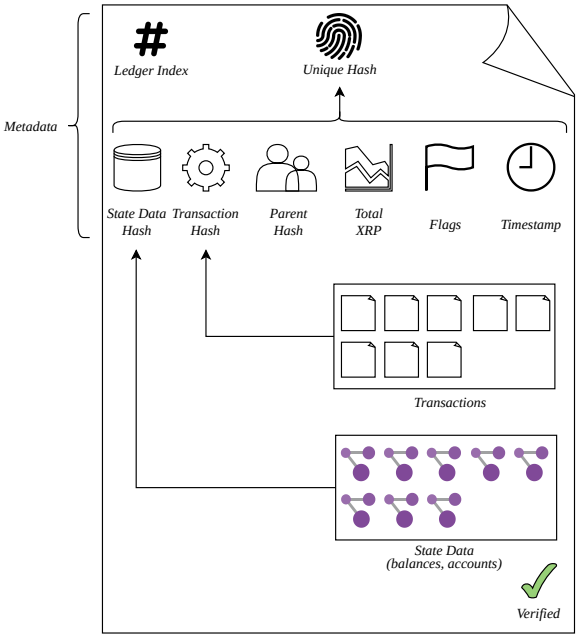


Figure 2.1: Ledger Anatomy.

2.3.1 Trust in XRP Ledger

The XRP Ledger is the first widely adopted blockchain to use flexible trust. The blockchain uses a quorum-based consensus protocol driven by specific servers called *validators*. Unlike traditional Byzantine systems, anyone can participate in the consensus process. Equally, each participant selects which *validators* to trust. The key idea is that a validator does not need to be trusted or trust everyone. Each participant maintains a *Unique Node List* (UNL) of trusted validators, which reflects subjective trust. During the consensus process, validators only process messages from members of their UNL.

Conceptually, XRP Ledger is a hybrid system with aspects of BFT-based protocols and permissionless blockchains. In principle, anyone can participate in the consensus protocol. Nevertheless, other participants are free to select whom to trust. This limits the influence of other validators. By expressing their trust assumptions, validators determine with which other participants they will participate in consensus, thereby entailing a certain degree of permissioning.

2.3.2 Ledgers

The shared global state is a series of individual ledgers. A ledger is a result of applying the transactions accepted by the consensus process to the previous ledger (See Figure 2.1).

A unique hash and an index identify each ledger. The ledger index is always one greater than the previous ledger. A single ledger version consists of A *state tree* which contains the settings, balances and a set of objects. A *tree* of transactions applied to the previous ledger to make the new ledger version. A *header*, which contains the following fields:

- ***Ledger Index*** a 32-bit unsigned integer which uniquely identifies a ledger. The ledger index is always one greater than the index of the previous ledger.
- ***Unique Hash*** a *SHA-512Half* of the ledger. It serves as a unique identifier of the ledger and its contents.
- ***State Data Hash*** The *SHA-512Half* of the state tree containing current balances and objects in the ledger.
- ***Transaction Hash*** The *SHA-512Half* of the transactions included in this ledger.
- ***Parent Hash*** the *Unique Hash* value of the direct predecessor of this ledger.
- ***Total XRP*** the total number of XRP drops owned by accounts in the ledger, omitting the coins destroyed by transaction fees.
- ***Closed*** a boolean value indicating whether the ledger accepts new transactions. However, unless validated, a closed ledger might be replaced by a different one.
- ***Timestamp*** An approximate time the ledger was closed as a number of seconds.

At any given time, a *rippled* instance maintains an in-progress *open ledger*, several *closed ledgers* yet to be accepted by the consensus protocol and any number of *fully validated ledgers*. This series of states is an evolutionary line that starts with the view of a single validator and ends with a permanent, globally accepted state.

The Open Ledger Each validator maintains a single *open ledger*. It represents the validator's current working version of the new state. Validators apply incoming transactions to the

open ledger in their arrival order. Eventually, a validator closes the ledger and begins the consensus protocol. It includes subsequent transactions in a later version.

The validator does not "close" the open ledger. Instead, it uses the transactions from the *open ledger* to create a new *closed ledger*. Validators determine which transaction to add to it through the consensus process.

The Closed Ledger A validator starts with a set of transactions agreed upon by consensus and a parent ledger to create a *closed ledger*. It applies them in a deterministic, pseudo-random order to the parent ledger. The order is deterministic as each validator executes the transactions in the same order, but the order itself will be pseudo-random. We detail the transaction ordering in Section 2.3.6.

Two attributes identify the closed ledger. The *ledger_index* is a monotonically increasing number. The *ledger_hash* uniquely identifies the ledger's content.

The closed ledger reflects the personal view of the validator about which transactions to include permanently. Due to Byzantine failures, different validators may calculate differing *closed ledgers*. As a result, there may be multiple closed ledgers with the same *ledger_index*, but different *ledger_hash* values competing to be validated. Therefore, unless fully validated, a closed ledger may be replaced by a differing *closed ledger* with a different set of transactions.

The Fully Validated Ledger The closed ledgers are candidates for a fully validated ledger. The fully validated ledger confirms the previous state of the ledger. It is final and immutable and represents the latest state of the ledger validators have agreed.

2.3.3 Stages of Consensus

The overall consensus process consists of two phases, *proposal* and *validation*. Throughout the consensus, validators broadcast proposal and validation messages. However, the receiving validators only regard messages from validators in their UNL.

Proposal A proposal is a signed message by the validator, which contains a set of transactions the validator believes should be included in the next ledger. Throughout the proposal

phase, validators will exchange multiple proposals. A divergence may occur between two validators when validators propose differing transactions. Such a concept is called a *dispute*.

Validation The aim of validation is for trusted validators to agree on which closed ledger should be declared fully validated.

2.3.4 Transactions

A transaction is the only way to update the state of the XRP Ledger. They are used to make payments, create offers on the XRPL's DEX, or manage account settings.

Analogues to the ledger, a unique hash and sequence number identify each transaction. The sequence number ensures that the transactions from a given sender execute only once and in the correct order. The sequence number of an account must match the sender's sequence number during the transaction execution. We detail accounts and their sequence numbers in Section 2.3.7

Finality Each *rippled* instance processes transactions independently and verifies that the rest of the trustees agree with the outcome. A server provisionally applies well-formed transactions it receives to the current *open ledger* and returns the tentative result. The transaction set and order is final only after the consensus. Therefore, a transaction may succeed during submission but fail after validation.

Queuing In addition to ledgers, a server maintains a transaction *queue*. When a validator removes a transaction from its proposed set, it adds it to its queue. Later, a server adds its queued transactions to the next open ledger.

Each transaction must *destroy* a small amount of XRP to safeguard against denial-of-service (DoS) attacks. Each transaction specifies a fee, the XRP amount to destroy. It increases with the network load, making DoS attacks expensive. Moreover, the transaction cost is deducted even when a transaction fails. Each server maintains a cost threshold based on its local load. A server discards transactions whose fee is below the load-based threshold.

In addition to the load cost, a transaction must meet the open ledger cost. Servers pick an open ledger size limit based on the number of transactions in the latest validated ledger.

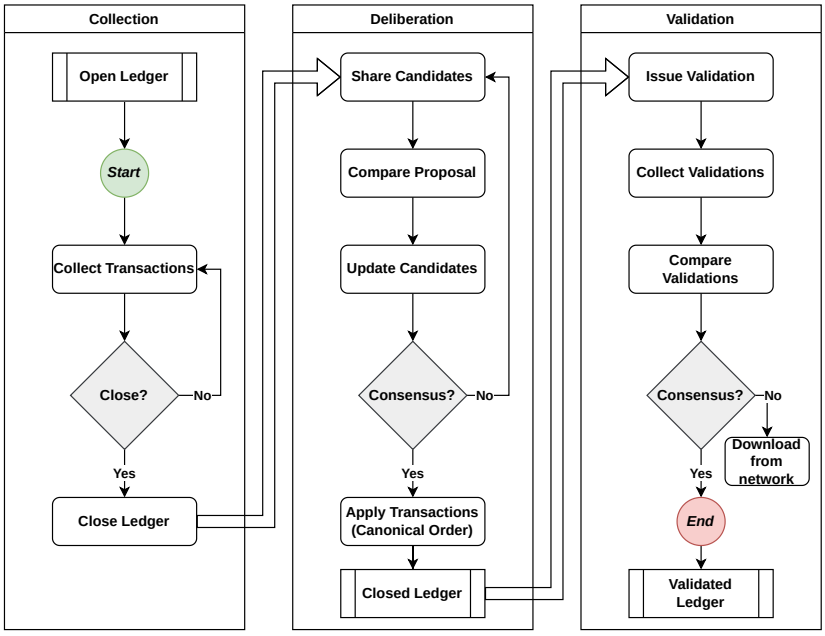


Figure 2.2: XRP Ledger Consensus Protocol Phases.

The open ledger cost is the minimum transaction cost, while the open ledger size is below the threshold. Afterwards, cost increases exponentially as the transaction volume grows. The server increases the soft limit for the next open ledger. However, the server decreases the size limit if the consensus process takes over 5 seconds. Instead of discarding, servers queue the transactions that don't meet the open ledger cost.

Within the queue, the transactions follow an order based on the transaction fee (high to low). Therefore, the transaction with the highest fee, relative to the minimum transaction cost, is added first to the next open ledger.

2.3.5 XRP Ledger Consensus Protocol

At the heart of the XRP Ledger lies the XRP Ledger Consensus Protocol, the mechanism by which *validators* agree on the state of the distributed ledger. The ultimate goal of the protocol is to allow the servers to decide which transactions to include in the next fully validated ledger.

The overall consensus protocol is a progression through three stages: *collection*, *deliberation* and *validation* (See Figure 2.2). These stages are common among most blockchain consensus algorithms. However, their details vary. On the XRP Ledger, the phases are as follows:

- **Collection**: The collection phase begins with an *open ledger* derived from the previously closed ledger. Validators accept well-formed transactions and apply them to the *open* ledger. Afterwards, they forward these transactions to their peers. Periodically validators close the ledger and progress to the *deliberation* stage.
- **Deliberation** During deliberation, validators attempt to agree on a set of transactions to include in the next ledger version by exchanging *proposal* messages. Initially, validators propose all transactions in the open ledger. However, validators may add or remove transactions from their proposal during disputes. After each update, validators check the percentage of trusted validators with the same transaction set. If at least 80% of trusted validators have agreed on a transaction set, they create a new closed ledger and advance to the final *validation* stage.
- **Validation** In the final stage, validators vote on a *closed ledger* calculated by the majority. Due to Byzantine failures, validators may create differing closed ledgers, i.e. ledgers with the same sequence number but a different hash. Therefore, a validator checks how many trustees computed the same *closed ledger* hash. If the validator's version has not reached a supermajority vote, the validator acquires the ledger version agreed upon by the majority from the network.

2.3.6 Canonical Order

The transactions in the *closed* and *fully validated* ledgers are applied in a *canonical order*[99]. In essence, it is a deterministic, pseudo-randomly shuffled list. It is deterministic because all servers calculate the same transaction order for each ledger. However, the order itself is pseudo-random. Validators calculate the canonical order as follows.

Initially, a validator stores the accepted transactions in a Merkle Tree. A Merkle Tree is a tree data structure in which each node has a label. The label of the leaf nodes is the cryptographic hash of the leaf's data block. The label of each inner node is a cryptographic hash of its children's labels. XRPL uses *SHA512-Half* to create Merkle Tree hashes. The hash of the root of the Merkle Tree acts as a source of randomness (called *salt* in the code) for the transaction shuffling. Kaminsky et al. [59] demonstrated that SHA512-Half produces random hashes. Therefore, the root of the Merkle Tree is a good source of randomness. For each transaction, an *order key* is calculated by XORing the account ID of each transaction

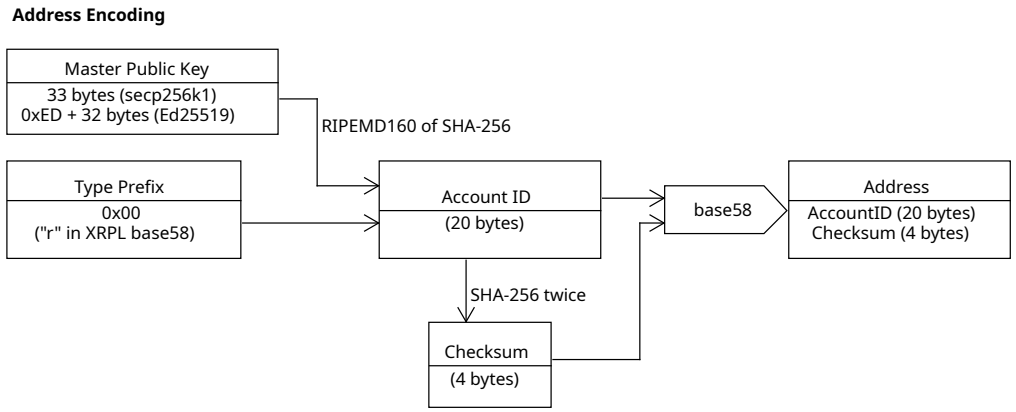


Figure 2.3: Illustrative example of address encoding [122].

with the random salt. The transactions are ordered by doing a pairwise comparison using the following rules:

Rule 1: If order keys are not equal, return ascending order. Otherwise, follow *Rule 2*.

Rule 2: If transaction sequence numbers are not equal, order them in ascending order. Otherwise, follow *Rule 3*.

Rule 3: Order by transaction hash in descending order.

This ordering results in a pseudo-random shuffle of the transactions, where transactions from the same account occur in order of their sequence number. The shuffled transactions are then processed one at a time according to its instructions. Failed transactions still appear in the ledger but with a *tec*-class result code. Some failures are retriable. These transactions are placed at the end of the canonical order and retried after all other transactions in the ledger are executed.

2.3.7 Accounts

An account on the XRP Ledger corresponds to a holder of XRP (i.e., XRPL's native cryptocurrency) and a sender of transactions. An account has two key elements:

- *Address* a unique 160-bit identifier derived from the account's public key. The address is usually presented in a *Base58* encoded string, called the *classic address*. It is

between 25 and 35 characters long and always starts with the character "r". It also includes a 4-byte checksum.

- *Sequence number* is a 32-bit unsigned integer that ensures that the account's transactions execute only once and in the correct order. When an account's transaction is validated and included in a ledger, the account's sequence number is increased by 1. The initial account's sequence number is the ledger index in which the account was funded.

The steps for generating an account address are as follows (See Figure 2.3):

1. Create a public, private key pair using one of *Ed25519* or *secp256k1* algorithms.
2. Create the account ID by hashing the public key with the *SHA-256* and the *RIPEMD160* algorithms and attach *0x00* hexadecimal prefix to the result.
3. Compute the checksum by hashing the account ID with *SHA-256* twice and taking the four most significant bits of the result.
4. Finally, append the checksum to the account ID and encode the result in Base58.

An account is created through a *funding* process. The user has someone with an existing account on the XRP Ledger send XRP to the generated address. The initial amount must satisfy the minimum reserve requirement, which at the time of writing is 10XRP. This XRP is locked up and is released to the funding account once the target account is deleted.

2.3.8 Topology

We discuss the process by which the XRP Ledger overlay network forms hereafter.

2.3.9 XRP Ledger Peer-to-Peer Protocol

The XRP Ledger is a decentralized peer-to-peer overlay network composed of nodes running the *rippled* software. The interconnected rippled servers form the network where each node maintains multiple outgoing connections and optional incoming connections to other nodes. These connections are established over public and private networks, creating a directed graph where the direction indicates which node initiated the connection.

A node requires a set of services to be able to participate in the XRP Ledger.

1. A node requires initial entry into the overlay network.
2. Once it has established initial connections, the node needs to find additional servers, to establish more outgoing connections until it reaches a desired limit.
3. A node needs a mechanism to advertise that it has open slots to accept incoming connections. When a node cannot accept more incoming connections, it needs to provide the requester with alternate node IP addresses to connect.

The *PeerFinder* module of the *rippled* server provides these services.

Initial Entry By default, a server gains initial entry into the network by connecting to a set of hub nodes whose domain names are hardcoded into the rippled implementation. Additionally, server owners may configure their servers to connect to alternative servers by providing a list of IP addresses in the *fixed_peers* stanza of the configuration file.

Establishing Connections Once a server gains initial entry into the network, it requests its peers for additional IP addresses of servers with available incoming connection slots. The server connects to these new peers and repeats the process until it reaches the desired number of outgoing connections. This process allows the server to remain reliably connected to the network, even if it loses some connections. Whenever a server restarts, it reconnects to any of its previous peers.

Accepting Connections A node owner may configure it to accept incoming connections from other nodes, granting them access to the network. The node advertises its available slots to its peers. They use a probabilistic broadcasting algorithm to disseminate the information across the network. As a result, most servers in the network learn of available slots. Any connecting server can use this information to connect to the advertiser. Once the node has reached its configured incoming connection limit, it directs the new node to other servers with available incoming slots.

2.3.10 Latest Development

XRP Ledger community and Ripple are continuously developing new features bringing innovation and long-term vision for the XRP Ledger. In the remainder of this section, we outline

some of the latest developments of the XRP Ledger.

Payment Channels XRP Ledger Payment Channels enable asynchronous micro payments settleable at a future date. Payment Channels are similar to the Interledger Protocol [127]. The process involves two parties, a payer and a payee. The payer sets aside XRP temporarily while creating Claims against the channel. The recipient can verify these claims without a ledger transaction and redeem them later. The participants can settle the payments sent at a later date. The rate of the micro-payments is limited only by the participant's ability to create and verify digital signatures.

Payment Channels on the XRP Ledger are highly versatile for transactions involving digital items that can be transmitted near-instantly, such as music, software, or other types of digital content. Content providers can monetize their products, for example, based on seconds streamed.

Another application of Payment Channels is for situations when the exact quantity of goods or services desired is unknown in advance, such as in bulk purchases. In such cases, Payment Channels facilitate the exchange of goods or services without multiple transactions.

Payment Channels can also be used when a high volume of transactions is expected, such as in online marketplaces or gaming platforms. They enable faster, cheaper, and more efficient transactions, allowing for a seamless user experience. Furthermore, Payment Channels can help reduce congestion on the XRP Ledger by enabling off-ledger transactions, which do not require consensus from the network.

Non-Fungible Tokens XRP Ledger NFTs, or non-fungible tokens, are unique digital assets representing ownership of a particular item or artwork on the XRP Ledger. These tokens are indivisible and cannot be used for payments. The XRP Ledger allows users to mint, hold, buy, sell, and burn these tokens. NFToken objects have various settings defined at the time of minting, including whether the issuer can burn the token, whether the holder can transfer it to others, and whether the holder can sell the token for fungible or non-fungible tokens.

Federated Sidechains A sidechain is another blockchain-based on XRP Ledger technology. It is an independent ledger with separate blockchain rules, transaction types and

consensus algorithms. Federation facilitates the migration of tokens or XRP between the sidechain and the XRP ledger mainchain.

Special *federator* servers enable federation. A federator listens for triggering transactions on both the mainchain and the sidechain. Each federator has a unique signing key that it uses to sign transactions. A quorum of federators must sign a transaction before submitting it. Federators create and sign valid transactions, collect signatures from other federators and submit transactions between the mainchain and the sidechain.

Developers can launch innovative features and applications based on the foundation of XRP Ledger technology using federated sidechains. They can customize the XRP Ledger protocol for a specific use case or project and run it as its blockchain.

The latest application of sidechains aims to bring smart contracts and Web 3.0 applications to the XRP Ledger community powered by an engine compatible with the Ethereum Virtual Machine (EVM). Similarly to Ethereum, the EVM sidechain uses a *Proof-of-Stake* consensus protocol.

Automated Market Makers Automated Market Makers (AMMs) provide liquidity in decentralized exchanges. They hold a pool of two assets and allow users to swap between them at an exchange rate set by a formula. For any given asset pair, there can be up to one AMM in the ledger, and anyone can create the AMM for an asset pair if it doesn't exist yet or deposit it to an existing AMM. Those who deposit assets into an AMM are called liquidity providers (LPs) and receive "LP Tokens" from the AMM. Automated Market Maker (AMM) functionality is part of the proposed XLS-30d extension to the XRP Ledger protocol [106].

LP Tokens enable liquidity providers to redeem their LP Tokens for a share of the assets in the AMM's pool, including fees collected, vote to change the AMM's fee settings (the votes are weighted based on how many LP Tokens the voters hold), and bid some of their LP Tokens to receive a temporary discount on the AMM's trading fees. However, when the relative price between the assets shifts, the liquidity providers can take a loss on the currency risk.

In this chapter, we introduced various relevant technical aspects of XRP Ledger. In the remainder of this thesis, we discuss our research work.

Part II

Infrastructure Layer

3 | Centralized or not?

Topology Analysis of the XRP Ledger

"Darbo šaknys karčios, bet jo vaisiai saldūs."

The roots of labour are bitter, but its fruits are sweet.

Lithuanian Proverb

In this chapter, we conduct an exploratory analysis of the XRP Ledger topology. We collected 1,290 unique XRP Ledger network snapshots over two months. First, we inspect standard graph properties, such as node degrees, density and connectivity. We then examine complex network properties, such as degree distribution, small-world property and assortativity.

Secondly, we analyse how these properties change over time. We uncover a small group of nodes that act as a networking backbone. In addition, we observe a high network churn, with a third of the nodes changing every five days.

3.1 Introduction

Blockchain technology relies on peer-to-peer (P2P) networks that feature stateful connections between the participating nodes. The ease of construction and dynamic nature of these P2P networks enables efficient dissemination of information and exhibits highly variable network topologies. Blockchains are heavily dependent on the underlying P2P network, as the topological properties of these networks directly impact the security, reliability and performance of the blockchain system. Consequently, it is imperative to examine these networks to uncover potential vulnerabilities. Despite this, there is a notable paucity of research on the structural properties of P2P networks in blockchain technology, with most of the studies focusing on Bitcoin [37, 48] and Ethereum [56, 90, 85] networks. To the best of our knowledge,

we are the first to examine the XRP Ledger Overlay Network.

Unlike blockchains that obfuscate their topology, the XRP Ledger provides a public peer crawler API [109]. The API allows anyone to infer the topology of the XRP Ledger. Therefore, we can determine various structural properties of the network. These properties are vital to messaging efficiency, message propagation delay and resilience.

In this chapter, we conduct an in-depth analysis of the graph-theoretic properties of the XRP Ledger overlay network. Our main contributions are as follows:

1. We measure the structural properties of the network, as well as their evolution over time. We discover a central component of the network.

2. We examine the stability of the nodes and their uptime. We show that less than 50% of the nodes maintained their presence during the measurement period.

3. Finally, we show that the network may be vulnerable to Autonomous System failures.

We organise the remainder of this chapter as follows. In Section 3.2, we present relevant work conducted on network topologies. We describe the analysis methodology in Section 3.3 and the results in Section 3.4. Finally, we conclude our work in Section 3.6.

3.2 Related Work

We discovered a significant corpus studying cryptocurrency networks, predominantly Bitcoin and Ethereum. We provide a summary of these works in this Section.

Miller *et al.* [37] were the first to determine the topology of the Bitcoin network. The authors discovered *"extremely high-degree nodes"*, which persist in the network over time. Furthermore, the Bitcoin network is not purely random. Delgado-Segura *et al.* [48] inferred the topology of Bitcoin using orphaned transactions. Due to the limitations of their method, they performed measurements only in the Bitcoin *testnet*. Their results indicate that the testnet is not a random graph.

Paphitis et.al. [85] conducted a graph-theoretic analysis of several blockchain overlay networks. The results indicate that blockchain overlays have varying network properties and degree distributions. Despite the significant variance, there is a strong correlation between the node's session length and the degree. In addition, the networks have small average

shortest paths but are not small-world. Finally, the overlay networks are resilient to random node failures, but targeted attacks can considerably affect their connectivity.

Similar studies focus on the Ethereum blockchain. Zhao *et al.* [90] performed a temporal, evolutionary analysis of the Ethereum blockchain interaction networks. The authors found a link between anomalies in structural properties and real-life events. Furthermore, they discovered that the network expansion follows a preferential attachment model.

In a later study, Gao *et al.* [56] conducted a graph-theoretic analysis of the peer-to-peer layer of the Ethereum network. They discovered an abundance of nodes that do not contribute to the Ethereum network. Furthermore, they showed that the degree distribution does not follow a power law. In contradiction to the work of Paphitis *et al.*, the authors found evidence of small-world property.

The research conducted in the XRP Ledger context is predominantly on the Consensus Protocol. Chase *et al.* [47] provide a detailed description and analysis of the Consensus Protocol. They demonstrate that at least a 90% overlap of the UNLs is required to ensure network safety. In a later study, Christodoulou *et al.* [67] show when fewer than 20% of nodes are malicious, the overlap of UNLs can be relaxed. Otherwise, an overlap of 90-99% is required. In a similar study, Amores-Sesar *et al.* [65] demonstrate that, in the presence of Byzantine nodes, the ledger may fork under standard UNL overlap requirements. Furthermore, the authors show that a single Byzantine node may cause consensus protocol to lose liveliness.

In a different line of research, Roma *et al.* [75] studied the energy efficiency of an XRP validator. They found that the annual validator running cost is significantly lower than that of a miner.

Aoyama[79] provides a unique view of the XRP network from the perspective of its transactions. They found a clear divide between groups accepting transactions and groups receiving transactions.

3.3 Methodology

This section outlines the metrics we considered for the XRP Ledger topology analysis, and describes the data collection process.

3.3.1 Network Properties

Size Size is the network node count expressed as N and the number of links L connecting them. Size is a fundamental property which we use to derive other, more complex properties.

Degree, Average Degree & Degree Distribution The degree of a node is simply the count of peers the node has. We denote the degree of the *i-th* node as k_i. Note that we assume a single connection between any two nodes. In the case of a directed network, i.e. the link connecting two nodes has a direction, we compute incoming and outgoing links separately (k_i^{in} and k_i^{out}).

The average degree of a network is $\langle k \rangle = \frac{1}{N} \sum_{i=1}^{N} k_i$, which also separately expresses the average incoming and outgoing degree for directed graphs.

Finally, the network's degree distribution is the probability that a randomly selected node has a degree k. Since p_k is a probability, $\sum_{k=1}^{N} p_k = 1$. The degree distribution plays a vital role in network science. The network degree distribution impacts many of its properties, such as message propagation delay and the resilience of the network [38]. Random networks have binomial degree distributions, whereas real-world networks contain a small number of highly connected nodes that cannot be accounted for by random models [134].

Density In a real network, the node and link count varies greatly. The number of links will always lie between 0 and L_{max}, where $L_{max} = \frac{N(N-1)}{2}$ is the link count in a complete graph. Density is the ratio between L and L_{max}. Higher values indicate a denser network. In dense networks, messages have a lower propagation delay but at the cost of increased redundancy [38].

Path, Diameter, Shortest Path & Average Shortest Path A path between any two nodes in the network is simply the sequence of links between them. Its *length* is the size of the sequence. The shortest path $d_{i,j}$ between nodes i and j is the path with the minimum

number of links. There can be multiple shortest paths of the same length between two nodes. The shortest path never contains loops. The diameter of the network d_{max} is the longest shortest path in the network. I.e. it is the largest distance between two nodes in the network.

The average shortest path of a node s is the mean of all shortest paths from the node to every other node in the network. It is expressed as $d_s = \frac{1}{n(n-1)} \sum_{t \in V} d(s,t)$, where V is the set of nodes in the network, $d(s,t)$ is the shortest path between s and t and $n = |V|$. A short average path length facilitates rapid message dissemination between the network nodes. The XRP Ledger consensus protocol depends on user transactions reaching validators as quickly as possible. Therefore, a closely connected network is desirable.

Connected Components A node not connected to other nodes in the network would be of limited use. Thus from a utility perspective, there must be a path between every node in the network. A network is *connected* if some path connects every node pair, and *disconnected* otherwise. A *connected component* is a subset of nodes in the network connected by a path.

Clustering Coefficient The clustering coefficient represents the degree to which the node's neighbours connect. It quantifies the node's local link density. For a node i with degree k_i, the *local clustering coefficient is*: $C_i = \frac{2L_i}{k_i(k_i-1)}$. L_i represents the number links between the neighbours of node i. Each link between two neighbours of i forms a triangle. Therefore we can also say that L_i is the number of triangles i forms. A $C_i = 0$ means that none of node i neighbours link. A $C_i = 1$ indicates that all the neighbours connect, forming a clique.

The *global clustering coefficient* captures the total number of closed triangles in the network. $C_\Delta = \frac{3 \times NumberOfTriangles}{NumberOfConnectedTriples}$. A connected triple is an ordered set of three nodes forming a triangle. The factor of three in the numerator is because we count each triangle three times [6].

Degree Correlation The *degree correlation* captures the node's preference to form connections with others that are similar in some way [38]. In the context of this study, we consider similarity in terms of node degree. A network is *assortative* when nodes tend to connect to others with a similar degree. In a *disassortative* network, small-degree nodes

prefer to link with high-degree nodes, and hubs tend to avoid each other. Finally, a network is *neutral* when the wiring between the nodes is random.

The *degree correlation* impacts the robustness of a network [22]. In an assortative network, node removal causes little fragmentation, as high-degree nodes form a core group and are redundant. In contrast, disassortative networks are easier to fragment [38]. High-degree nodes connect to many small-degree nodes, forming a hub-and-spoke structure. The small-degree nodes become disconnected once a high-degree node fails.

Let e_{jk} define the probability that a randomly selected edge in an undirected network connects nodes with a degree *k* and *j*. We note that e_{jk} satisfies the following sum rules

$$\sum_{jk} e_{jk} = 1 \qquad \sum_{j} e_{jk} = q_k \qquad \sum_{k} e_{jk} = q_j \tag{3.1}$$

Where q_j and q_k are the probability that a randomly selected edge is a node with a degree j and k. It is

$$q_k = \frac{kp_k}{z} \qquad z = \sum_{k} kp_k \tag{3.2}$$

Where z is the mean degree of the network, and p_k is the network's degree distribution, i.e. p_k is the probability that a randomly selected node will have a degree k. The network is neutral when $e_{jk} = q_i q_k$, that is, small and high-degree nodes connect randomly without trend in e_{jk}.

Matrix e_{jk} carries degree correlations. Instead of inspecting the matrix, we can express the degree correlation as a single scalar coefficient *r* [21]

$$r = \frac{\sum_{jk} jk(e_{jk} - q_j k_k)}{\sigma_q^2} \tag{3.3}$$

where σ^2 is the standard deviation of the distribution q_k

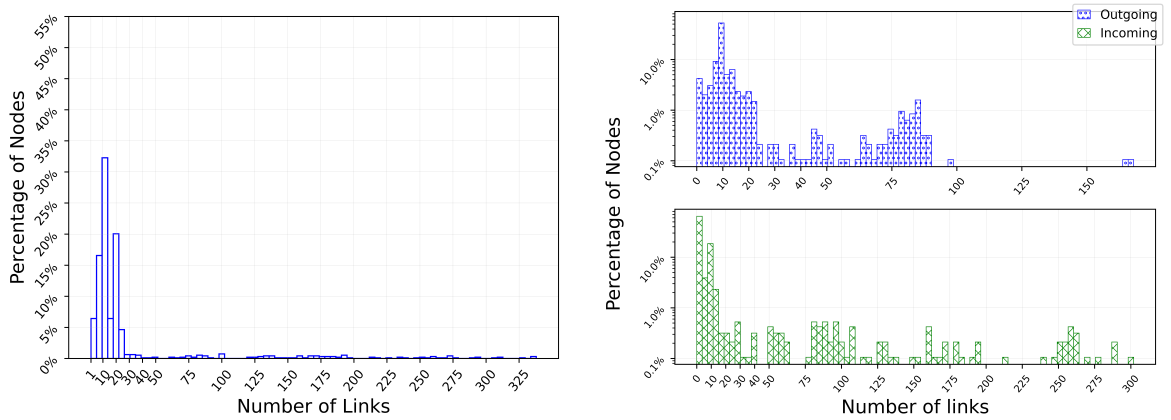
$$\sigma^2 = \sum_{k} k^2 q_k - \left(\sum_{k} k q_k \right)^2 \tag{3.4}$$

In general r varies between $-1 \leq r \geq 1$ [38]. Network is assortatative when $r > 0$. It is disassortative when $r < 0$. Finally, when $r = 0$, a network is neutral.

3.3.2 Data Collection

We captured XRP Ledger network snapshots using the XRP Ledger Crawler [119]. The crawler began by querying the peers of a single rippled server, with the starting node being r.ripple.com. It added the discovered nodes to a list and repeated the process for every node with a known IP address. We enriched the snapshot data with Autonomous System information. We collected network snapshots over two months, from May 1st, 2022, to March 1st, 2022, at one-hour intervals. In total, we collected 1,290 snapshots. The dataset generated by this study is openly available online for further research [120].

3.4 Network Analysis



(a) Total connection distribution.

(b) Outgoing (top) and Incoming (bottom) degree distribution.

Figure 3.1: Illustrative node degree distribution.

3.4.1 Network Properties

We summarize the basic properties of the XRP Ledger Network in Table 3.1.

Size The network is relatively small. We observed 948 nodes and 15,010 links on average. In comparison, Bitcoin has 50,000 nodes and Ethereum 12,000 nodes [85]. We measured a fluctuation of 2% in the total node count and 1% in the edge count.

	Mean	STD
Nodes	948.53	18.54
Edges	15010.26	508.92
In-Degree	15.82	45.62
Out-Degree	15.82	19.94
Density	0.03	0.00
Diameter	5.1	0.33
Avg. Shortest Path	2.31	0.03
Connected Component	1	0.00
Global Clustering Coefficient	0.76	0.02
Degree Correlation	-0.48	0.02

Table 3.1: Basic XRP network properties.

In&Out Degrees Each outgoing connection corresponds to an incoming one, and the nodes report only the active links (not the potential ones). Therefore, the means of incoming and outgoing degrees are equal. The standard deviation of incoming connections is 45.62. The difference in deviations suggests that some nodes in the network accept significantly more incoming connections than others.

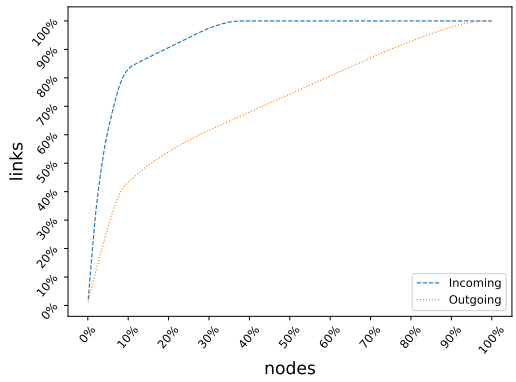
Connected Components XRP Ledger is consistently connected, as indicated by the single connected component and zero-value standard deviation. However, this may also be due to the nature of the crawler. The crawler can only discover the nodes that are members of the same connected component as the initial entry node. However, any node not connected to the core could not participate in the blockchain.

Network Density The XRP Ledger network has a density of 0.03. In comparison, the density of Bitcoin and Ethereum are 0.002 and 0.0006, respectively [85].

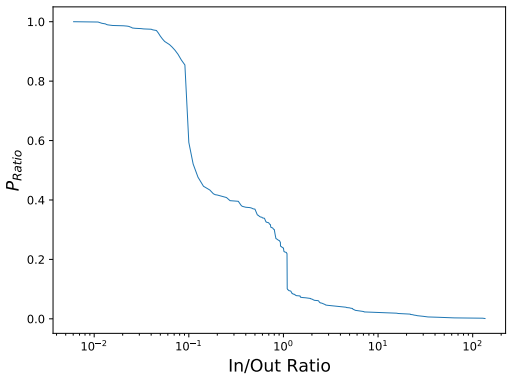
Clustering Coefficient The Global Clustering coefficient is 0.78. The low average shortest path and high clustering coefficient of the XRP Ledger Network suggest that it may exhibit the small-world property.

Degree Correlation The degree correlation coefficient for XRP Ledger is -0.48. In comparison, the degree correlation of an equivalent ER network is zero. We conclude that the XRP Ledger network is disassortative. It has a hub-and-spoke network structure and may

be vulnerable to targeted attacks.



(a) Connection Distribution amongst the nodes.



(b) In/Out Degree Ratio.

Figure 3.2: Representative network properties.

Degree Distribution

In Figure 3.1a, we illustrate the percentage of nodes (y-axis) with a given number of combined incoming and outgoing connections (x-axis). The distribution shape is similar to a gamma distribution with a long right tail. The majority of nodes, 32.5%, have between 10 and 15 connections. At the tail end, nodes have over 325 peers, six times more than nodes at the beginning of the tail.

We illustrate incoming and outgoing degree distributions separately in Figure 3.1b. The upper plot depicts the outgoing connections. The majority of nodes establish between 1 and 22 connections. The largest bin holds 50% of all the nodes, with ten peers. The spike reflects the default *rippled* configuration. At the time of the data collection, the default number of outgoing connections was 10. This value was since updated to 21 [107]. Other nodes connect to between 22 and 90 peers. We also found three outliers; two nodes with well over 150 and one with just under 100 connections.

In the lower plot, we depict the incoming connection distribution. The first bin contains 60% of nodes without incoming connections. There is no incentive to accept connections, but there is a server maintenance cost. Therefore, the majority of servers only establish outgoing connections. In addition, the first bin may also include validators. By default, for security reasons, they do not accept incoming connections.

The second largest group represents 19% of nodes with between 9 and 11 connections.

The remaining bins contain 11% of nodes. These account for the vast majority of the incoming connections in the network. Nodes with around 150 incoming connections are the hubs.

In Figure 3.2a, we illustrate the cumulative sum of incoming and outgoing connections. There are two outgoing connection groups. The first group, indicated by the exponential portion of the curve, holds nodes whose out-degree is above the mean. It contains $\approx 15\%$ of the nodes that account for 0% of all connections. The second group, indicated by the linear portion of the curve, holds the remaining 85% of the nodes. Finally, a deeper inspection revealed two outlier nodes with over 150 outgoing connections.

We similarly grouped the incoming connections. The first group, indicated by the sharp spike of the curve, dominates the overall network connectivity. It contains 11% of nodes with an in-degree above the mean. These nodes account for 85% of incoming connections. The second group, depicted by the short linear portion of the curve, contains 27% of nodes. They account for approx 15% of the incoming links. The final group, reflected by the plateau, holds nodes without incoming connections and accounts for the remaining 62% of nodes.

The incoming and outgoing connection distributions are heavy-tailed. However, they seem to have different shapes. We discuss which model best describes these distributions in Section 3.4.1. We also observe that a small subset of nodes holds the majority of connections. Our findings suggest that the network has a group of authoritative nodes.

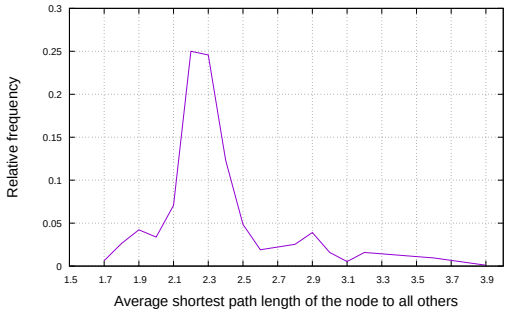


Figure 3.3: Distribution of the average shortest path length.

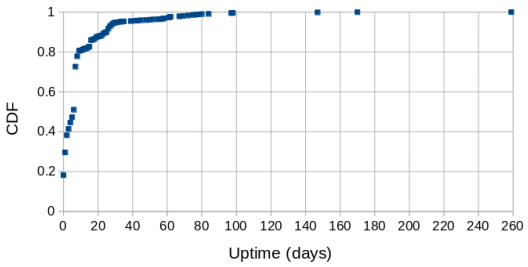


Figure 3.4: Node uptime CDF.

Scale-Free Property

Across scientific domains, it is often claimed that real-world networks are scale-free. Details vary, but in general, a network is scale-free when nodes with degree k follow a power-law

distribution $k^{-\alpha}$, where α is the scaling criterion $\alpha > 1$. However, other versions of this hypothesis are stricter, e.g. $2 < \alpha < 3$ [38]. Cohen *et al.* show that scale-free networks are highly resilient to random attacks but are vulnerable to targeted attacks [12]. Therefore, it is important to understand the type of degree distribution.

We used the *fitter* [102] Python library to find the most accurate model. We used the Akaike Information Criterion (AIC) and Bayesian Information Criterion (BIC) to determine the quality of a fit. A lower AIC or BIC value indicates a better fit. The analysis in Section 3.4.1 revealed that the in and out degrees have different distributions. Therefore we modelled the in, out, and combined distributions separately. We found that the in-degree was best captured by the *power-law* distribution, with α 1.2. On the other hand, the out-degree was best described by the *generalized normal distribution*, with a heavy, long tail. Likewise, *generalized normal distribution* fits the overall degree distribution the best.

The ubiquity of scale-free networks in the real world has been questioned [134]. Therefore, we avoid claiming that the XRP network is scale-free. However, our findings indicate that the XRP network is not random. Furthermore, the power-law distribution fit of the in-degree offers further evidence that the network relies on a subset of nodes for connectivity.

Small-World Property

The well-studied small-world property indicates that a short path connects any two nodes in the network [38]. An average shortest path *l* is short when $l \approx \frac{lnN}{ln\langle k \rangle}$, where N is the size of the network, and $\langle k \rangle$ is the average degree.

Manfred Kochen and Ithiel de Sola Pool [1] formalized the effect, which was popularized by the well-known Milgram experiment that inspired the *six degrees of separation* phrase.

Network G is small-world if it has a similar average shortest path length but a greater clustering coefficient than an equivalent random graph. Two graphs are equivalent when they have an equal number of nodes. Let L_g be the average shortest path length of G and C_g its clustering coefficient. In random graphs, equivalent properties are L_{rand} and C_{rand}. Network G is said to be small-world if $L_g \geq L_{rand}$ and $C_g >> C_{rand}$.

A quantitative measure of small-worldness is $\gamma_g = \frac{C_g}{C_{rand}}$ and $\lambda_g = \frac{L_g}{L_{rand}}$, where γ_g is the clustering coefficient ratio and λ_g is the average shortest path ratio of network G and an equivalent random graph. Then a measure of small-worldness is $S = \frac{\gamma_g}{\lambda_g}$. A network is

small-world when $S > 1$ [28].

We used the ErdsRényi (ER) model to generate random graphs. To ensure the robustness of the small-worldness calculation, we used Monte Carlo sampling of 1000 equivalent ER graphs. We measured $S = 8.3$ for the XRP Network. We, therefore, conclude that the XRP network has the small-world property.

In/Out Degree Analysis

Link analysis is a method to identify authoritative nodes in a network [15, 135]. We use it to identify selfish nodes that do not reciprocate the connections they establish by accepting incoming links. We express the link ratio as $\lambda = \frac{In+1}{Out+1}$, a ratio between the incoming and outgoing number of connections. All degrees are incremented by 1 to account for no incoming or outgoing connections. A high ratio $\lambda > 1$ suggests that a node is altruistic - it establishes more incoming connections than outgoing ones. Conversely, $\lambda < 1$ indicates nodes that consume more connectivity than they provide.

We illustrate the ratio distribution in Figure 3.2b. We observe that 15% of nodes have $\lambda << 1$. Interestingly, we find a significant percentage of nodes have a $\lambda = 0.09$. These nodes use the default *rippled* configuration, with ten outgoing and zero incoming connections. In contrast, only about 10% of nodes have more incoming than outgoing connections, and only 3% $\lambda >> 1$.

There are no direct incentives to participate in the XRP network. However, running a node that accepts incoming connections requires significant investment. Such a server has to be reliable and available. Therefore, most nodes connect to the network as *consumers*, and only relatively few behave altruistically. In the next section, we discuss the preference of nodes to connect to other similar nodes.

Average Shortest Path Distribution

Figure 3.3 illustrates the average shortest path distribution. The X-Axis is the average path length (in hops), rounded to the tenth. The Y-Axis is the percentage of nodes with the given path length. We observe that around 50% of the nodes have an average distance between 2.2 and 2.3 hops. The distribution has a bell-like shape with a long tail towards longer distances. IP networks have a similar distribution, although the average path length

is around nine hops [20]. The shortest path distribution suggests that the network's topology assists in timely message delivery.

Node Distribution over Autonomous Systems

Rank	AS number	AS name	XRP nodes
1	16509	Amazon.com	177
2	24940	Hetzner Online	115
3	14618	Amazon.com	71
4	8987	Amazon DS Ireland	70
5	396982	Google	52
6	8075	Microsoft Corporation	25
7	16276	OVH	23
8	14061	DigitalOcean	18
9	38895	Amazon.com	18
10	134963	Alibaba.com Singapore	17

Table 3.2: AS with the highest number of nodes.

The Autonomous System (AS) number is a 32-bit unique identifier. It represents a collection of IP networks administered by a single entity. We used the AS number to compute the node distribution per AS. In the remainder of this section, we discuss our findings.

Most systems contain only a handful of nodes, while a few AS have a large node count. Around 18% of discovered nodes did not reveal their IP addresses. Therefore, we do not know their AS details.

We split AS details across two tables to illustrate the heavy-tailed nature of node distribution between the AS. Table 3.2 shows the top ten AS by the number of nodes. These systems, owned by the largest cloud service providers (Amazon, Google, Microsoft), hold nearly 62% of the XRP Ledger nodes. Routing failures in one or more dominant Autonomous Systems may disrupt XRP Ledger operations. Therefore, a concentration of nodes may represent a weakness for the ledger.

The remaining 38% of the nodes are distributed evenly over 117 AS. Table 3.3 shows the number of AS possessing a given number of nodes. We observe that 84 AS only host one XRP node. As the node count per AS increases, the number of autonomous systems rapidly approaches 1.

Nodes per AS	Total number of AS
1	84
2	13
3	8
4	6
5	3
6	1
8	2

Table 3.3: Number of AS with a given number of nodes.

Node Uptime Distribution

The *rippled* software reports the number of seconds (uptime) it has been running. We plot the cumulative distribution function (CDF) of the reported uptime in Figure 3.4. The X-Axis depicts the uptime in days, rounded to the closest hour.

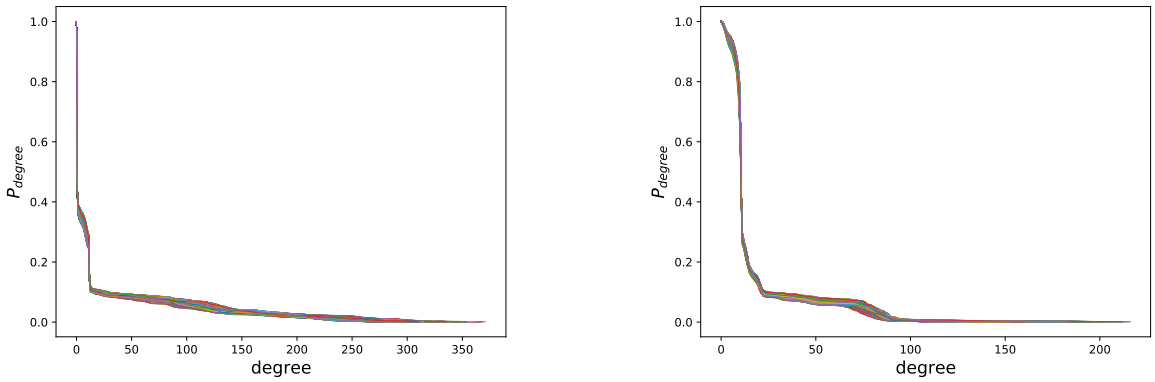
The average uptime is 9.7 days, with a standard deviation of 18.4 days. Just under 18% of nodes had an uptime of fewer than 12 hours, whereas the oldest node ran for 259 days. Approximately 18% of nodes reported an uptime between 20 and 60 days, and 2% were running for up to 80 days. We observed only a handful of nodes older than 100 days.

3.5 Temporal Analysis

In this section, we discuss the evolution of the network over time. We begin with a summary of the temporally stable properties. The small-degree node preference to connect to high-degree nodes remains constant over time. Likewise, the global clustering coefficient and average shortest path are stable. Furthermore, all network snapshots have the small-world property. These findings suggest that no significant disruptions occurred in the network during the observation period.

The relatively small change in the network's size had a non-negligible effect on the average incoming and outgoing degree, as indicated by the standard deviation. We dedicate the rest of this section to discussing these changes.

Degree Distribution We illustrate the degree complementary cumulative distribution (CCDF) in Figure 3.5. Both distributions have long tails, and their shapes are stable. However, we see some variance over time as indicated by the changing thickness of the plots.



(a) Incoming connections.

(b) Outgoing connections.

Figure 3.5: Temporal complementary cumulative degree distribution.

We plot the CCDF of incoming connections in Figure 3.5a. Our first observation is that consistently $\approx 60\%$ of nodes do not accept incoming connections. Likewise, we see negligible variance at the tail-end of the spectrum. We see slightly more variance in nodes close to the mean and nodes with a degree between 250 and 300. We observe the largest variance in nodes with an in-degree between 50-150. Our observations suggest that the nodes at the ends of the distribution are saturated. They cannot accept new peers. Therefore, nodes in the middle of the distribution handle the new connections to the network. Furthermore, the majority of new nodes do not accept incoming links.

We depicted the CCDF of the outgoing connections in Figure 3.5b. The long, thin tail of the distribution suggests the existence of a few stable nodes with a high number of outgoing connections. We see a much higher variance in the group of nodes with an out-degree between 50 and 100. Finally, the majority of new nodes had an out-degree under the mean.

Two versions of the *rippled* software came out during the data collection. Some observed variances may be explained by nodes leaving the network to update their version. However, overall the network has a stable member group.

In/Out Degree Analysis We display the degree ratio plot for all captured snapshots in Figure 3.6. We observe little change in the overall degree ratio. The majority of nodes establish more outgoing than incoming connections. Only ~10% of nodes establish more incoming than outgoing connections.

The lack of change in the shape of the curve confirms our initial observation that nodes do not reciprocate the connections they consume.

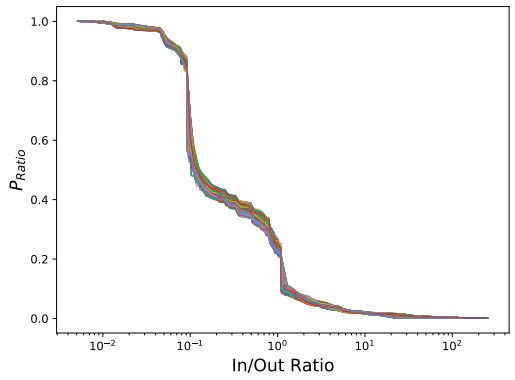


Figure 3.6: Temporal in/out degree ratio.

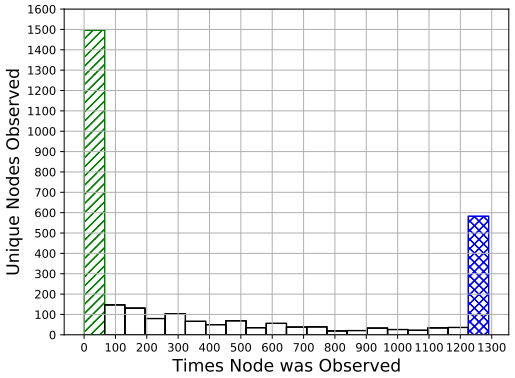


Figure 3.7: Frequency of node occurrences within a network.

Membership Stability Over the collection period, we discovered 3,000 unique nodes. In Figure 3.7, we outline the lifespan of these nodes. The green, striped bar indicates nodes with the shortest lifespan. These nodes were present in around 5% of all network snapshots. On the other side of the plot, the blue crossed bar represents the most stable nodes. They were present in at least 95% of all the snapshots. The remaining 1/5th of the nodes have a gradually decreasing lifespan.

The fully present nodes have an average in-degree of 26.1. In comparison, the nodes we observed in the 5% of snapshots have an average in-degree of 16. The difference between the values suggests that the fully present nodes are the ones that form the network backbone, which we discovered in Section 3.4.1.

We further analyzed the presence of the top 10% of the highest in-degree nodes in the network over time. The group of the first network snapshot contains 95 nodes. The last network snapshot group holds 98 nodes. However, 23 nodes or 24% from the first group, are missing in the second group. Four nodes changed their IDs but had the old IP addresses and similar degree profiles. However, we did not find the other 19.

Node Uptime Over Time

We measured the uptime of the 410 nodes present during every network crawl. Figure 3.8 presents two illustrative examples of the observed uptime. We limit our selection to the most representative nodes, in which we can observe clear patterns.

The top graph depicts the uptime graph of the 259 days old node we discussed in Section 3.4.1. There are two distinct features in the figure. The top line indicates that a server

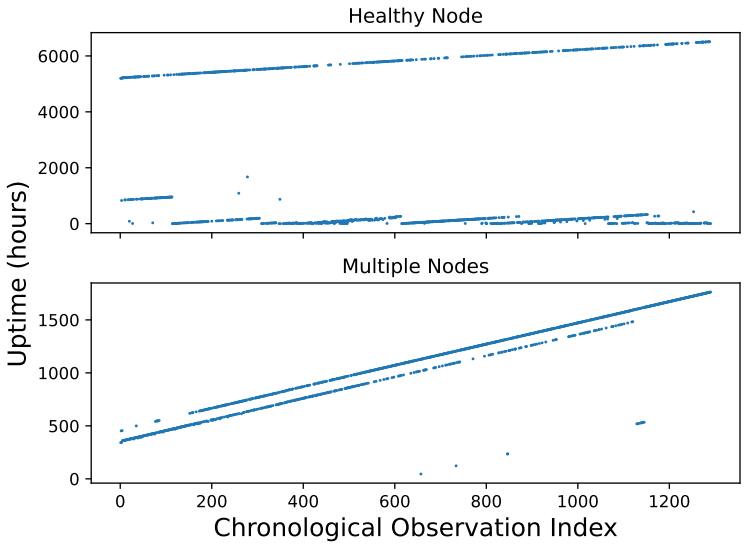


Figure 3.8: Illustrative uptime of representative nodes.

was operating without issues for the observation period. In contrast, the bottom feature suggests a server started and failed multiple times. These features suggest two separate instances of *rippled* running behind a single IP address.

The bottom plot offers a better illustration of two servers behind one IP address. There are two parallel lines of similar length. When we query the server uptime, we receive a response from one of the two servers. The fragmentation in the lines is information gaps caused by a different server handling the uptime request.

In all plots, we notice inexplicable uptime values. These could suggest that a new *rippled* instance started or, more worryingly, uptime reporting issues. However, we leave the study of these observations for future work.

3.6 Discussion & Conclusion

In the remainder of this chapter, we contrast the XRP Ledger and other blockchains.

3.6.1 Discussion

In Table 3.4 we outline key topology properties of XRP Ledger and other blockchains. With the exception of Ethereum and XRP Ledger, other blockchains are Bitcoin derivatives. At the time of writing all except XRP Ledger used *Proof-of-Work* consensus protocol.

	XRP Ledger	Bitcoin	Ethereum	Dogecoin	Litecoin	ZCash
Nodes	948	50,000	12,000	1,200	8,200	1,500
Edges	15,010	4.794×10^6	59,000	1.16×10^5	7.41×10^5	1.06×10^5
Density	0.03	0.002	0.0006	0.0805	0.0112	0.0617
Degree	15	92	4.3	74	104	56
Avg. Shortest Path	2.31	2.55	3.78	1.77	1.96	1.72
Global Clustering Coefficient	0.76	0.049	0.0022	0.28685	0.0735	0.3094
Degree Correlation	-0.48	-0.2	-0.02	-0.13	-0.01	-0.22

Table 3.4: Comparison of structural properties of various blockchains [85].

Our first observation is that XRP Ledger is the smallest network of them all. It's size only comparable to that of Dogecoin and ZCash. The other blockchains have significantly larger number of nodes. Interestingly, although ZCash and Do gecoin are of comparable size, they have a significantly large number of edges than XRP Ledger as captured by their density values. Unlike XRP Ledger, none of the other blockchain have the small-world property. Although the average distance is low in all of them, the clustering coefficient is not high enough to be considered small-world.

The XRP Ledger degree distribution has an exponential-like shape. We did not find conclusive evidence that it is scale-free. However, like other blockchains, the topology is not random [85]. Overall, the size of the XRP Ledger is consistent over time. However, we captured a significant amount of churn. Given these observations, we suspect many nodes join the network to conduct their business and leave shortly.

The XRP overlay network may be vulnerable to targeted attacks. We discovered the existence of a small subset of influential nodes that provide the backbone of the network connectivity. Furthermore, a malicious actor can use the publicly available topology to identify these nodes.

We revealed a vast disparity between nodes that accept incoming connections and nodes that do not. Link analysis showed that most nodes do not accept incoming connections. As a result, they increase the dependence on influential nodes and contribute to network centralization. We suspect that a lack of financial incentives contributes to this behaviour, as running a reliable server is costly. However, there are no rewards from the blockchain for doing so. Natural centralization is a common problem in decentralized peer-to-peer networks[30][23].

3.6.2 Conclusion

We use a publicly available crawler to capture 1,290 snapshots of the underlying overlay network over two months. We find that it is significantly smaller than other blockchain overlay networks. The nodes are connected via short paths and tightly clustered. Furthermore, the clusters tend to have a hub-and-spoke structure, as shown by the high assortativity of the network. Unlike other blockchain overlay networks, XRP has a small-world topology. Our results raise further questions about the security and vulnerability of the XRP network. Research works [12] [42] show that networks with a long-tail degree distribution are susceptible to targeted attacks. We continue the exploration of the XRP Ledger Graph Layer in Chapter 4, where we examine its robustness to network attacks.

4 | Federated Byzantine Agreement Protocol Robustness to Targeted Attacks

"Kiaušinis vištą moko."

The egg is teaching the hen.

Lithuanian Proverb

Federated Byzantine Agreement protocols use voting to reach a consensus. Each participant selects whom to trust in the network and effectively communicates with the trustees to reach an agreement on transactions. Most trustees must agree on the same transaction set to include them in the blockchain. However, disruptions to communication will prevent the trustees from reaching an agreement.

In this chapter, we propose a novel robustness metric to measure the Federated Byzantine Agreement protocol tolerance to network failures. We demonstrate that the XRP Ledger Consensus Protocol is vulnerable to targeted network attacks. An attacker has to disconnect only 9% of the highest-degree nodes to halt the blockchain. We propose a mitigation strategy which maintains critical XRP Ledger topology properties whilst improving the robustness by 36%.

4.1 Introduction

Consensus Protocols such as the one implemented in the XRP Ledger can tolerate two types of failures: *crash faults* and *Byzantine faults* [2]. Under Byzantine failures, a par-

ticipant may behave arbitrarily. They might respond correctly, not respond at all, or reply incorrectly. A significant corpus of research examines the Byzantine fault tolerance of the XRP Ledger [65, 47, 73].

In contrast, crash faults are less complex. A participant does not respond to and does not perform any operations. Percolation theory dictates that a sudden crash (absence) of a single node will have little to no impact on the network. However, a critical threshold of failures exists, after which the network fragments into isolated components [38]. Consider an attacker whose goal is to halt the XRP Ledger. Assuming that validators run the recommended secure configuration, an attacker is unlikely to target them directly. However, the attacker may disable other nodes until the validators "fall off" the network. In Chapter 3, we showed that a small subset of around 100 nodes forms the connectivity backbone of the XRP Ledger. This backbone makes the XRP Ledger especially vulnerable to attacks directed at these nodes.

In network science literature, a network's ability to continue providing its services when its nodes are absent is called *Robustness*. More strictly, it is the percentage of nodes that have to disappear from the network for the Largest Connected Component to have fewer than half of the remaining nodes [38]. Cohen *et al.* [10, 12] provided theoretic thresholds for scale-free network resilience to random failures and targeted attacks. The authors demonstrated that scale-free networks are highly resistant to arbitrary failures but are susceptible to targeted attacks. Salah *et al.* [136] extended the model for directed networks, while Balashov *et al.* [52] provided an optimal strategy for fragmenting scale-free networks and performance bounds on optimal attack strategies on scale-free networks. Furthermore, Rohrer *et al.* [62] crafted a measure of the attacker's advantage based on network topology and performed a categorical analysis of potential attack vectors.

In the context of blockchains, Seres *et al.* [76] performed a topological analysis of Bitcoin's Lightning Network (LN). They found that LN had a critical threshold of 14% in the face of targeted attacks. However, the authors did not provide defence strategies. Lee *et al.* [72] continued the analysis of the Lightning Network. The authors evaluated LN's robustness to several attack types and proposed defence strategies against them. They found that default configuration settings in the LN client led to centralisation and recommended changes to fix them. In addition, Zhao *et al.*. [90] conducted a temporal evolution analy-

sis of Ethereum network interactions. The authors found that the network growth follows a preferential attachment model, and they get sparser as they mature over time. In addition, the studied blockchains are not resilient against partitioning and message propagation delay attacks. Gao *et al.* [56] scraped the P2P layer of the Ethereum network and conducted a graph-theoretic analysis of the derived topology. They showed that the Ethereum network is resilient against random failures and targeted attacks.

Research conducted on XRP Ledger predominantly focuses on the XRP Ledger Consensus Protocol. However, the consensus protocol depends on reliable message delivery by the underlying peer-to-peer network. Disruptions to the network would undermine the reliability of such protocols. Therefore, we introduce a stricter alternative to Robustness, *Quorum Robustness*. It expresses the *Federated Byzantine Agreement* Protocol robustness to node failures, it is:

> *"The percentage of nodes to be removed, such that there are not enough trusted nodes to reach a consensus."*

To distinguish the two robustness metrics, throughout this chapter, we will refer to the classic *Robustness* metric as *Network Robustness* and our proposed metric as *Quorum Robustness*.

We summarise our contributions as follows:

1. We conduct an empirical analysis of the Network and Quorum Robustness of the XRP Ledger Peer-to-Peer.

2. We show the conditions under which network fragmentation will halt the consensus protocol.

3. We provide an effective defence strategy to improve the robustness of the XRP Ledger.

We organise this chapter as follows. We provided the necessary background information in Section 4.2. We describe the evaluation methodology and results in Sections 4.3 and 4.4. In Section 4.5, we showcase our mitigation strategy and discuss our findings in Section 4.6. Finally, we conclude our work in Section 4.7.

4.2 Robustness

Failure of a single node has a limited impact on the integrity of the network. However, the network fragments into multiple isolated components as more nodes fail.

4.2.1 Network Robustness

Robustness, sometimes called resilience, is the ability of a network to maintain its functions when some of its nodes are missing. It's quantified as the percentage of nodes that have to fail until the *Largest Connected Component* contains fewer than half of the remaining active nodes [38].

When measuring network robustness, nodes are removed using one of the two strategies: *random* and *targeted*. Depending on the network topology, they produce significantly different robustness results.

Random Strategy The random strategy assumes that all nodes can fail with an equal likelihood. Random strategy models typical node behaviour, such as crashes or restarts. The Scale-Free networks, i.e. those whose degree distribution follows a Power-Law, are known to be remarkably resilient to random breakdowns [10].

Targeted Strategy A targeted removal strategy is a model of a malicious attacker whose goal is to cause as much damage to the network as possible. The attacker can use the readily available network topology to identify authoritative nodes whose absence would cause the most damage to the network. Scale-free networks are especially vulnerable to targeted attacks [12]. Removal of only a few hubs causes the network to begin fragmenting. Continuing the attack breaks the network into small clusters rapidly.

4.2.2 Quorum robustness

The robustness metric previously discussed provides a threshold for complete network fragmentation, at which point it is considered non-functional. However, the core function of the XRP Ledger is to process user transactions using an FBA consensus protocol. Thus, an attacker may halt the XRP Ledger by preventing the FBA quorum from forming.

The version of the XRP Ledger advances when 80% of trusted validators agree on a set of transactions, and a participant receives the same new ledger version from 80% of its trusted validators. Participants in the XRP Ledger are recommended to use the UNL curated by the XRP Ledger Foundation [117]. If trusted validators become unavailable, the ledger may halt, or its resistance to Sybil Attacks [19] may weaken. In light of previously mentioned limitations, we propose a stricter definition of robustness called *Quorum Robustness*:

> *"The percentage of nodes to be removed, such that there are not enough trusted nodes to reach a consensus."*

In the remainder of this chapter, we compare and contrast the two robustness metrics and propose a mitigation strategy to improve these metrics for the XRP Ledger.

4.3 Methodology

In this section, we outline the methodology for evaluating the robustness of XRP Ledger.

XRP Ledger Topology We used the same dataset as in Chapter 3.

Synthetic Networks We compare the robustness of the XRP Ledger overlay network to other network topologies. We generated three topologically different but equal graphs. Graphs are equal when they contain the same number of nodes and edges. The robustness of random graphs is a well-studied topic [9, 32, 8, 25]. However, we include a random graph generated using the ErdsRényi (ER) model for completeness. Real-world networks are not random [38, 134]: their degrees are prone to follow an exponential-like distribution. Therefore, we compare the robustness to a scale-free network generated with the BarabásiAlbert model. As we showed in Chapter 3, The XRP Ledger topology is small-world, and the degree is power-law-like. We used Klemm-Eguiluz (KE) model [16] to generate a small-world, scale-free network. We use this network to reveal the existence of some latent properties of the XRP Ledger, which make it less robust to targeted attacks.

Node Selection We consider two node failure models: *random failures* and failures due to *targeted attacks*. We examined two metrics for target selection for targeted attacks: *node*

degree and *betweenness centrality*.

The *betweenness centrality* of a node *v* is the fraction of shortest paths that pass through *v*. Both metrics produce similar robustness values. Therefore, we only present the results of the degree metric.

After the highest-degree node fails, the overall network degree distribution changes. Thus we recalculate the priorities of each target after node removal. Furthermore, we assume the attacker cannot target validators directly.

Validator Selection XRP Ledger developers recommend running a validator connected to a cluster of trusted tracking servers. The tracking servers, in turn, connect to the remaining network and relay incoming and outgoing validator messages. As a result, the validators are not present in the crawled topology. A set of validators is required to measure the robustness of the XRP Ledger Consensus Protocol. However, as they are not in the network crawl, we label a random set of 34[1] existing nodes in the network as validators.

Measuring Robustness We compute the robustness with a Monte-Carlo simulation to ensure the randomly selected validators do not skew the robustness metric. The simulator works as follows: At each iteration, the simulator labels 34 randomly picked nodes as validators. It then computes the robustness metrics by removing nodes using one of the random or targeted node selection strategies. Once the network reaches the failure threshold, the simulator captures the percentage of nodes removed, resets the network state and begins the next iteration. It continues running until the standard deviation converges. The results presented in Section 4.4 are an average of the simulator results (including standard deviation).

4.4 Evaluation

We present the evaluation results in Table 4.1. The entries are the percentage of nodes (including standard deviation) removed before reaching the robustness threshold.

Random Failures Although random failures are well-studied, we include our measurements for completeness. We assume the validators are impervious to these failures. Other-

[1] At the time of writing, the recommended UNL contains 34 validators.

Network Type	Strategy	Network (std)	Quorum (std)
XRP	Targeted Attack	20% (7%)	9% (3%)
XRP	Random Failure	94% (2%)	84% (12%)
Scale-Free	Targeted Attack	78% (2%)	78% (9%)
Scale-Free	Random Failure	95% (5%)	91% (4%)
Random	Targeted Attack	86% (0.07%)	88% (3%)
Random	Random Failure	95% (0.00%)	92% (7%)
Klemm-Eguiluz	Targeted Attack	68% (3%)	64% (8%)
Klemm-Eguiluz	Random Failure	94% (1%)	90% (11%)

Table 4.1: The robustness of different network topologies.

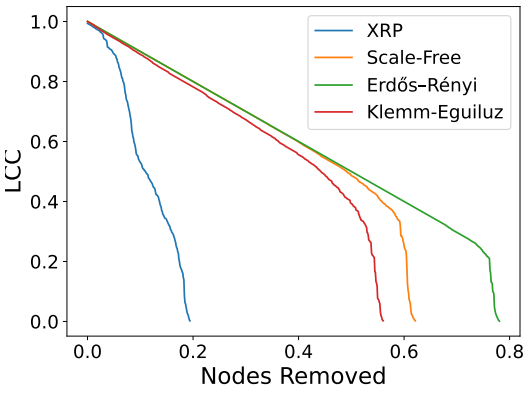
wise, the simulator removes all the validators from the network resulting in skewed results. Our findings indicate that the networks are highly robust under both metrics. The fragmentation occurs only when over 80% of the nodes fail. All networks, excluding the Erds-Rényi (ER) network, have a disproportionate number of small-degree nodes compared to high-degree nodes. Consequently, the likelihood of a failed node having a small degree is higher. The failure of small-degree nodes has a minimal impact on the overall robustness of the network, making fragmentation only possible after most nodes have failed.

Targeted Attacks Our evaluation shows that the XRP Ledger is the least resilient among the studied networks. The network fully fragments when approx. 20% of highest degree nodes fail. Furthermore, we measured that the consensus protocol may halt when only 9% of the authoritative nodes fail.

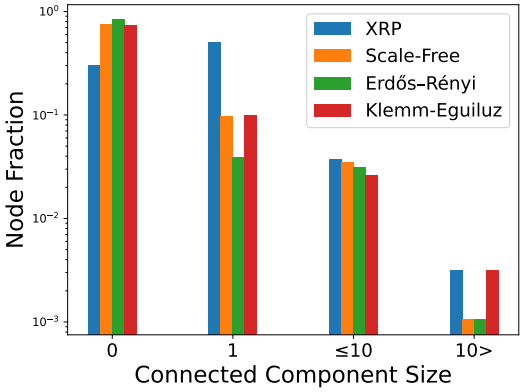
Around 100 authoritative nodes form the backbone of the XRP Ledger. The failure of a few of these nodes has a limited impact on the overall network connectivity, as multiple redundant paths connect these nodes. However, as the attack progresses, the overlay quickly begins to fragment.

In comparison, the Scale-Free and KE synthetic networks are much more resilient. We measured 78% robustness for both metrics in the Scale-Free network. In the KE network, we captured Network Robustness of 68% and Quorum Robustness of 64%. The Random network was the most resilient, with observed values of 86% and 88% of Network and Quorum robustness, respectively.

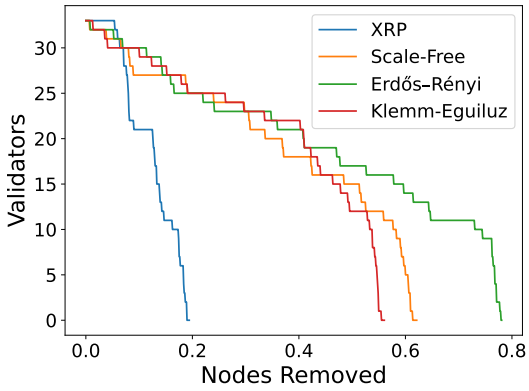
To improve the liveliness of the ledger, XRP Ledger developers implemented a *Negative UNL* feature [126]. It allows the XRL Ledger to make forward progress in the event of a



(a) Largest Connected Component Size.



(b) Network post fragmentation.



(c) Validator Count.

Figure 4.1: Illustrative example of network property degradation under targeted attacks.

partial outage. The participants adjust their effective agreement threshold based on which validators from their UNL are operational. For example, if only 70% of UNL members are available, a validator will lower its consensus threshold to 70%. The lower bound for negative UNLs is 60%. Under these conditions, the quorum robustness increases to 11% with a standard deviation of 7%, but still vulnerable when compared to other topologies.

Standard Deviation We observed varying standard deviation values for all networks irrelevant to the robustness metric. In our simulation, given some topology, the only changing variable is the set of *validator* nodes, which the simulator will ignore when selecting which node to remove. When considering *Quorum Robustness*, the existence of standard deviation suggests that the location of the validators in the network topology has a non-negligible effect on the robustness of the FBA protocol. Therefore, an optimal topological position for the validators which maximises the *Quorum Robustness* might exist for any given topology.

Effects of targeted attacks The removal of authoritative nodes affects various properties of the network. We illustrate the largest connected component size (LCC) degradation in Figure 4.1a. The X-Axis indicates the fraction of nodes removed, while the Y-Axis refers to the relative LCC size. Initially, all networks are connected, and thus the LCC size is one. We observe that in comparison to other networks, XRP Ledger deteriorates rapidly. Scale-Free, KE and ER show a significantly slower fragmentation rate, but all three networks fragment immediately once they reach a critical failure threshold.

In Figure 4.1b, we provide the connected component size distribution after the network fragment. The X-axis displays the size of the connected component, and the Y-axis represents the fraction of nodes accounted for by each. The zero-size column accounts for the removed nodes.

The results indicate that the XRP Ledger network fragments earlier and to a greater extent than the other networks, as demonstrated by the 401 individual nodes. It is worth noting that the quorum is lost well before the network collapses, as depicted in Figure 4.1c. The shape of the figure is similar to that of the LCC size degradation, reflecting the relationship between the LCC size and the number of validators in it.

4.5 Mitigation

In this section, we propose a property-preserving strategy to improve the XRP Ledger *network* and *quorum* robustness.

The XRP Ledger topology is small-world and highly dissasortative [132], we argue that these properties are critical to the performance of the XRP Ledger and thus must be preserved by the mitigation strategy.

Small-World The small-world property pertains to the observation that the average shortest path length is small relative to the size of the network. At the same time, the network exhibits a high degree of clustering. This combination of local clustering and global connectedness is what characterizes small-world networks [38].

Assortativity Network assortativity refers to the tendency of nodes in a network to connect to other nodes with a similar degree. A network is said to be assortative if high-degree nodes

preferentially connect with other high-degree nodes and low-degree nodes tend to connect with other low-degree nodes. Conversely, a network is disassortative if high-degree nodes connect with low-degree nodes and vice versa, forming a *hub-and-spoke* structure. As we showed in Section 4.2, disassortative networks are vulnerable to targeted attacks [38]. High-degree nodes connect to many small-degree nodes, forming a hub-and-spoke structure. The small-degree nodes disconnect once a high-degree node fails. This problem is made worse by the new *quorum robustness* metric, as it reduces the failure threshold of the XRP Ledger.

Trade-off The XRP Ledger hub nodes create the high disassortativity and the small-world property. However, the hubs rapidly disseminate messages across the network. Thus, they are vital to the healthy operation of the blockchain.

This reliance on the hub nodes creates a trade-off in the network. Most nodes depend on these hubs for access to the XRP Ledger. When they fail, the network fragment. In other words, while the hub nodes are essential for the efficient functioning of the XRP Ledger, they also represent a potential point of failure.

4.5.1 Mitigation Strategy

Algorithm 1 Selecting a new slot for node *N*.

```
1:  procedure SLOTS(S : slots[], desiredRatio, desiredPeers)
2:      S ← sortAscending(S)
3:      while True do
4:          smallPeers, highPeers ← N.peersByDegree()
5:          ratio ← |smallPeers| ÷ |highPeers|
6:          if |N.peers()| > desiredPeers then
7:              if ratio < desiredRatio then
8:                  N.replace(highPeers.first(), S.first())
9:              else
10:                 N.replace(smallPeers.first(), S.last())
11:         else
12:             if ratio < desiredRatio then
13:                 N.connect(S.first())
14:             else
15:                 N.connect(S.last())
```

Assumptions We developed the mitigation strategy for improving the robustness of the XRP Ledger with the following assumptions in mind:

- The small-world nature of the XRP Ledger is considered critical to its function. Thus the strategy must preserve it.
- We assume there are enough incoming connection slots to satisfy the demand, and the slots available for a node do not include existing peers.

We define a node as *high-degree* when it has exactly or more than *100* peers. Otherwise, the node is *small-degree*. This threshold covers the top 9% of nodes whose removal would cause the quorum failure.

Mitigation Approach Based on the previously mentioned assumptions, we propose a mitigation strategy to maintain a ratio between low-degree and high-degree peers. To illustrate this ratio, consider a node with *ratio* of three. For each high-degree peer, the node will maintain three low-degree peers. The strategy increases the assortativity of the XRP Ledger, thereby improving its robustness while preserving its small-world property. In case of an attack, the failure of hub nodes will result in an increased average shortest path, but it will not halt the consensus process.

Our proposed mitigation strategy involves several modifications to the *PeerFinder* module of the *rippled* server. The first change is to extend the slot advertisements to include the current degree of the node. Although the degree information may become outdated, the short lifespan of the slot advertisements will result in a negligible impact.

We detail the new peer selection procedure in Algorithm 1. The algorithm calculates the low-degree and high-degree peer ratio and then makes connections accordingly. The algorithm replaces existing peers when it has the desired number of peers but not the ratio. Otherwise, it connects to new nodes. If the actual ratio is lower than the desired ratio, the node connects to a low-degree node. Otherwise, the node connects to a high-degree slot.

4.5.2 Evaluation

Robustness We illustrate the impact of the mitigation strategy in Figure 4.2. We computed these metrics by following the methodology outlined in Section 4.3. For brevity, we restrict our results to the XRP Ledger. The X-axis in the figure displays the balance between low-degree and high-degree peers, where a ratio of 3.0 implies three low-degree peers per each high-degree peer. The Y-axis shows the percentage of nodes removed through a

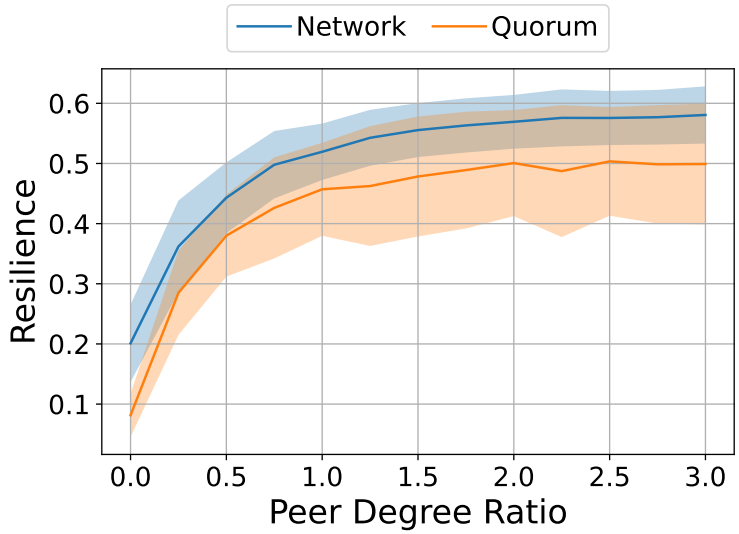


Figure 4.2: Illustrative example of the mitigation strategy impact on XRP Ledger robustness.

targeted removal process before network fragmentation occurs. The shaded area indicates the standard deviation. The results offer insights into the efficacy of our proposed mitigation strategy in enhancing the stability and robustness of the XRP Ledger. The optimal ratio is 1:1, resulting in 52% network robustness and 45% quorum robustness. Although higher ratios offer a slight increase in robustness, the improvement rate slows.

	Original	**Rewired (1.0)**
Assortativity	-0.46	-0.21
Avg. Shortest Path	2.33	2.34
Avg. Clustering Coefficient	0.74	0.39
Diameter	5	4
Avg. Low/High Deg. Peer Ratio	0.16	0.91

Table 4.2: XRP Ledger properties after rewiring.

Graph Properties In Table 4.2, we summarize the effects of the mitigation strategy using a ratio of *1.0* on various network properties. We reduced the assortativity by half whilst preserving the small-world property, as indicated by the Average Clustering Coefficient and the Average Shortest path. Interestingly, we also reduced the network diameter. We observe that the algorithm did not achieve the exact ratio across the whole network. Just under half of the nodes have an *odd* number of peers. Therefore, they cannot reach an exact 1:1 peer ratio.

4.6 Discussion

4.6.1 Network Robustness

The *network robustness* of the XRP Ledger is 20%. In contrast, the *network robustness* of Bitcoin and Ethereum are 6% and 4% respectively [85]. As these blockchains do not use communication-based consensus, we don't calculate their *quorum robustness*.

However, note that these blockchains are orders of magnitude larger than the XRP Ledger. For instance, Bitcoin has 50,000 nodes, and Ethereum has 12,000 nodes [85]. In contrast, XRP Ledger has only approx. 950 nodes [132]. The other blockchains exhibit a lower robustness percentage. However, they are safe against network-based targeted attacks due to their size.

4.6.2 Quorum Robustness

The stability of the Federated Byzantine Agreement (FBA) protocols depends on the robustness of the underlying peer-to-peer network. The topology of the XRP Ledger is highly disassortative, with small-degree nodes tending to connect to high-degree nodes [132]. The failure of high-degree nodes leads to the progressive disconnection of small-degree nodes from the network, causing a cascading network failure.

Validators in the XRP Ledger, when configured following recommended best practices, typically have a small degree and thus depend on hub nodes for access to the rest of the network. Our experiments have shown that the failure of approximately 9% of high-degree nodes will halt the consensus process. The standard deviation we observed also indicates that the topological position of validators has a significant effect on the robustness of the blockchain, potentially making it more or less robust.

While the introduction of Negative UNLs results in a marginal improvement in robustness of 2%, this is insufficient as it does not address the underlying disassortative structure of the XRP Ledger.

4.6.3 Mitigation Strategy

A simple yet effective strategy to improve the robustness of the topology is to replace some of the existing connections to high-degree nodes with those to low-degree nodes. By main-

taining a 1:1 ratio of connections to low and high degree nodes, we improve the *quorum robustness* by approx. 36%.

However, our solution is not without limitations. We assume that low-degree nodes will have available open slots to accept new incoming connections. It may not be the case in reality. Link analysis of XRP Ledger identified that most nodes do not reciprocate their connectivity. I.e. they establish outgoing but do not accept any or only accept a small number of incoming connections [132]. By design, there are no direct incentives to participate in the XRP Ledger [114]. However, running a node that accepts incoming connections requires significant investment. Such a server has to be reliable and available. Therefore, in reality, there might not be enough available open slots owned by small-degree nodes.

4.7 Conclusion

XRP Ledger Consensus Protocol relies on a robust overlay network for message dissemination. Disruptions to the network will prevent the validators from reaching a consensus. In this chapter, we have shown that in the context of classical *Network Robustness*, 20% of highest-degree nodes have to fail to fragment the network. However, this is a complete network breakdown. Percolation theory dictates that when high-degree nodes fail, nodes that use them to gain access to the network will also disconnect. In addition, to disrupt the XRP Ledger, we only need to prevent the validators from forming a quorum. Therefore, we proposed a novel *Quorum Robustness* metric, which captures the percentage of nodes that have to fail so that enough validators disconnect from the network that a quorum cannot form. We demonstrated that when approx. 9% of the highest-degree nodes fail the XRP Ledger Consensus Protocol will halt.

We demonstrated that the small-worldness of XRP Ledger is crucial for efficient message dissemination. Therefore, we implemented a mitigation strategy which maintains the core small-world property. Our solution increases the *Quorum Robustness* of the XRP Ledger to 45%.

Part III

Network Layer

5 | *Pemcast*

Probabilistic Edge Multicast Routing for the XRP Ledger

"Ir mažas kelmas išverčia vežimą."

Even a small stump overturns a cart.

Lithuanian Proverb

The XRP Ledger relies on a trusted set of validator nodes to advance the ledger history. Nodes use flood-based broadcasting to disseminate messages. Flooding offers strong message delivery guarantees at the cost of high network utilisation caused by duplicate messages.

In this chapter, we present PEMCAST, an application layer algorithm for efficient one-to-many message routing. The algorithm leverages limited topology awareness and application layer multicasting to deliver messages in the network. The evaluation shows that compared to flooding and gossiping algorithms, PEMCAST can maintain similar reliability whilst generating significantly less redundant traffic.

5.1 Introduction

The XRP Ledger servers use a broadcast algorithm to disseminate messages. When a server receives a new message, it forwards (by flooding) it to all nodes except the sender. Flood-based protocols are reliable as they explore every path in the network. However, they generate numerous duplicate messages. Naumenko et al. [61] reveal that up to 44% of Bitcoin network traffic is redundant.

Gossip protocols reduce the amount of redundant traffic generated while propagating messages in the network. The term "gossip protocol" was coined by Alan Demers in 1987 [4], who was studying methods to disseminate information in unreliable networks. Gossip pro-

tocols were extensively studied at the beginning of the century. Due to the advances in blockchain technology [66, 77], they are receiving a renewed interest.

Bimodal Multicast [7] was one of the pioneering works to combine multicasting and gossiping to disseminate messages in large-scale distributed systems.

The algorithm has two stages: (1) the message is sent using IP-Multicast, and (2) participants engage in gossip routing to deliver messages lost in the first stage. This approach has well-known drawbacks. IP-Multicast is not widely deployed [11]. Furthermore, the algorithm uses two different protocols, which introduces unnecessary complexity. Lightweight Probabilistic Broadcast [13] (*lpbcast*), is a decentralised probabilistic broadcast algorithm. Nodes maintain a local view of a *fixed* size whose members they obtained randomly. When every node in the network is known by multiple other nodes fault tolerance is preserved. In a typical gossip-based algorithm, a node sends a message to a random subset of its 1-hop neighbours. The size of the set is often called a *fanout*. Leitao *et al.* [31] showed a trade-off between the fanout and protocol reliability and an inverse correlation between fanout and message redundancy. Scribe [18] is an application layer multicast infrastructure built on top of a Pastry [17] overlay network. Scribe uses multicast groups with multiple senders. Scribe constructs a distribution tree for each group. In the tree, some nodes serve as rendezvous points and root of the multicast tree. Non-root nodes are aware of their parent node and actively monitor the health of that node. Such an approach requires regular heartbeat messages from the root node to notify its children that it is still alive, which generates a high number of control messages when they are not piggy-backed on data messages.

GossipSub [77] is a publish-subscribe messaging system. It constructs an overlay network (*mesh*) per topic. Messages are broadcasted in the mesh. Furthermore, nodes gossip information about the messages they have to nodes not part of the *mesh*. The actual messages are cached so that nodes receiving the gossip can request the messages with a control request. Distributing messages via routed spanning trees is another popular method. [45, 43, 46]. These algorithms rely on a central node in the network to behave as a root of the spanning tree. However, this introduces a central failure point.

In 2019, Craig et al. implemented a forwarding state reduction mechanism for multi-tree multicast by using Bloom filters. However, their proposal only works in Software Defined Networks [54]. In 2021, Newport et al. designed an asynchronous algorithm to implement a

new gossip strategy in smartphone-based peer-to-peer networks. [83]. It is one of the most recent papers on gossiping. However, it targets periodic communication, which is not the case in the XRP network.

In this paper, we propose a novel probabilistic multicast *application layer* routing algorithm - *pemcast*. It relies on partial network views (neighbourhoods) and multicasting to distribute messages with less redundant traffic. Our key contributions are as follows:

1. We propose a novel application layer routing algorithm which uses limited topology awareness and application layer multicasting to achieve efficient message routing.

2. We designed distinct metrics to evaluate the performance of *pemcast*, flooding and probabilistic broadcast algorithms.

3. We conduct an experimental evaluation using the existing XRP network topology. We demonstrate that *pemcast* in comparison to flooding and gossip-based protocols, can: (1) propagate a message to multiple targets whilst generating fewer duplicates, (2) send the message to fewer nodes not involved in the broadcasting of a message to a given destination.

We arrange the remainder of this chapter as follows. In Section 5.2, we describe the algorithm and discuss the methods to reduce network flooding. In Section 5.3, we evaluate *pemcast*, and discuss the simulation results. We conclude and discuss future work in Section 5.4.

5.2 Probabilistic Multicast Routing

5.2.1 Algorithm Design

Goals

We designed *pemcast* with three goals in mind:

1. Minimise the number of duplicate packets created whilst delivering a message. We consider a message duplicate when some node *u* receives it more than once.

2. Reasonable length paths to the destination node lead through a subset of nodes in the network. Our goal is to reduce the portion of the network explored during message propagation.

3. It is inevitable that nodes in the network will fail, and the neighbourhood view will become stale. Messages must be delivered even when there are inactive nodes (as defined below) in the network.

On the one hand, typical multicast protocols based on a shared tree or source-rooted trees would significantly reduce the amount of traffic. On the other hand, trees have a high convergence time recovering from failures and must hold numerous memory states. In addition, only a single path exists between any two nodes in the tree. A node could easily interfere with traffic to a given destination. Probabilistic multicast ensures that a message travels to the destination across multiple paths. For these reasons, we do not consider tree-based techniques, and we do not evaluate such protocols in Section 5.3.

Assumptions

We make several assumptions about the behaviour of the nodes.

1. The nodes are not Byzantine faulty. A node can either receive a message and respond to it correctly or not respond at all.

2. The nodes can be either active or inactive. An active node correctly executes the protocol. An inactive node does not respond to received messages. Furthermore, other nodes are not aware of its state.

Overlay Network

Formally we define the network in which *pemcast* is running as an unweighted, bi-directional simple graph $G = (V, E)$, where V is a finite set of nodes *u*, and $E \subset V \times V$ a set of links connecting the nodes. A unique, cryptographically secure identifier identifies each node. Peers (i.e., 1-hop neighbours) of *u* are nodes with which *u* has a direct link $P(u) = \{v | \{u, v\} \in E\}$.

Each node in the network maintains a view of all nodes up to *r* hops away. The view is called *neighbourhood* of *u*, and *r* is its radius. neighbourhood of *u* is rooted in *u*, we define it

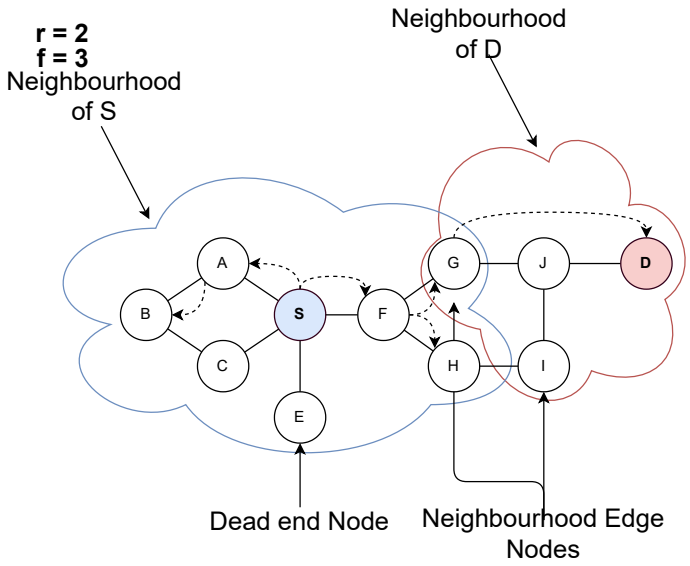


Figure 5.1: Illustration of pemcast elements.

as $N_r(u) = \{v | d_v(u) \leq r\}$; *r* is the upper bound on the shortest path length, and $d_v(u)$ is the length of the shortest path between nodes u and v. *neighbourhood edge* is a set of nodes in the neighbourhood that are exactly r hops away or are end-vertices.

We illustrate a neighbourhood in Figure 5.1[1]. A neighbourhood of node *S* has a radius of 2. $N_2(S) = \{A, B, C, E, F, G, H\}$, and the edge nodes are $\{B, E, G, H\}$. Note that G belongs to the neighbourhood of both S and D.

Neighbourhood Maintenance

When a new node N' joins the network, N' builds its local neighbourhood. Nodes up to r hops away from N' add it to their neighbourhood. When N' leaves the network, other nodes remove it from their neighbourhoods.

A membership discovery algorithm is responsible for managing this process. The details of these algorithms are outside the scope of this paper. Instead, we defer the reader to previous works [14, 24, 27].

Control Parameters

pemcast is controlled by two parameters *fanout* and *neighbourhood radius*.

- **Fanout** affects the number of neighbourhood edge nodes selected by the sender at

[1] For simplicity, we have omitted highlighting neighbourhoods of all other nodes in the network.

each multicasting step. The trade-off is between reliability and redundancy. Higher values increase the reliability at the cost of increased redundant traffic.

- **neighbourhood radius** controls the maximum length of the shortest path from the root node to every other node in the network. Higher values create larger neighbourhoods, which reduce the amount of traffic. However, large neighbourhoods will have a high membership maintenance cost.

pemcast execution consists of two phases. (1) *Path Discovery* - during which a message is transmitted from an arbitrary source node to some destination node using a combination of multicasting and source routing. (2) *Path Establishment* - the fastest discovered path to the destination is fixed in place. Subsequent messages between the source and destination nodes travel across this path. In the rest of this section, we discuss the phases in more detail.

Algorithm 2 Select edge nodes at node u

```
1:  procedure SELECTEDGENODES
2:      targets ← []
3:      edgeNodes ← getEdgeNodes()
4:      for |targets| < fanout ∧ !edgeNodes.empty() do
5:          node ← edgeNodes.popRandom()
6:          path ← getShortestPath(node)
7:          if |path| < neighbourhoodRadius then
8:              continue
9:          if path.contains(m.sender) ∨ path.contains(m.source) then
10:             continue
11:         targets[path[0]].append(path[l_path − 1])
12:     return targets
```

5.2.2 Path Discovery

During path discovery, a node can have one of the following roles: *Source* (S) - a node from which the message originates. *Destination* (D) - the final recipient of the message. *Sender* - node that multicasts the message to the edge of its neighbourhood. *Receiver* - a node on the neighbourhood edge that receives the message. *Forwarder* - any node simply forwarding the message.

A unique path ID PID_{SD} identifies a path between *source* and *destination* nodes. The path ID uniquely combines the node IDs, irrespective of message direction.

Nodes maintain two path tables: *pending paths* and *established paths*. Both tables use the path ID for indexing. The *Pending Paths* table holds candidate paths between a source and destination nodes. For example, the pending paths table for node F for PID_{SD} is $[(S, G), (S, H)]$. The *Established Paths* table holds confirmed paths. For example, $PID_{SD} \rightarrow (S, G)$.

Entries in both tables expire after a period of inactivity. Entries in the established paths table expire when a node does not observe any messages. Similarly, entries in the pending paths expire when the path is not confirmed.

Nodes maintain a view of nodes up to *r* number of hops away. The neighbourhood is stored as a graph. The graph allows nodes to retrieve paths to other nodes in their neighbourhood. In the remainder of this section, we describe node roles in greater detail.

Source Node

Algorithm 3 Multicasting path request message m from node u to the edge of $N_n(u)$

```
1: procedure PATHREQUEST
2:     targets ← selectEdgeNodes(m)
3:     for t ∈ targets do
4:         m′ ← copy(m)
5:         m′.destinations ← t.destinations
6:         sendMessage(m)
7:         p_ID ← newPathID(m.source, m.destination)
8:         addPendingPath(t.peer, p_ID)
```

The role of the source node *S* is to deliver message *m* to the destination node *D*. If a path between S and D exists in the *Established Paths* table, S sends the message over it. Otherwise, *S* initiates the path discovery process.

If S finds at least two paths between itself and D in its neighbourhood, S routes the messages over every found route. Otherwise, S proceeds to multicast the message as follows.

The source node constructs multicast sub-trees for a subset of neighbourhood edge nodes, as depicted in Algorithm 2. First, S selects neighbourhood edge nodes. Next, until a *fanout* number of target nodes are selected, *S* picks a random candidate node and computes the shortest path to it. S skips the candidate node if (1) the path's length is shorter than the neighbourhood radius, implying the node is a dead-end, or (2) the path contains the source

or the sender nodes. Otherwise, S adds the candidate to the set of *targets*. A single entry in *targets* is a mapping $(peer, destinations)$, from some peer P of S, to a list of edge nodes that can be reached through P. For example, for node S this could be $(F, [G, H])$, as both G and H nodes are reached via F.

Once S constructs the multicast sub-trees, it sends the messages as depicted in Algorithm 3. For each mapping *m*, the source node prepares a new message, sets its destinations field to $m.destinations$, sends the message to $m.peer$, and finally adds an entry to the list of potential paths.

Forwarder Node

A node determines its role in message handling by checking whether its identifier is in the *destinations* field of a message. If the identifier is present, the node behaves as a *receiver*, and *forwarder* otherwise.

The *forwarder* node is responsible for forwarding messages to their respective destinations. For each entry in the message's *destinations* field, the forwarder: (1) computes the shortest neighbourhood path to it, (2) computes a multicast mapping, as outlined in Algorithm 2, and (3) forwards a copy of the message to the next node on the path.

Receiver Node

The node behaves as a *receiver* when its identifier is present in the *destinations* field of a message. The *receiver* has two tasks: (1) to unicast the message to the final *destination* node if it is a member of the *forwarder's* neighbourhood. (2) to multicast the message to the edge of its neighbourhood by executing Algorithms 3 and 2

5.2.3 Path Establishment

When the destination node D receives a message addressed to itself, it responds with a path acknowledgement, which is sent back to S via the reverse path. Each node on the reverse path executes Algorithm 4 upon receiving a path acknowledgement message. First, a node retrieves the pending path entry from the *Pending Paths* table using the path and sender IDs. The *sender ID* is the identifier of the node that sent the acknowledgement. It identifies which pair of nodes were involved in forwarding the message from the source to the

Algorithm 4 Handling of path reply message m in node u

1: **procedure** PATHREQUEST
2: $ID_p \leftarrow newPathID(m.source, m.destination)$
3: $pendPath \leftarrow getPendingPath(ID_p, m.sender)$
4: $next \leftarrow pendPath.from = m.sender \ ? \ pendPath.to : pendPath.from$
5: $m' \leftarrow copy(m)$
6: $m'.sender \leftarrow ID_u$
7: $m'.target \leftarrow next$
8: $sendMessage(m')$
9: $Est_u[ID_p] \leftarrow pendPath$
10: $Pnd_u[ID_p] \leftarrow []$

destination. One of the nodes in the pair is always the *Sender ID*. Next, the node forwards the path acknowledgement message to the peer, which is not the sender. For example, node F has a pending path entry (S, G). When F receives a path acknowledgement from G, it will know to forward the message to S. Finally, the node moves the confirmed entry to the *Established Paths* table and removes it from the *Pending Paths* table.

5.2.4 Minimising Flooding

In this section, we discuss how the algorithm design addresses redundant traffic.

Effects of the neighbourhood

When the destination node is in the neighbourhood of some node *u*, the node can terminate the multicasting process and proceed to route the message directly to the destination. The neighbourhood's size impacts how soon the message can be routed directly to the destination. Furthermore, multicasting the packet to the edge of the neighbourhood reduces the volume of created traffic. By multicasting, the *forwarder* nodes only propagate the message to selected peers. Therefore, they do not introduce redundant copies of the message to the network. As a result, nodes can use higher fanout values than traditional gossip algorithms without creating duplicate packets.

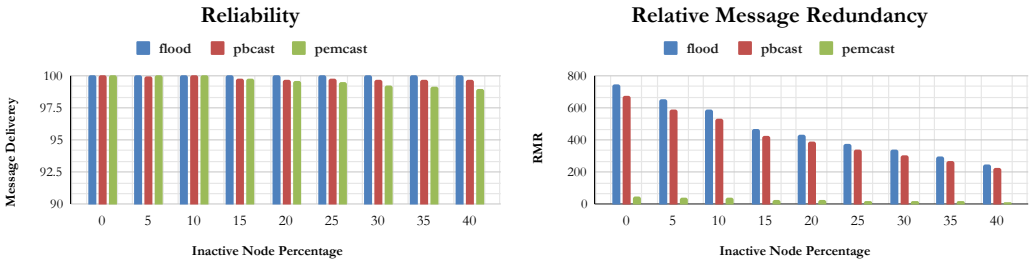
Message Cycling

The *pemcast* algorithm uses pseudo-random *nonce* numbers to uniquely identify messages. Upon receiving a message, a node determines whether it has recently seen a message by consulting its cache. In case of a cache hit, either the message is looping, or it travelled

over multiple paths and is therefore not on the fastest path. In either scenario, the node will drop the message. During path confirmation, cycling is not possible as the message travels on the reverse path. The *nonce* mechanism enables nodes to drop duplicate messages and prevents message cycles without running a control plane algorithm.

In this section, we provided a detailed description of *pemcast*. Next, we discuss the experimental evaluation setup and explore simulation results.

5.3 Experimental Evaluation



(a) Protocol Reliability. **(b)** Protocol Relative Message Redundancy.

Figure 5.2: Protocol performance comparison.

The goal of the experimental evaluation is to determine how well *pemcast* achieves the goals outlined in Section 5.2. In the remainder of this section, we evaluate *pemcast*.

5.3.1 Metrics

We used the following metrics to evaluate the performance of the algorithm:

- **Reliability** is the percentage of successful message deliveries. A value of 100% indicates that the protocol delivered every message sent. A flood-based algorithm will achieve a 100% delivery as long as at least one path is available to the destination. Note that we do not consider message loss due to network or transport layer failures.

- **Relative Message Redundancy** [31] (RMR) captures message overheads of a routing protocol. It is expressed as $(m/n-1)-1$, where *m* is the total number of messages sent and *n* is the number of nodes that participated in delivering the message. A zero value indicates that each node sent exactly one message. An increase in value indicates a

poorer utilisation of the network. Low RMR may indicate a failure to establish a path. Therefore, it should be followed by a high-reliability value.

- **Path Stretch** is a ratio between the length of the found path and the shortest path. A value of one indicates that an algorithm found the shortest path. Greater values indicate path stretch.

- **Network Explored** is the percentage of nodes that received a message. A zero value suggests that the message did not visit any nodes in the network (and therefore was not sent). A value of 100% indicates that all nodes in the network received the message.

5.3.2 Experimental Setup

Algorithms

There are many variations of gossip and flooding algorithms, too vast to benchmark against each. Instead, we compare the performance of *pemcast* to that of flood-based broadcast and probabilistic broadcast algorithms as outlined below.

We selected flood-based broadcast as it provides the highest reliability at the expense of efficiency. We implemented the algorithm as follows: After receiving a message, a node forwards it to all of its peers except the sender. The node ignores the message if it has processed it before. The algorithm terminates when the destination node receives the message.

Probabilistic broadcasting (*pbcast*) is a widely used mechanism to distribute messages in a peer-to-peer network. Thus it is vital to compare its performance to *pemcast*. The basic implementation of the probabilistic algorithm is as follows: Each node randomly selects a *fanout* number of peers and forwards the message to them. Nodes repeat this process until the destination node receives the message. There is a high variety [35] of optimisation and peer sampling strategies. We chose the minimalist implementation of the algorithm, which accurately represents the baseline behaviour of the algorithm.

Simulator

In order to evaluate *pemcast* we implemented a discrete event simulator[2]. The simulator triggers a message propagation event. The event is to deliver a message from a randomly selected, active source node to a randomly selected set of active destination nodes. Multiple destination nodes simulate communication to a group of trusted validator nodes. As of September 2021, the recommended trusted list of validators contained 41 nodes [78]. A single iteration of the simulator is complete when there are no more events to be processed. The simulator iterates until it reaches a 5% Relative Statistical Error in all the metrics outlined previously for a 95% confidence level. We performed the experiments using the XRP Network topology. The network contains 849 nodes and 8,136 edges and has a mean degree of 19.1, with an SD of 42.

Control Parameters

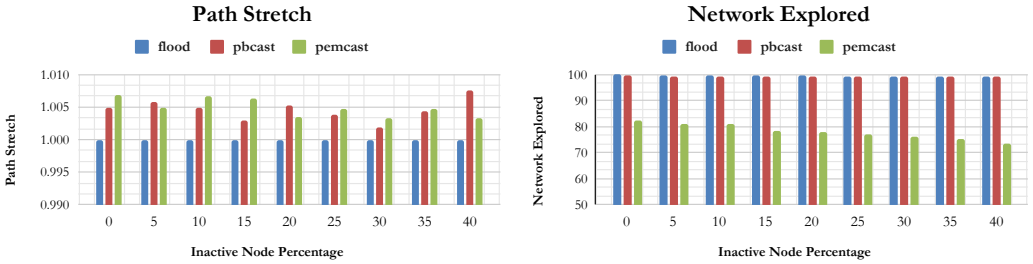
We use a range of different configuration parameters for the experiments. First, to accurately measure the effects of fanout between *pemcast* and *pbcast* we express it as a percentage of peers or neighbourhood edge nodes to which to send the message. The fanout ranges from 10% to 95%. Due to the network's density, we use a neighbourhood radius of two for *pemcast*. Finally, up to 40% of nodes in the network were randomly set as inactive.

Due to size limitations, we discuss only the fanout, which produced the highest reliability. 55% and 90% for *pemcast* and *pbcast* respectively. In the remainder of the section, we compare the performance results of the algorithms in the XRP network.

5.3.3 Reliability

We compare the reliability of all three algorithms in Figure 5.2a, together with relative message redundancy in Figure 5.2b. The *pemcast* algorithm achieves 100% reliability in a network with up to 10% of inactive nodes. When 40% of nodes are inactive, reliability drops to 99%. *pemcast* achieves this reliability whilst maintaining an RMR between 50 and 12. We measured these numbers with a fanout of 55%. Higher fanout values showed no improvement in reliability.

[2] https://bitbucket.org/vytautastumas/pblearn/src/master



(a) Path Stretch Comparison. **(b)** Percentage of the Network Explored.

Figure 5.3: Protocol overhead comparison.

In contrast, *pbcast* achieves marginally better reliability, between 100% and 99.9% with a fanout of 90%. However, the RMR is significantly higher, between 671 and 224, slightly lower than the flood-based algorithm. Smaller fanout value results in smaller RMR. For example, a fanout of 55% results in RMR between 409 and 132. However, the reliability drops down to between 99% and 93%.

As expected, the flood-based algorithm achieves 100% reliability with up to 40% of inactive nodes. However, the RMR is the highest: 742 in a fully active network and 250 when 40% of nodes are down.

Our proposed routing algorithm achieves a marginally (0.9%) lower reliability than *pbcast* and flooding. However, the amount of generated redundant traffic is significantly lower, *14 times lower than flooding and 13 times lower than pbcast*. The experimental evaluation shows that we have successfully achieved the goals of reducing redundant traffic and maintaining acceptable reliability even when 40% of nodes in the network are down. Next, we compare the overheads of the algorithms.

5.3.4 Overheads

All algorithms come with overheads, *pemcast* is no exception. Figure 5.3a depicts the stretch ratio between the established path and the shortest path. We calculate the shortest path using Dijkstra's Algorithm, considering only active nodes. The flood-based algorithm has little impact on the path length. It guarantees to find the shortest path as the algorithm finds all routes to the destination.

The impact of both *pbcast* and *pemcast* is insignificant. The *pemcast* algorithm achieves this by performing neighbourhood routing. It enables a node to route a message directly to

the destination when it is present in the node's neighbourhood. Due to the dense nature of the XRP network, the neighbourhoods are large, with around 500 nodes. Therefore, the probability of sending a message directly to the destination is high.

It is well established [31] that there is a clear trade-off between reliability and network exploration. Intuitively, in the presence of inactive nodes, as we explore more of the network, the higher the probability of finding a path to the destination. We show this in Figure 5.3b. The figure depicts the percentage of the network explored to achieve the reliability rates discussed in Section 5.3.3. Messages sent by the flooding algorithm reach 100% of nodes in the network. This behaviour is expected. Each node forwards the message to each of its peers (except the sender). Therefore, the message reaches every node in the network. To achieve high-reliability *pbcast* has to use a high fanout value. As a result, messages reach 99% of nodes. In contrast, messages distributed with *pemcast* reach between 82% and 73% of nodes. By multicasting messages to the edge of the neighbourhood, the algorithm can see a higher percentage of the nodes in the network without directly sending messages to them. The experimental results show that we also achieved the second goal of visiting a minimal portion of the network.

5.4 Conclusions

With the advent of blockchain technology, probabilistic broadcasting has been attracting renewed research interest to improve communication efficiency. In this chapter, we proposed a novel probabilistic, multicast, *application layer* routing algorithm called *pemcast*. It uses local topology awareness to increase routing efficiency whilst maintaining high reliability. Simulation results show that, compared to flooding and probabilistic broadcasting, *pemcast*: (1) can establish a path whilst sending fewer messages, (2) provides established paths which are closer to the shortest path, in comparison to probabilistic broadcasting, (3) explores a smaller portion of the network. Indeed, *pemcast* requires one order of magnitude fewer messages than the typical broadcasting techniques. A critical piece of our future work is to evaluate the performance of *pemcast* as the communication protocol for the XRP network.

Part IV

Protocol Layer

6 | A Ripple for Change

Analysis of Frontrunning in the XRP Ledger

"Arklius išvogus rakina tvartą."

The stable is locked after the horses are stolen.

Lithuanian Proverb

Blockchains are disrupting traditional finance by reducing the number of intermediaries and providing transparency. Blockchains, however, come with their own set of prominent issues. One such challenge is frontrunning. Attackers try to influence the transaction order so that their transaction executes before their victims' transaction. While frontrunning is a well-studied topic on Ethereum, it is unknown whether other blockchains are also susceptible to such attacks.

One proposed defence strategy against frontrunning attacks is to randomize the transaction execution order. XRP Ledger is the highest-value blockchain to use such a strategy. Furthermore, it runs a Decentralized Exchange, which provides ample frontrunning opportunities. Therefore, in the context of XRP Ledger, we examine whether randomized transaction order provides sufficient protection against frontrunning.

Our results show that the mechanism embedded in the XRP Ledger protocol is insufficient to prevent these attacks. We showcase two strategies to perform frontrunning attacks. The first, "naive" strategy, uses randomly generated accounts, whereas the second uses carefully selected accounts to improve the attack's success. Based on our analysis of the XRP Ledgers' historical data, we estimate that attackers could generate up to approx. 1.4M USD profit over two months, provided they succeeded to frontrun every opportunity.

6.1 Introduction

Blockchain technology is disrupting traditional finance with the rise of decentralized finance (DeFi). A core part of the DeFi ecosystem are so-called Decentralized Exchanges (or "DEXes"). A DEX is a peer-to-peer marketplace where users can exchange assets in a non-custodial way. In contrast to centralized exchanges, DEXes offer transparency and remove intermediaries such as banks, brokers, payment processors, or other institutions, thereby reducing costs. However, DEXes have a severe drawback: pending blockchain transactions are public. Adversaries can *frontrun* pending orders by observing them and placing their own competing orders.

For example, on Ethereum [34], the transaction execution order is based on the transaction fee, from high to low. Therefore, adversaries can influence where their transactions occur in a block by changing the transaction fee. As a result, many users of Ethereum DEXes become victims of these attacks. Frontrunning is a well-studied issue on Ethereum[68, 92, 88]. However, to our knowledge, these attacks are yet to be examined on other blockchains.

XRP Ledger users were victims to frontrunning in the past [36]. Similarly to other blockchains, the pending transactions on XRP Ledger are public. Adversaries could generate low transaction IDs to ensure their adversarial orders are applied first. As a response, the XRP Ledger developers introduced a new transaction ordering strategy to make the transaction order difficult to predict [101]:

> *"The order transactions execute within a ledger is designed to be unpredictable, to discourage frontrunning."*

The new transaction ordering strategy, similar to suggestions in previous works [110], creates a pseudo-random shuffle of the to-be-executed transactions. As a result, an attacker using a single account has a 50% probability of a successful attack.

A naive attacker may use multiple randomly generated accounts to increase their likelihood of success. However, we show that when using multiple accounts, the winning probability does not increase as expected.

Therefore, we propose a more advanced strategy that leverages carefully crafted attacker accounts to increase the effectiveness of frontrunning attacks. Using historical mainnet data,

we simulate how often a frontrunning attack would have been successful on the XRP Ledger and measure the profitability of mounting such an attack.

Contributions. We summarize the contributions presented in this paper as follows:

1. We present an advanced attack strategy that leverages selected accounts, which weakens the resistance of the XRP Ledger to frontrunning attacks.

2. We evaluate two frontrunning attack strategies on the XRP Ledger. We show that with *five* accounts, an attacker would have a success rate of 80% and 96% when using random accounts and carefully crafted ones, respectively.

3. We analyze historical mainnet data and estimate an upper bound of approx. 1.4M USD profit which attackers could have made when frontrunning every opportunity on the XRP Ledger over the two months.

We organize the remainder of this chapter as follows. In Section 6.2, we detail frontrunning attacks and the XRP Ledger. In Section 6.3, we describe the attack strategies and evaluate them in Section 6.4. In Section 6.5, we discuss our results, and in Section 6.6, we highlight related work. Finally, in Section 6.7, we conclude the chapter.

6.2 Background

In traditional markets, frontrunning is the use of insider information on a future deal that is about to happen in order to place a competing order right before to reap benefits at its expense. It is a well-studied issue in centralized exchanges [5]. Recently, frontrunning also started to take place on decentralized exchanges (DEXes). Given the public nature of the transactional data, these attacks are common and pose a serious concern across multiple DEXes [68, 92, 88].

A frontrunning attack is successful only when the attacker's transactions execute before the victim's transaction. For example, on Ethereum, miners process transactions based on their fee, starting from the highest. Therefore, an attacker can ensure their transaction executes before the victim by paying a greater transaction fee than the victim.

In contrast, XRP Ledger uses a pseudo-random shuffling algorithm to determine the execution order. Therefore an attacker cannot guarantee that their transaction will occur

before the victim's transaction. However, as shown later in Section 6.3, an attacker can employ other mechanisms to increase their chance of victory. In the remainder of this section we provide the background on XRP Ledger and discuss its transaction ordering mechanism.

6.2.1 Decentralized Exchange

A decentralized exchange (DEX) is a peer-to-peer marketplace where users can exchange assets in a non-custodial way, often leveraging blockchain technology. As opposed to centralized exchanges, DEXes offer transparency and remove intermediaries such as banks, brokers, payment processors, or other institutions, thereby reducing costs. DEXes are considered a cornerstone of decentralized finance (DeFi). The two most common types of DEXes are automated market makers (AMMs) and order books.

Automated Market Makers AMMs are the most widely adopted type of DEX. They enable instant liquidity and allow anyone to create their own market for any token. However, traders can swap their tokens only via a particular liquidity pool, where an algorithm automatically determines the price, usually leveraging the proportion of tokens in the pool. The fact that the price is computed automatically enables instant access to liquidity in markets that otherwise may have low liquidity. Yet, a negative aspect of AMMs is their dependency on liquidity providers (i.e., other users depositing enough liquidity into each pool).

Order Book Order books do not depend on liquidity pools and, therefore, not on users providing liquidity. Moreover, traders themselves determine the price instead of an algorithm. This allows for more profitable trades. In order book-based DEXes, traders typically place their buy and sell orders in realtime. Then, an internal algorithm is responsible for efficiently matching buy and sell orders. However, as opposed to AMMs, traders have to wait for their orders to be matched, which depending on the current supply and demand, might take a long time to settle. Moreover, order books require high transaction throughput, which most blockchain technologies cannot provide. One blockchain technology that does provide efficient order book-based trading is the XRP Ledger[1].

XRPL runs an order book-based DEX, in which users can buy and sell tokens for XRP and other tokens. Tokens represent any type of asset other than XRP. Trade orders are

[1] https://xrpl.org

called "offers", which represent limit orders to buy a specific amount of one token (or XRP) in exchange for a specific amount of another. Offers are placed via *OfferCreate* transactions. The network executes an offer only if there are matching offers for the same currency pair. Offers are consumed by starting with the best exchange rate first. An offer can be filled fully or partially. If it is not filled right away, the offer becomes an object in the ledger for the remaining amount. Later, other offers can match and consume it. Due to this, offers can execute at a better than their requested exchange rate, or at exactly their stated exchange rate later on. An offer can be cancelled manually with a *OfferCancel* transaction. A user can also configure the offer to automatically expire in the future. Trade offers are executed only on ledger execution, every 3 to 5 seconds. Therefore, XRPL is not suitable for high-frequency trading. Furthermore, the random transaction ordering discourages frontrunning.

6.2.2 Entities

For the reader's convenience we highlight the properties of transactions and accounts relevant to this work. For further details, we refer the reader to Chapter 2.

Transactions A Transaction is the only way to update the state of the XRP Ledger. A unique *hash* identifies each transaction. In addition, a transaction has a *sequence* number and the *address* of the sending account. A transaction is valid only if its *sequence* number is exactly *1* greater than the previous transaction from the same account. The sequence number ensures that transactions submitted by the same account execute in the order of submission.

Accounts An account in XRPL corresponds to a holder of the XRP native currency and a sender of transactions. A unique *address* and a *sequence* number identify each account. The former is a 20-byte address in Base58 format derived from the account's public key. The latter is a 32-bit unsigned integer that ensures that transactions are executed only once and in the correct order.

Transaction Order

Consensus requires that validators execute accepted transactions in the same order to reach the same result. XRP Ledger uses a shuffle algorithm to create a deterministic, pseudo-

randomly ordered list of transactions [99] in which transactions of a single account occur one after another in their submission order. XRP Ledger nodes process transactions one at a time. If a transaction fails, it still appears in the ledger, but with a *tec*-class result code, at the end of the canonical list. Once all transactions are processed, failed transactions are retried. However, they remain at the end of the canonical list.

Order Analysis

The address of an account is 160 bits long, whereas the hash of the Merkle Tree root is 256 bits. During the shuffling, the address of an account occupies 256 bits, but the first 96 most significant bits are all zeros. Therefore, the first 96 bits of each *account key* are equal, and only the remaining 160 bits effectively determine the transaction order. In the remainder of our work, we assume the starting position is the 160th bit.

Further, we assume that the account addresses and the salts come from a uniform distribution (i.e. each value has an equal probability of occurring). Thus, each bit can have a value of 0 or 1, with an equal likelihood.

Let A and V denote the attacker's and victim's account addresses and S the salt. The attacker wins when

$$A \oplus S < V \oplus S \tag{6.1}$$

Let A_i denote the value of the i-*th* most significant bit, $A_i \in [0, 1]$. We can say that Rule 6.1 is true, when

$$A_i \oplus S_i < V_i \oplus S_i \tag{6.2}$$

We can further deduce that Rule 6.2 is true if and only if $A_i \neq V_i$, and $A_i \oplus S_i = 0$, and $V_i \oplus S_i = 1$. We know that, $A_i \oplus S_i = 0 \iff A_i = S_i$, and $V_i \oplus S_i = 1 \iff V_i \neq S_i$.

With the information above, we can express the probability of A winning as

$$\begin{aligned} P(A_{win}) &= P(A_i = S_i \wedge V_i \neq S_i) \wedge P(A_i \neq V_i) \\ &= \frac{1}{4} + \left(\frac{1}{4} \cdot \frac{1}{2^1}\right) + \left(\frac{1}{4} \cdot \frac{1}{2^2}\right) + ... + \left(\frac{1}{4} \cdot \frac{1}{2^n}\right) \\ &= \frac{1}{2} \sum_{n=1}^{160} \frac{1}{2}^n = \frac{1}{2} - \frac{1}{2}^{160} \approx 0.5 \end{aligned} \tag{6.3}$$

Let's assume the attacker uses two accounts, A and B, to increase their odds of winning. There will be pairs of salt and victim address, for which A and B will both win, and pairs for which both will lose. The probability of winning with either A or B is

$$P(A_{win}) \vee P(B_{win}) = P(A_{win}) + P(A_{lose} \wedge P(B_{win})) \tag{6.4}$$

Therefore we need to estimate the probability of B winning when A loses. We illustrate this scenario in Figure 6.1. The event of B winning and A losing can occur only when $A_i \neq B_i$ and A loses, and B, V reach a draw at the ith position. Therefore B can only win with a probability of $\frac{1}{4}$ from the $i+1^{th}$ round. This occurs as a result of B drawing i times and winning from the $i+1$ turn onwards, meaning

$$P(A_{lose} \wedge B_{win}) = \frac{1}{2}\sum_{n=1}^{160-i}\frac{1}{2^i}\frac{1}{2^n} = \frac{1}{2^{i+1}}\left(1 - \frac{1}{2^{160-i}}\right) \tag{6.5}$$

For small values of i, the probability of losing with A and winning with B is $\approx \frac{1}{2}^{i+1}$. Thus, the probability of winning with either A or B, is

$$\begin{aligned} P(A_{win}) \vee P(B_{win}) &= \frac{1}{2} - \frac{1}{2^{160}} + \frac{1}{2^{i+1}}\left(1 - \frac{1}{2^{160-i}}\right) \\ &\approx \frac{1}{2} + \frac{1}{2^{i+1}} \end{aligned} \tag{6.6}$$

We can say that the probability of winning with two accounts, with different MSB at position i, for low values of i is $\frac{1}{2} + \frac{1}{2}^{i+1}$. For example, let's assume the first three MSBs of A and B are 000 and 001, respectively. The first difference in the MSB happens at position 3, $i = 3$. The combined probability of victory with these two accounts would be $\frac{1}{2} + \frac{1}{2}^4 = 0.5625$. The XRP Ledger shuffling mechanism ensures that an attacker using a single account has only 50% chance of winning. Furthermore, as we have shown, even when using multiple, randomly generated accounts, XRP Ledger provides some defense against frontrunning attacks. In the following section, we will discuss our proposed frontrunning model and showcase a strategy which bypasses the XRP ledger resilience to frontrunning.

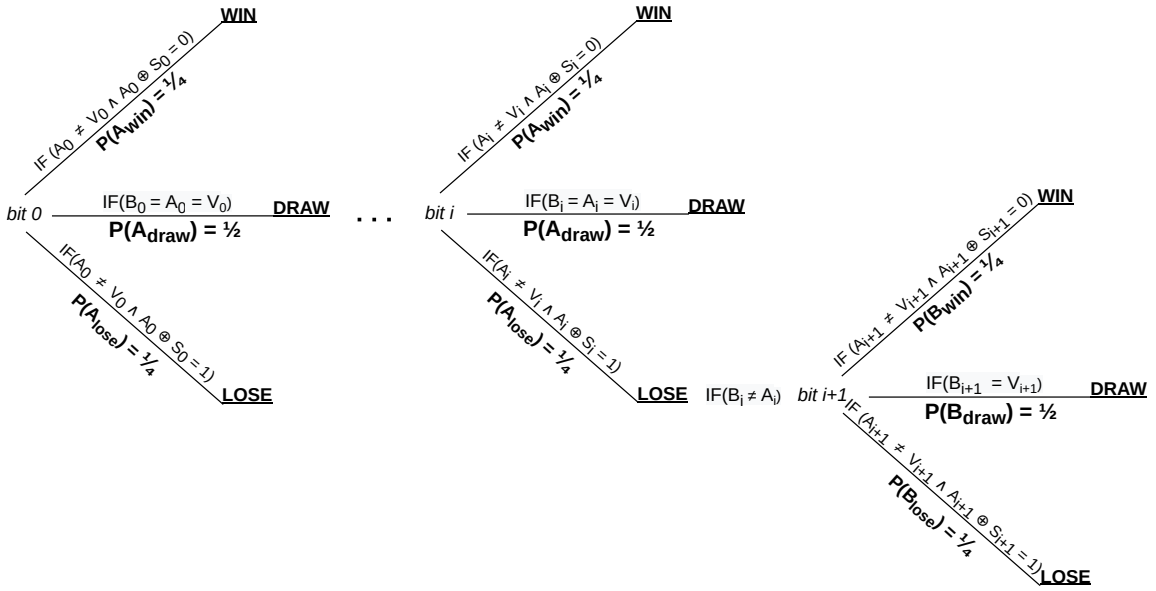


Figure 6.1: Illustrative example of XOR ordering as a sequential series of bit comparisons.

6.3 Methodology

In this section, we describe the attack model and our frontrunning strategy.

6.3.1 Attacker Model

We begin with an example of an opportunity. Imagine Alice is selling 2 *FOO* for 1 *XRP* and Bob is selling 5 *FOO* for 5 *XRP*, at an exchange rate of 0.5 *XRP/FOO* and 1 *XRP/FOO*, respectively. A victim submits an offer to buy 7 *FOO* for 7 *XRP* at an exchange rate of 1 *XRP/FOO*. This is an opportunity for the attacker to buy Alice's offer before the victim and sell it back to the victim at a higher price. Therefore, the attacker submits two offers: (i) a buy offer to take Alice's offer and (ii) an order to sell 2 *FOO* for 2 *XRP*. If the attacker's buy offer occurs before the victim's, the attacker wins. The victim will take the attacker's and Bob's offers, resulting in the attacker making a profit of 1 *XRP* minus the transaction fee.

Our attacker is required to monitor opportunities (pending offer transactions) in real time. We define an opportunity as follows. Assume an available sell offer O_{sell} for a token T at a price P, and an incoming buy offer O_{buy} at a price P'. We define the case when $P' > P$ as an opportunity to make a profit $\delta P = P' - P$, also called the spread.

During some preliminary analysis, we found 65,291 such opportunities. For example,

in the ledger 74,037,904, the transaction **2F6CAF5C87F...** created a buy order at a 13.3, thousand times higher than the market average price.

6.3.2 Frontrunning Strategy

Identifying Opportunities

The attacker requires a stream of pending offers and existing sell orders on the ledger to detect an opportunity. They can get this information via the XRP Ledger WebSocket API [104]. The API allows anyone to subscribe to a stream of pending transactions. The same API also provides book order information. The arbitrage system *JACK* [86] successfully used the WebSocket API to detect arbitrage opportunities in real time. Therefore, we use the same API to detect frontrunning opportunities.

As discussed in Section 6.2, candidate transactions propagate through the network via broadcasting. We can reduce the delay between transaction submission and detection by running a *rippled* server connected to many peers. Such a server will quickly learn about new transactions submitted anywhere in the network.

Frontrunning Attack

The attacker controls N accounts $A = \{a_1, a_2, a_N\}$, with which it submits the frontrunning transactions. The attacker monitors incoming proposed offer transactions for an offer O_{V-buy}, which fully consumes the cheapest existing O_{sell} offer for some token T. The attacker ignores the opportunity if the spread between the victim's and the seller's offers is below a profit threshold. Otherwise, the attacker prepares two transactions for each of the N accounts: a buy offer O_{A-buy} to consume O_{sell}, and a sell offer O_{A-sell} with the price equal to O_{V-buy}. The attacker cancels the O_{A-buy} offer if the account does not win. If the accounts do not hold token T, then the O_{A-sell} transaction will fail. However, the attacker will incur a loss of $2 \times$ transaction fees. In case the attacker's buy offer occurs *before* the victim's buy offer, the attack will be successful. The attacker makes a profit of $P' - P - 2 \times$ transaction fees.

Improving the Odds

A single account has a 50% probability of winning. An attacker can use multiple accounts to increase their odds. However, as discussed in Section 6.2.2, randomly generated accounts

do not provide sufficient guarantees. In the following, we outline a strategy to improve the winning odds.

For the attacker's account A to occur before the victim's account V, the following has to occur, $A \oplus S < V \oplus S$.

The outcome of the XOR between the most significant bits of the salt and the account address will determine if the result ends up in the lower half of the range of possible accounts or in the upper half.

Therefore in the event that $V_0 \oplus S_0 = 1$ where they represent the first MSB of the victim and the salt respectively, we want our attacker to have $A_0 \oplus S_0 = 0$, such that we win the ordering 50% of the time. With the same reasoning, in the event that $V_0 \oplus S_0 = 0$, we need to focus on the second MSB. Therefore if $V_1 \oplus S_1 = 1$, then we need to pick an attacker such that: $A_0 \oplus S_0 = 1 \wedge A_1 \oplus S_1 = 0$.

We continue this technique for the first n most significant bits, as described in Algorithm 5.

Algorithm 5 Selecting attackers from pool

```
1: procedure SELECTATTACKERS(A_victim, N, P)
2:     V ← V ≪ (256 − N)
3:     A = []
4:     for i in range(M) do
5:         Index = V ⊕ 2^(N+1−i)
6:         A.append(P[Index])
7:     return P
```

A limitation of this approach is that the attacker has to maintain a pool of accounts. Each account with a unique combination of the first n bits, for a total of 2^n accounts.

The pool has to be initialised ahead of conducting the frontrunning attacks. Therefore, our proposed strategy can result in a high initial capital requirement, given that each account requires a reserve of XRP (see Section 6.3.2). An attacker may leverage partial order fulfilment if they do not have enough capital. This would allow the attacker to "bootstrap" the initial funds. Although, they might not be able to take full advantage of all opportunities.

The Price of Victory

There are three costs associated with each frontrunning attack.

Fees. The transaction fee is a small amount of XRP paid when submitting a transaction. It is non-recoverable and is the only source of loss for the attacker. Transaction fees scale exponentially, as outlined in Section 6.2.2. Therefore, depending on the potential profit from a transaction, an attacker may pay larger fees than expected to ensure their transactions appear in the next ledger.

Reserve. The attack has associated reserve requirements. The *Base Reserve* is the minimum XRP required for each account in the ledger. These funds are frozen but are released when the attacker deletes the account. At the time of writing, the base reserve was 10 XRP for an account [103]. The *Owner Reserve* is a requirement of 2 XRP for each object that an account owns within the ledger. The relevant objects for the frontrunning attacks are offers and trustlines. Each attacker's account has to reserve $10 + 2 * offers + 2 * trustlines$.

Liquidity. Liquidity represents the funds that are necessary to consume an offer. It forms a significant portion of the attacker's capital. Every attacker's account has to contain enough liquidity to place the buy orders. XRP Ledger DEX supports partial order fulfilment. Therefore, the attacker may perform frontrunning with fewer funds. Though, this will reduce the profit, making some attacks unprofitable.

6.4 Evaluation

In this section, we evaluate frontrunning on the XRP Ledger testnet and mainnet.

6.4.1 Frontrunning on the Testnet

Owner	Type	Offer	Exchange Rate
Seller	Sell	1 TOK : 1 XRP	1
Seller	Sell	2 TOK : 3 XRP	1.5
Seller	Sell	3 TOK : 6 XRP	2
Victim	Buy	6 TOK : 12 XRP	2
Attacker	Buy	1 TOK : 1XRP	1
Attacker	Buy	2 TOK : 3 XRP	1.5
Attacker	Sell	3 TOK : 6 XRP	2

Table 6.1: Testnet transaction summary.

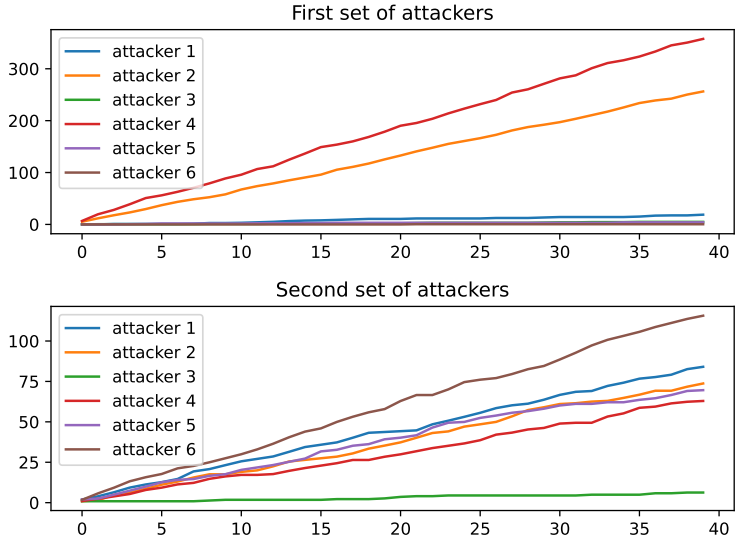


Figure 6.2: Testnet cumulative profit growth using two different sets of attackers.

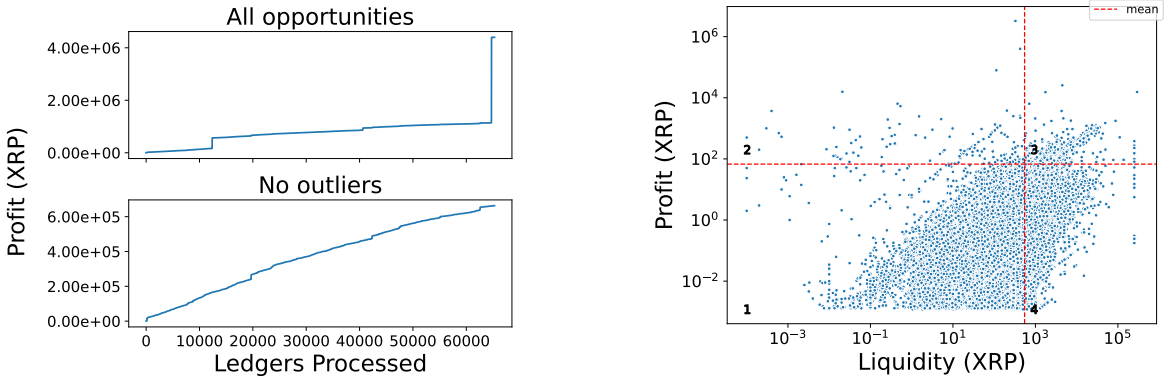
The aim of conducting the frontrunning attack on the testnet is to show that the attack is possible without causing any harm. The *testnet* is the testing ground for the XRP Ledger. Users can experiment with the XRP Ledger without risking real money. To not interfere with other users of the testnet, we issued our token: *TOK*. We summarise the transactions submitted for each experiment in Table 6.1.

Setup Our experimental setup is as follows. First, we create an artificial opportunity for the attacker to make 2 XRP profit. Then, we attack with six randomly generated accounts. After the attack, we clear the offers and restore *TOK* balances. We experiment with two sets of accounts, performing 3,000 attacks with each, and capture the XRP balance of each account every 50 iterations.

Results The evaluation on the testnet shows positive results (see Figure 6.2). The frontrunning attack is feasible and profitable. In both experiments, all accounts made a profit but exhibited different profit growths.

From the first set of accounts (see Figure 6.2 top), accounts 2 and 4 were the most successful, with a profit of 250 and 350 XRP, respectively. In contrast, the combined earnings of the other accounts were around 33 XRP.

From the second set of accounts (see Figure 6.2 bottom), account 6 was the most prosperous, with a final balance of 115 XRP. The least profitable account, 3, made only 6 XRP.



(a) Cumulative profit growth on the mainnet.

(b) Illustrative example of profit and liquidity requirements for opportunities.

Figure 6.3: Frontrunning profitability on the XRP Ledger.

The others profited between 62 and 73 XRP. These results demonstrate that randomly selected accounts do not win a proportional number of times.

It is important to note that, during the evaluation, our transactions overwhelmingly filled each ledger, and we attacked the same victim with a few randomly generated accounts. This may have affected the transaction order. To ensure the success of the attack was not the effect of the testnet and further validate the attack, we evaluate frontrunning on the mainnet.

6.4.2 Frontrunning on the Mainnet

Opportunities in the Wild

We scanned the XRP Ledger for opportunities as defined in Section 6.3. Between June 30th and August 30th 2022, we found 65,291 profitable transactions and evidence of frontrunning.

Total Profit The potential profit in two months was 4,402,433.5815 (4.4M) XRP or an estimated 1.4M USD. The most profitable token was *SOLO*. Opportunities involving this token were worth 3.4M XRP or 1.0M USD. An order of magnitude more valuable than opportunities with the 10th most profitable *USD* token. These opportunities had a profit potential of 9.5K USD.

A significant portion of the profit came from a single order to buy *SOLO* at 13.3 thousand times higher price than the market average (see Figure 6.3a) in ledger *74,037,904*. The second most profitable transaction was *xSPECTAR* in ledger *72,890,193*, 1.9 thousand times higher than the market average. It is hard to say whether these transactions were attempts

at market manipulation or human errors. For example, we found one transaction in which a user bought *BLA* token at an exchange rate of 0.45 when the market average exchange rate was 0.045. Without the outlying transactions, the profit growth is linear (see Figure 6.3a bottom).

Profit and Investment The profitability of each opportunity O is the difference between the revenue, required liquidity for one account, and transaction fees. Liquidity is the amount of XRP needed for one attacker's account to frontrun one opportunity.

We illustrate the ratio between profitability and liquidity in Figure 6.3b. Both axes are on a logarithmic scale due to the high variability between the data points. We divide the figure into four quadrants by the average profit and liquidity. Based on profitability and capital requirements, it is clear that opportunities are not equal.

The 4th quadrant contains the least desirable opportunities. These require exceedingly high liquidity and provide small returns. The worst offender required a liquidity of $2,000$XRP and returned only 0.002XRP profit. We conclude that it is impractical to frontrun these opportunities.

Members of the 3rd quadrant require more careful consideration. Some of them require a liquidity greater than $87,000$ XRP. These liquidity requirements make the majority of these opportunities infeasible to be frontrun. However, an attacker may use partial order fulfilment to reap partial profits.

The opportunities in the 1st quadrant are of low reward, but they also need equally little liquidity. These opportunities might be worth attacking. However, increasing the number of accounts will further diminish their profitability. For the same reason, these opportunities should be fully consumed and not rely on partial order fulfilment.

The 2nd quadrant provides the best return on liquidity. In this quadrant, we find all the profit outliers. The attacker should use the highest number of accounts to maximise the chances of winning.

Evidence of Frontrunning

During our analysed period, we also found an attacker performing frontrunning attacks. For example, in ledger 73,698,802, the buy offer of account **r3Vh9ZmQ...** was frontrun by account **robp8v3o...**. The attacker was active for three weeks. On September 10th, they

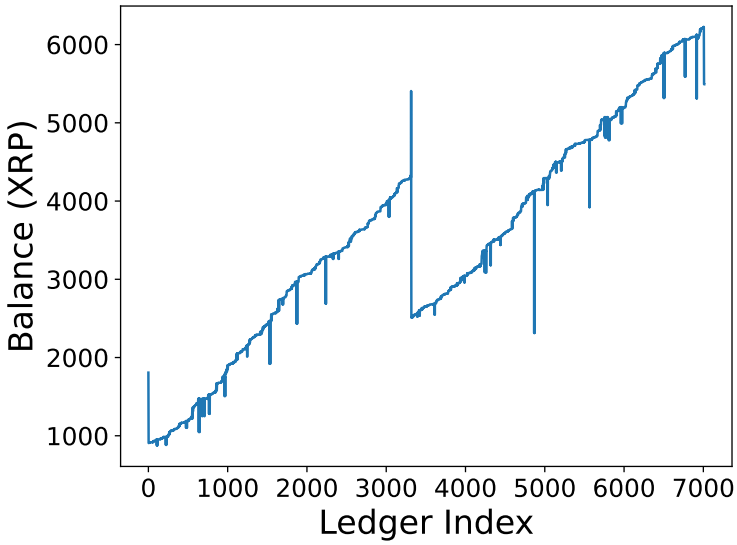


Figure 6.4: Frontrunner balance growth.

deleted their accounts. During this time, the attacker performed 4,075 attacks, out of which 2,435 were successful, and 1,640 were not. In Figure 6.4, we illustrate the balance growth of the attacker's account. We estimate that the attacker made a total profit of around 8,000 XRP or approx. 3,200$. The account started with an initial balance of around 1,000 XRP. Halfway, they sent their profits to another account, as indicated by the sharp balance drop.

Optimizing the Attack

We have started the process of responsible disclosure of our finding to the XRP Ledger developers. Therefore, we evaluate our attack strategy with a simulator to not interfere with their investigation.

The period of two months, during which we found the opportunities and discovered the frontrunner, provide the ideal data to evaluate our attack strategy. In the remainder of this section we discuss our results.

Simulator The simulator scans ledgers for offers O_{buy} that consumed offers O_{sell}, such that the asking price of O_{sell} was lower than O_{buy}. The simulator injects frontrunning transactions using up to M accounts and orders the transactions following the rules discussed in Section 6.2.2.

We validated the correctness of our implementation of the shuffling algorithm using

100,000 salts sampled from ledgers between the 10th and 14th of August 2022. For every sample, we were able to recreate the exact transaction order.

As outlined in Section 6.2.2, the ordering mechanism requires a random salt, which is the hash of the root of a Merkle Tree. In addition to the sampled seeds, we generated 150,000 random 256-bit seeds. We used these values to simulate the transaction ordering. The simulator then captures the number of times in which at least one attacker's transaction occurred before the victim. In addition, the simulator calculates the potential profit, assuming the win is guaranteed.

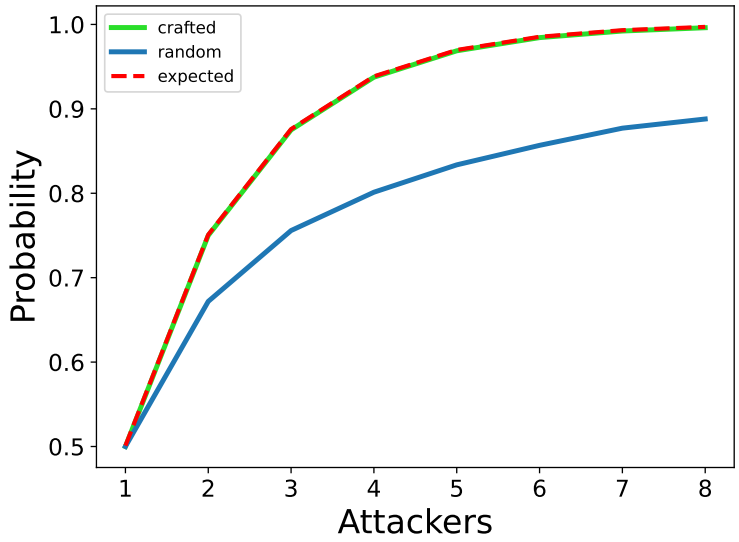


Figure 6.5: Winning probabilities per frontrunning strategy.

Data We created two pools of accounts using different generation strategies. The first pool contains randomly generated accounts created with the XRP Ledger Python library [121]. We paid no attention to the bit sequence of these accounts. Furthermore, during the attack, the attacker picks accounts from this pool in an arbitrary order.

The second pool contains accounts that cover every combination of the first eight most significant bits. The attacker picks from this pool in a way that maximises the odds of winning. Both pools contained 256 entries. We simulate both strategies on a set of 2,000 victims randomly picked from the opportunities we found. For each victim, we select up to 8 frontrunning accounts.

Results We illustrate the winning probabilities of these strategies in Figure 6.5. The red dashed line indicates theoretical probability derived via $\frac{2^M-1}{2^M}$, where M is the number of accounts used for the attack. The blue line indicates the winning probability with the naive strategy, and the green uses the advanced strategy.

Both approaches achieve a winning probability of 50% when attacking with a single account. However, we already see a divergence between the strategies of approx. 8% when using two accounts. The gap increases up to 11% as we increase their number. The approaches slowly converge when raising the count beyond 8 (not shown). However, even with 30 accounts, the naive approach does not reach a 96% success probability.

Randomly selected accounts perform poorly due to overlaps within the first 8 MSB patterns. An arbitrarily added account will provide some benefit. However, in most cases, another existing account would have complemented the victim's ID. Therefore, unless selected carefully, a new account adds little benefit.

6.5 Discussion

Order Fairness Achieving transaction order fairness in blockchains is a prevalent topic. Several works have demonstrated that order fairness based on transaction submission time is impossible [69, 71]. Instead, works rely on weaker fairness definitions. For instance, the work by Kelkar et al. [69] demonstrates batch order fairness, which states that if majority of honest participants receive t_a before t_b, then t_a must be in an older, or the same block as t_b. However, this does not prevent in-block reshuffling. On the other hand, authors of [71] propose applying orders linearly, such that, if all honest participants receive t_a before t_b, then t_a must be applied before t_b. However, delays in message propagation, still allow attackers to send their transactions before all nodes received t_a, and thus apply the malicious transactions before t_a. In contrast, XRP Ledger offers account level fairness. If all honest nodes receive t_a before t_b, and both transactions were submitted by the same account, t_a will execute before t_b, where We assume that the execution of t_a will be successful first. However, XRP Ledger does not provide any guarantees for transaction execution order for different accounts.

Strengths The XRP Ledger shuffling mechanism guarantees that the order in which transactions execute is difficult to predict. An attacker has a 50% probability of a successful attack with a single account. However, even when using multiple randomly generated accounts, the odds of winning do not grow as expected. We demonstrate that due to overlapping most significant bits of the account addresses, there will be cases when accounts share their wins or loses. Therefore, the shuffling algorithm offers some security against naive frontrunning, although it does not provide guarantees.

Limitations We demonstrate that an attacker can generate accounts with alternating bit sequences to maximise their odds of winning. Furthermore, the new ordering mechanism introduced by the XRP Ledger displays some properties that might deter its use in traditional shuffle requirements. For example, when shuffling two "neighbour accounts" (accounts only differing in the last bit), they will end up together in the result but in reverse order. While this type of preserved locality might be an issue in some shuffling scenarios, the XRP Ledger uses it to shuffle 256-bit hashes that, by design, are near impossible to reverse. Therefore, it is unrealistic to take advantage of this property. For example, if an attacker, in a realistic time frame, could generate an account that is one bit off from the victim, the attacker could also create the keys to the victim's account and steal their funds directly.

The developers of the XRP Ledger are about to extend the DEX with an AMM [106]. AMMs are known to be vulnerable to frontrunning attacks on Ethereum. The developers claim that their AMM would resist frontrunning attacks due to Canonical transaction ordering. However, the two strategies proposed in this work are not exclusive to Order Book-based DEX. They can be equally applied to AMM exchanges to generate specific accounts. However, the attacker would have to perform backrunning attacks (i.e. the transactions would have to occur after the victims' transactions). To ensure its transactions occur at the end of the ledger, the attacker can exploit the feature of retriable transactions. Failed transactions move to the end of the canonical set, where they are processed after all other transactions. Therefore, the attacker would have to craft transactions that successfully execute only on a second try. This approach, combined with the methods we showed in this paper, would allow the attacker to perform sandwich attacks, needed to frontrun AMM-based DEXes.

We have disclosed our findings to the XRP Ledger developers at Ripple. We are working together with the team to improve the robustness of XRP Ledger's blockchain to frontrunning

attacks. We leave the evaluation of any future mitigation strategy to future work.

6.6 Related Work

Frontrunning has been thoroughly studied on the Ethereum blockchain [55, 68, 92, 88]. Eskandari et al. [55] were the first to provide a taxonomy of frontrunning attacks. Daian et al. [68] were the first to provide evidence of frontrunning practices by monitoring Ethereum's pool of pending transactions. Zhou et al. [92] made an effort to provide a theoretical upper bound regarding the maximum profit attackers could make. Torres et al. [88] were the first to provide a lower bound on the actual profit made by attackers by measuring the prevalence of different frontrunning attacks across Ethereum's transaction history. Our work is the first to study frontrunning on a different blockchain than Ethereum. In particular, we demonstrate the practicality of frontrunning attacks on the XRP Ledger and provide the first lower bound regarding the maximum profit attackers could make on the XRP Ledger by applying our attacks.

Besides analyzing the prevalence of frontrunning, several countermeasures have been proposed to mitigate the effects of frontrunning on Ethereum.

A prevalent strategy is to prevent frontrunning attacks at the application layer. For example, by using commit-and-reveal scheme [98], splitting large trades into smaller trades with tighter slippage protection [105], or designing new decentralized exchanges that are more frontrunning-resistant [80, 53, 100, 108, 91, 115, 111]. However, these solutions often introduce higher transaction costs and do not protect against every type of frontrunning attack. Another solution is to prevent frontrunning at the consensus layer by either ordering transactions in a fair manner [51, 69, 71, 82] or making transactions private [70, 89, 112]. However, these techniques are not widely adopted as they are not applicable to dynamic public blockchains, where users frequently join and leave the network. As an alternative to modifying the application layer or the consensus layer, projects based on private transaction pools such as Flashbots [84] and Eden [87] have been proposed. These pools establish private agreements with miners, which allow users to bypass the public pool of pending transactions and privately submit their transactions. However, Weintraub et al. [118] and others [110, 98] discovered that private pools are excessively being used to perform frontrunning attacks on Ethereum and that the profit distribution within Flashbots is heavily skewed towards miners.

Despite having thoroughly analyzed the liveness and safety properties of the XRP Ledger's consensus protocol [73, 47, 65, 67] including its topology [132], little effort has been made to analyze whether its transaction ordering strategy is resistant to frontrunning attacks. Peduzzi et al. [86] demonstrated that it is possible to observe transactions in realtime on the XRP Ledger and to perform arbitrage on its decentralized exchange. However, they did not study whether frontrunning attacks would be feasible. Our work shows that frontrunning is possible and that the countermeasures employed by the XRP Ledger to prevent frontrunning are not effective enough. We show that due to the low transaction fees on the XRP Ledger, sharp-witted attackers will be able to perform frontrunning attacks and make a significant profit.

6.7 Conclusion

Frontrunning is a well-studied and understood issue on Ethereum. However, little effort has been made to analyze whether other blockchains are potentially vulnerable to these attacks. In this chapter, we conduct the first security analysis of the XRP Ledger blockchain with respect to frontrunning attacks. We show that the XRP Ledger is vulnerable to frontrunning despite its efforts to make frontrunning more difficult by enforcing a canonical transaction order. We demonstrate that these attacks are profitable. Furthermore, we showcase two attack strategies. In the first strategy, accounts are chosen randomly. In the second, accounts are selected carefully to maximize the success probability of frontrunning the victim. We first evaluate the feasibility of our strategies on the XRP Ledger testnet by creating and successfully frontrunning our own generated opportunities. We then show the profitability of the two attack strategies on the historic mainnet data. We discover, over two months, frontrunning opportunities worth 1.4M USD, including evidence of a user already performing frontrunning attacks in the wild.

Part V

Final Remarks

7 | Conclusions

"Aukštai kilęs, žemai pulsi."

The higher you climb, the further you fall.

Lithuanian Proverb

Within the chapters of this thesis, we have explored three layers of the XRP Ledger: *Infrastructure Layer*, *Network Layer*, and last but not least the *Protocol Layer*. At each of these layers, we found opportunities for further improvement. In this chapter, we summarize the answers to the research questions posed in Chapter 1.

7.1 Research Question 1

"What are the structural properties of the XRP Ledger Peer-to-Peer Network?"

Summary A decentralized peer-to-peer overlay network forms the backbone of the XRP Ledger. We found that XRP Ledger topology is significantly smaller than other blockchain networks. The nodes are connected via short paths and are tightly clustered. Furthermore, the clusters tend to have a hub-and-spoke structure, as shown by the high assortativity of the network. Unlike other blockchain overlay networks, XRP Ledger has a small-world topology. However, it does share some similarities with other blockchains. For example, similar to other blockchains, the topology of XRP Ledger is not random. The network degree distribution has an exponential-like shape. Though, because the distribution does not follow a power-law, the topology is not scale-free.

We revealed a vast disparity between nodes that accept incoming connections and nodes that do not. Link analysis revealed that approx. 90% of nodes consume more connec-

tions than they consume. Conversely, only 10% of nodes are truly altruistic. Furthermore, 15% of all nodes are extremely selfish, and do not accept incoming connections. These nodes increase the dependence on the influential nodes. Thus, contributing to network centralization. We suspect that a lack of financial incentives contributes to this behaviour, as there are significant costs associated with running a reliable node. Natural centralization is a common problem in decentralized peer-to-peer networks[30][23]. A common solution is to introduce communal incentives or mandatory behaviour.

The stability of the XRP Ledger Consensus Protocol is heavily dependent on the robustness of the underlying peer-to-peer network. The topology of the XRP Ledger is highly disassortative, with small-degree nodes tending to connect to high-degree nodes. This results in a cascading failure scenario where the failure of high-degree nodes leads to the progressive disconnection of small-degree nodes from the network. Our experiments have shown that the failure of approximately 9% of high-degree nodes will halt the consensus process. While the introduction of Negative UNLs results in a marginal improvement in robustness of 2%, this is insufficient as it does not address the underlying disassortative structure of the XRP Ledger.

An effective strategy to improve the robustness of the topology is to replace some of the existing connections to high-degree nodes with those to low-degree nodes. By maintaining a 1:1 ratio of connections to low and high degree nodes, we improve the *quorum robustness* by approx. 36%.

7.2 Research Question 2

"Can the routing efficiency of the XRP Ledger be further improved?"

Summary To answer this question we proposed *pemcast* [138], a probabilistic edge multicast routing algorithm. The algorithm uses local topology awareness to increase the routing efficiency without sacrificing reliability. Our simulation results show that, compared to flooding and probabilistic broadcasting, *pemcast*: (1) establishes a path whilst sending fewer messages, (2) finds shorter paths than probabilistic broadcasting, (3) explores a smaller portion of the network. Indeed, *pemcast* requires one order of magnitude fewer messages

than the typical broadcasting techniques. Therefore, it is possible to improve the routing efficiency of XRP Ledger. However, our propose solution requires significant changes to the routing algorithm. Therefore, we believe that further improvements are possible.

7.3 Research Question 3

"Does pseudo-random transaction execution order provide sufficient protection against frontrunning attacks?"

Summary XRP Ledger offers account-level fairness. If all honest nodes receive t_a before t_b, and both transactions were submitted by the same account, t_a will execute before t_b. We assume that the execution of t_a will be successful first. However, XRP Ledger does not provide any guarantees for transaction execution orders for different accounts. The XRP Ledger shuffling mechanism guarantees that the order in which transactions execute is difficult to predict. An attacker has a 50% probability of a successful attack with a single account. Furthermore, even when using multiple randomly generated accounts, the odds of winning do not grow as expected. We demonstrate that due to overlapping most significant bits of the account addresses, there will be cases when accounts share their wins or loses. Therefore, the shuffling algorithm offers some security against naive frontrunning, although it does not provide guarantees. We demonstrate that an attacker can generate accounts with alternating bit sequences to maximise their odds of winning.

A coin toss odds would make most attacks unprofitable on other blockchains such as Ethereum, where transaction fees are significantly higher than on XRP Ledger. However, the low transaction cost on the XRP Ledger enables attacks to use multiple accounts to frontrun a victim. Therefore, on the XRP Ledger specifically, pseudo-random transaction execution does not provide sufficient protection against frontrunning.

7.4 Future Directions

We explored three layers of the XRP Ledger. In each, we found opportunities to harden the security of the blockchain. However, we only scratched the surface. XRP Ledger does

not receive enough academic interest. In the following, we present possible directions to continue our work.

Extended Frontrunning Analysis In Chapter 6, we analyzed a type of frontrunning attack on the XRP Ledger. It is unclear whether this problem is impactful enough to require immediate attention. However, any solutions to this problem would require careful consideration and invasive updates to the transaction processing mechanism. Therefore, the first part of the future work is to determine, together with the XRP Ledger Community, whether frontrunning has enough impact to warrant a solution.

In theory, more complex frontrunning attacks, such as *Insertion*, also called *Sandwich Attacks*, are possible in XRP Ledger. In an insertion attack, the attacker, having observed a profitable transaction T_V from victim V, submits two transactions T_{A-buy} and T_{A-sell}, in a manner such that T_{A-buy} executes before T_V, and T_{A-sell} executes after T_V. One example of a sandwich attack is an arbitrage on a decentralized exchange. An attacker observes a large trade, called a whale, and sends a buy transaction before the trade and a buy transaction after it. In essence, "sandwiching" the victim's transaction. These attacks primarily target Automated Market Maker (AMM) decentralized exchanges, in which the execution of the whale trade would drive the price up, thus allowing the attacker to profit.

Although, as discussed in Chapter 2, transactions from the same account that appear in the same ledger execute one after another, it is possible to break this order. We know that some transactions, upon initial execution failure, are placed at the end of the canonical order and are executed once other transactions are processed. Therefore, an attacker could conduct the sandwich attack by first submitting the T_{A-sell} transaction, which will fail initially if the account does not hold the particular token. However, once the T_{A-buy} executes successfully, the T_{A-sell} will successfully execute after a retry.

XRP Ledger community is working towards an AMM-based decentralized exchange [106], which may introduce new attack vectors. Therefore, our work can be continued to explore whether other types of frontrunning attacks are feasible on the XRP Ledger.

Predicting Transaction Order Multiple works attempt to predict which transactions will appear in a Bitcoin block [44, 50, 60] with varying degrees of success. In XRP Ledger, the random seed used to derive the Canonical Order is computed from the transactions

confirmed by the consensus protocol. Therefore, if one could predict which transactions validators accept, one could determine the exact transaction order. This would allow the user to influence the transaction order or submit their frontrunning transactions when success is guaranteed. Therefore, the final continuation of work on frontrunning would be to examine whether it is possible to predict and influence the transaction order on the XRP Ledger, given the short transaction confirmation delay.

Scaling the XRP Ledger *Payment Channels* [129], significantly improve the payment throughput of XRP Ledger. However, studies have shown that most transactions on XRP Ledger are not payments but trades on the decentralized exchange [74]. Therefore, these transactions do not benefit from the throughput improvement of payment channels.

XRP Ledger supports up to 1,500 transactions per second [126]. This claim originates from a performance benchmark conducted within a localized synthetic network [131]. In Chapter 3, we demonstrated various XRP Ledger topology properties. However, the benchmark does not reflect these properties. Furthermore, recently, David Schwartz, one of the original developers of XRP Ledger, confirmed [125]:

> @JoelKatz (1 pm, April 14, 2023):
> ...
> 3. We've never seen 1,500 TPS on the live XRPL. I suspect it could, as currently configured, probably sustain about 300-500.
> ...

Multiple variables affect the transaction throughput. For example, network topology dictates how often a message will be re-broadcasted until it reaches all nodes in the system. Network bandwidth affects the time it takes to transfer a message between nodes. The implementation of the blockchain affects the efficiency of servers processing the message. Thus, it is not entirely clear where the current bottlenecks of XRP Ledger are. Therefore, our final proposal for continuing our work is to create a more realistic benchmark of XRP Ledger transaction throughput. With this information, future authors may be able to identify new performance bottlenecks and thus further improve the XRP Ledger.

Bibliography

[1] Ithiel de Sola Pool and Manfred Kochen. “Contacts and influence”. In: *Social networks* 1.1 (1978), pp. 5–51.

[2] Leslie Lamport, Robert Shostak, and Marshall Pease. “The Byzantine Generals Problem”. In: *ACM Trans. Program. Lang. Syst.* 4.3 (July 1982), pp. 382–401. ISSN: 0164-0925. DOI: 10.1145/357172.357176. URL: https://doi.org/10.1145/357172.357176.

[3] Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. “Impossibility of distributed consensus with one faulty process”. In: *ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*. 1983.

[4] Alan Demers, Dan Greene, Carl Hauser, Wes Irish, John Larson, Scott Shenker, Howard Sturgis, Dan Swinehart, and Doug Terry. “Epidemic Algorithms for Replicated Database Maintenance”. In: *6th ACM Symposium on Principles of Distributed Computing*. PODC ’87. 1987. DOI: 10.1145/41840.41841.

[5] Jerry W Markham. “Front-running-insider trading under the commodity exchange act”. In: *Cath. UL Rev.* 38 (1988), p. 69.

[6] Stanley Wasserman and Katherine Faust. “Social Network Analysis: Methods and Applications”. In: *Structural analysis in the social sciences*. 1994.

[7] Kenneth P. Birman, Mark Hayden, Öznur Özkasap, Zhen Xiao, Mihai Budiu, and Yaron Minsky. “Bimodal multicast”. In: *ACM Trans. Comput. Syst.* 17 (1999), pp. 41–88.

[8] Réka Albert, Hawoong Jeong, and A L Barabasi. “Error and attack tolerance of complex networks”. In: *Nature* 406 (2000), pp. 378–382.

[9] Duncan S. Callaway, Mark E. J. Newman, Steven H. Strogatz, and Duncan J. Watts. “Network robustness and fragility: percolation on random graphs.” In: *Physical review letters* 85 25 (2000), pp. 5468–71.

[10] Reuven Cohen, Keren Erez, Daniel Ben-Avraham, and Shlomo Havlin. “Resilience of the Internet to random breakdowns”. In: (2000). URL: http://helix.nature.com/webmatters/tomog/tomog.html,.

[11] C. Diot, B.N. Levine, B. Lyles, H. Kassem, and D. Balensiefen. “Deployment issues for the IP multicast service and architecture”. In: *IEEE Network* 14.1 (2000), pp. 78–88. DOI: 10.1109/65.819174.

[12] Reuven Cohen, Keren Erez, Daniel Ben-Avraham, and Shlomo Havlin. “Breakdown of the Internet under intentional attack”. In: (2001).

[13] Patrick Th. Eugster, Rachid Guerraoui, Sidath B. Handurukande, Petr Kuznetsov, and Anne-Marie Kermarrec. “Lightweight probabilistic broadcast”. In: *2001 International Conference on Dependable Systems and Networks* (2001), pp. 443–452.

[14] A. Ganesh, A. Kermarrec, and L. Massoulié. “SCAMP: Peer-to-Peer Lightweight Membership Service for Large-Scale Group Communication”. In: *Networked Group Communication*. 2001.

[15] Jon M. Kleinberg and Steve Lawrence. “The Structure of the Web”. In: *Science* 294 (2001), pp. 1849–1850.

[16] Konstantin Klemm and Víctor M. Eguíluz. “Growing scale-free networks with small-world behavior.” In: *Physical review. E, Statistical, nonlinear, and soft matter physics* 65 5 Pt 2 (2001), p. 057102.

[17] Antony Rowstron and Peter Druschel. “Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems”. In: *Middleware 2001*. Ed. by Rachid Guerraoui. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 329–350. ISBN: 978-3-540-45518-9.

[18] M. Castro, P. Druschel, A.-M. Kermarrec, and A.I.T. Rowstron. “Scribe: a large-scale and decentralized application-level multicast infrastructure”. In: *IEEE Journal on Selected Areas in Communications* 20.8 (2002), pp. 1489–1499.

[19] John R. Douceur. “The Sybil Attack”. In: *International Workshop on Peer-to-Peer Systems*. 2002.

[20] D. Magoni and J.-J. Pansiot. “Evaluation of Internet topology generators by power law and distance indicators”. In: *10th IEEE International Conference on Networks*. 2002, pp. 401–406. DOI: 10.1109/ICON.2002.1033345.

[21] M E J Newman. “Mixing patterns in networks”. In: (2003).

[22] Alexei Vázquez and Yamir Moreno. “Resilience to damage of graphs with degree correlations”. In: (2003).

[23] Daniel Hughes, Geoff Coulson, and James Walkerdine. “Free riding on Gnutella revisited: the bell tolls?” In: *IEEE distributed systems online* 6.6 (2005).

[24] Spyros Voulgaris, Daniela Gavidia, and Maarten van Steen. “CYCLON: Inexpensive Membership Management for Unstructured P2P Overlays”. In: *J. Network Syst. Manage.* 13 (June 2005), pp. 197–217. DOI: 10.1007/s10922-005-4441-x.

[25] Vern Paxson. “End-to-end routing behavior in the internet”. In: *Comput. Commun. Rev.* 36 (2006), pp. 41–56.

[26] Ivan Damgård, Yvo Desmedt, Matthias Fitzi, and Jesper Buus Nielsen. “Secure Protocols with Asymmetric Trust”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. 2007.

[27] Joao Carlos Antunes Leitao, J. Pereira, and Luís E. T. Rodrigues. “HyParView: A Membership Protocol for Reliable Gossip-Based Broadcast”. In: *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07)* (2007), pp. 419–429.

[28] Mark D. Humphries and Kevin Gurney. “Network 'small-world-ness': A quantitative method for determining canonical network equivalence”. In: *PLoS ONE* 3.4 (Apr. 2008). DOI: 10.1371/JOURNAL.PONE.0002051.

[29] Satoshi Nakamoto. “Bitcoin: A peer-to-peer electronic cash system”. In: *Decentralized Business Review* (2008), p. 21260.

[30] Murat Karakaya, Ibrahim Korpeoglu, and Özgür Ulusoy. “Free riding in peer-to-peer networks”. In: *IEEE Internet computing* 13.2 (2009), pp. 92–98.

[31] João Leitão, José Pereira, and LuÍs Rodrigues. “Gossip-Based Broadcast”. In: 2010, pp. 831–860. DOI: 10.1007/978-0-387-09751-0_29.

[32] Jun Wu, Mauricio Barahona, Yuejin Tan, and Hongzhong Deng. “Robustness of random graphs based on graph spectra.” In: *Chaos* 22 4 (2012), p. 043101.

[33] David Schwartz, Noah Youngs, Arthur Britto, et al. “The ripple protocol consensus algorithm”. In: *Ripple Labs Inc White Paper* 5.8 (2014), p. 151.

[34] Gavin Wood et al. “Ethereum: A secure decentralised generalised transaction ledger”. In: *Ethereum project yellow paper* 151.2014 (2014), pp. 1–32.

[35] Daniel Gutiérrez-Reina, S. L. T. Marín, P. Johnson, and F. Barrero. “A survey on probabilistic broadcast schemes for wireless ad hoc networks”. In: *Ad Hoc Networks* 25 (2015), pp. 263–292.

[36] Donovan Hide. *Exploiting Ripple Transaction Ordering For Fun And Profit*. 2015. URL: http://availableimagination.com/exploiting-ripple-transaction-ordering-for-fun-and-profit/.

[37] Andrew Miller, James Litton, Andrew Pachulski, Neal Gupta, Dave Levin, Neil Spring, and Bobby Bhattacharjee. “Discovering bitcoins public topology and influential nodes”. In: *et al* (2015).

[38] Albert-László Barabási and Márton Pósfai. *Network science*. Cambridge: Cambridge University Press, 2016. URL: http://barabasi.com/networksciencebook/.

[39] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, et al. “On scaling decentralized blockchains”. In: *International conference on financial cryptography and data security*. Springer. 2016, pp. 106–125.

[40] Arthur Gervais, Ghassan O. Karame, K. Wüst, Vasileios Glykantzis, Hubert Ritzdorf, and Srdjan Capkun. “On the Security and Performance of Proof of Work Blockchains”. In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016).

[41] Till Neudecker, Philipp Andelfinger, and Hannes Hartenstein. “Timing Analysis for Inferring the Topology of the Bitcoin Peer-to-Peer Network”. In: *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld)* (2016), pp. 358–367.

[42] Guan-Sheng Peng, Suo-Yi Tan, Jun Wu, and Petter Holme. “Trade-offs between robustness and small-world effect in complex networks OPEN”. In: (2016). DOI: 10.1038/srep37317.

[43] Stefanie Roos, Martin Beck, and Thorsten Strufe. “Anonymous addresses for efficient and resilient routing in F2F overlays”. In: *35th IEEE International Conference on Computer Communications*. 2016, pp. 1–9. DOI: 10.1109/INFOCOM.2016.7524553.

[44] Beltran Fiz, Stefan Hommes, and Radu State. “Confirmation Delay Prediction of Transactions in the Bitcoin Network”. In: *Computer Science and its Applications / Computer Science and Ubiquitous Computing*. 2017.

[45] Giulio Malavolta, Pedro Moreno, Aniket Kate, and Matteo Maffei. “SilentWhispers: Enforcing Security and Privacy in Decentralized Credit Networks”. In: Jan. 2017. DOI: 10.14722/ndss.2017.23448.

[46] Stefanie Roos, Pedro A. Moreno-Sanchez, Aniket Kate, and Ian Goldberg. “Settling Payments Fast and Private: Efficient Decentralized Routing for Path-Based Transactions”. In: *ArXiv* abs/1709.05748 (2017).

[47] Brad Chase and Ethan MacBrough. *Analysis of the XRP Ledger Consensus Protocol*. 2018. arXiv: 1802.07242 [cs.DC].

[48] Sergi Delgado-Segura, Surya Bakshi, Cristina Pérez-Solà, James Litton, Andrew Pachulski, Andrew Miller, and Bobby Bhattacharjee. *TxProbe: Discovering Bitcoin’s Network Topology Using Orphan Transactions*. 2018. DOI: 10.48550/ARXIV.1812.00942. URL: https://arxiv.org/abs/1812.00942.

[49] Varun Deshpande, Hakim Badis, and Laurent George. “BTCmap: Mapping Bitcoin Peer-to-Peer Network Topology”. In: *2018 IFIP/IEEE International Conference on*

Performance Evaluation and Modeling in Wired and Wireless Networks (PEMWN) (2018), pp. 1–6.

[50] Beltran Borja Fiz Pontiveros, Robert Norvill, and Radu State. “Monitoring the transaction selection policy of Bitcoin mining pools”. In: *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium* (2018), pp. 1–6.

[51] Solana. *Proof of History Explained by a Water Clock*. [Online; accessed 07. oct. 2022]. June 2018. URL: https://medium.com/solana-labs/proof-of-history-explained-by-a-water-clock-e682183417b8.

[52] Nicole Balashov, Reuven Cohen, Avieli Haber, Michael Krivelevich, and Simi Haber. “Optimal shattering of complex networks”. In: (Dec. 2019). DOI: 10.1007/s41109-019-0205-5. URL: http://arxiv.org/abs/1912.04044%20http://dx.doi.org/10.1007/s41109-019-0205-5.

[53] Iddo Bentov, Yan Ji, Fan Zhang, Yunqi Li, Xueyuan Zhao, Lorenz Breidenbach, Philip Daian, and Ari Juels. “Tesseract: Real-Time Cryptocurrency Exchange Using Trusted Hardware”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (2019).

[54] Alexander Craig, Biswajit Nandy, and Ioannis Lambadaris. “Forwarding State Reduction for Multi-Tree Multicast in Software Defined Networks using Bloom Filters”. In: *IEEE International Conference on Communications*. 2019.

[55] Shayan Eskandari, Seyedehmahsa Moosavi, and Jeremy Clark. “SoK: Transparent Dishonesty: Front-Running Attacks on Blockchain”. In: *Financial Cryptography and Data Security - FC, St. Kitts and Nevis, February 18-22, 2019*. Springer, 2019, pp. 170–189.

[56] Yue Gao, Jinqiao Shi, Xuebin Wang, Qingfeng Tan, Can Zhao, and Zelin Yin. “Topology Measurement and Analysis on Ethereum P2P Network”. In: *Proceedings - IEEE Symposium on Computers and Communications* 2019-June (2019). DOI: 10.1109/ISCC47284.2019.8969695.

[57] Muhammad Anas Imtiaz, David Starobinski, Ari Trachtenberg, and Nabeel Younis. “Churn in the Bitcoin Network: Characterization and Impact”. In: *2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)* (2019), pp. 431–439.

[58] JoelKatz. “Suggestion: Reduce Relaying”. In: *Xrp Chat* (Oct. 2019). URL: https://www.xrpchat.com/topic/33075-suggestion-reduce-relaying.

[59] Alan Kaminsky. “Testing the randomness of cryptographic function mappings”. In: *Cryptology ePrint Archive* (2019).

[60] Kyungchan Ko, Taeyeol Jeong, Sajana Maharjan, Chaehyeon Lee, and James Won-Ki Hong. “Prediction of Bitcoin Transactions Included in the Next Block”. In: *International Conference on Blockchain and Trustworthy Systems*. 2019.

[61] Gleb Naumenko, Gregory Maxwell, Pieter Wuille, Alexandra Fedorova, and Ivan Beschastnikh. “Erlay: Efficient Transaction Relay for Bitcoin”. In: *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security* (2019).

[62] Elias Rohrer, Julian Malliaris, and Florian Tschorsch. “Discharged payment channels: Quantifying the lightning network’s resilience to topology-based attacks”. In: *IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*. 2019, pp. 347–356.

[63] Muhammad Saad, Victor Cook, Lana Nguyen, My T. Thai, and Aziz Mohaisen. “Partitioning Attacks on Bitcoin: Colliding Space, Time, and Logic”. In: *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)* (2019), pp. 1175–1187.

[64] Shijie Zhang and JongHyouk Lee. “Double-Spending With a Sybil Attack in the Bitcoin Decentralized Network”. In: *IEEE Transactions on Industrial Informatics* 15 (2019), pp. 5715–5722.

[65] Ignacio Amores-Sesar, Christian Cachin, and Jovana Micic. “Security Analysis of Ripple Consensus”. In: *24th International Conference on Principles of Distributed Systems, OPODIS 2020, December 14-16, 2020, Strasbourg, France (Virtual Conference)*. Ed. by Quentin Bramas, Rotem Oshman, and Paolo Romano. Vol. 184. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 10:1–10:16.

[66] Nicolae Berendea, Hugues Mercier, Emanuel Onica, and Etienne Rivière. “Fair and Efficient Gossip in Hyperledger Fabric”. In: *CoRR* abs/2004.07060 (2020). arXiv: 2004.07060. URL: https://arxiv.org/abs/2004.07060.

[67] Klitos Christodoulou, Elias Iosif, Antonios Inglezakis, and Marinos Themistocleous. “Consensus crash testing: exploring Ripples decentralization degree in adversarial environments”. In: *Future Internet* 12.3 (2020), p. 53.

[68] Philip Daian, Steven Goldfeder, Tyler Kell, Yunqi Li, Xueyuan Zhao, Iddo Bentov, Lorenz Breidenbach, and Ari Juels. “Flash Boys 2.0: Frontrunning in Decentralized Exchanges, Miner Extractable Value, and Consensus Instability”. In: *2020 IEEE Symposium on Security and Privacy, SP 2020, San Francisco, CA, USA, May 18-21, 2020*. IEEE, 2020, pp. 910–927.

[69] Mahimna Kelkar, Fan Zhang, Steven Goldfeder, and Ari Juels. “Order-Fairness for Byzantine Consensus”. In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 269.

[70] Eleftherios Kokoris-Kogias, Enis Ceyhun Alp, Linus Gasser, Philipp Jovanovic, Ewa Syta, and Bryan Ford. “CALYPSO: Private Data Management for Decentralized Ledgers”. In: *Proc. VLDB Endow.* 14.4 (2020), pp. 586–599.

[71] Klaus Kursawe. “Wendy, the Good Little Fairness Widget: Achieving Order Fairness for Blockchains”. In: *AFT ’20: 2nd ACM Conference on Advances in Financial Technologies, New York, NY, USA, October 21-23, 2020*. ACM, 2020, pp. 25–36.

[72] Seungjin Lee and Hyoungshick Kim. “On the robustness of Lightning Network in Bitcoin”. In: *Pervasive Mob. Comput.* 61 (2020), p. 101108.

[73] Lara Mauri, Stelvio Cimato, and Ernesto Damiani. “A Formal Approach for the Analysis of the XRP Ledger Consensus Protocol”. In: *Proceedings of the 6th International Conference on Information Systems Security and Privacy, ICISSP 2020, Valletta, Malta, February 25-27, 2020*. Ed. by Steven Furnell, Paolo Mori, Edgar R. Weippl, and Olivier Camp. SCITEPRESS, 2020, pp. 52–63.

[74] Daniel Perez, Jiahua Xu, and Benjamin Livshits. “Revisiting Transactional Statistics of High-scalability Blockchains”. In: *Proceedings of the ACM Internet Measurement Conference* (2020).

[75] Crystal Andrea Roma and M Anwar Hasan. “Energy consumption analysis of XRP validator”. In: *2020 IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*. IEEE. 2020, pp. 1–3.

[76] István András Seres, László Gulyás, Dániel A Nagy, and Péter Burcsi. “Topological analysis of bitcoins lightning network”. In: *Mathematical Research for Blockchain Economy*. Springer, 2020, pp. 1–12.

[77] Dimitris Vyzovitis, Yusef Napora, Dirk McCormick, David Dias, and Yiannis Psaras. *GossipSub: Attack-Resilient Message Propagation in the Filecoin and ETH2.0 Networks*. 2020. arXiv: 2007.02754 [cs.NI].

[78] *XRP Validators*. [Online; accessed 21. Sep. 2021]. Oct. 2020. URL: https://xrpcharts.ripple.com/%5C#/validators.

[79] Hideaki Aoyama. “XRP Network and Proposal of Flow Index”. In: *Proceedings of Blockchain in Kyoto 2021 (BCK21)*. 2021, p. 011003.

[80] Carsten Baum, Bernardo David, and Tore Kasper Frederiksen. “P2DEX: Privacy-Preserving Decentralized Cryptocurrency Exchange”. In: *Applied Cryptography and Network Security - 19th International Conference, ACNS*. 2021, pp. 163–194.

[81] Federico Franzoni, Xavier Salleras, and Vanesa Daza. “AToM: Active topology monitoring for the bitcoin peer-to-peer network”. In: *Peer-to-Peer Networking and Applications* 15 (2021), pp. 408–425.

[82] Mahimna Kelkar, Soubhik Deb, Sishan Long, Ari Juels, and Sreeram Kannan. “Themis: Fast, Strong Order-Fairness in Byzantine Consensus”. In: *IACR Cryptol. ePrint Arch.* 2021 (2021), p. 1465.

[83] Calvin Newport, Alex Weaver, and Chaodong Zheng. “Asynchronous Gossip in Smartphone Peer-to-Peer Networks”. In: *17th International Conference on Distributed Computing in Sensor Systems*. 2021.

[84] *Overview | Flashbots Docs*. [Online; accessed 23. Nov. 2021]. Nov. 2021. URL: https://docs.flashbots.net/flashbots-auction/overview.

[85] Aristodemos Paphitis, Nicolas Kourtellis, and Michael Sirivianos. “A First Look into the Structural Properties and Resilience of Blockchain Overlays”. In: *arXiv preprint arXiv:2104.03044* (2021).

[86] Gaspard Peduzzi, Jason James, and Jiahua Xu. “JACK THE RIPPLER: Arbitrage on the Decentralized Exchange of the XRP Ledger”. In: *2021 3rd Conference on Blockchain Research & Applications for Innovative Networks and Services (BRAINS)*. IEEE. 2021, pp. 1–2.

[87] Chris Piatt, Jeffrey Quesnelle, and Caleb Sheridan. “Eden Network”. In: (2021). URL: https://edennetwork.io/EDEN_Network___Whitepaper___2021_07.pdf.

[88] Christof Ferreira Torres, Ramiro Camino, and Radu State. “Frontrunner Jones and the Raiders of the Dark Forest: An Empirical Study of Frontrunning on the Ethereum Blockchain”. In: *30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021*. Ed. by Michael Bailey and Rachel Greenstadt. USENIX Association, 2021, pp. 1343–1359.

[89] David Yakira, Avi Asayag, Gad Cohen, Ido Grayevsky, Maya Leshkowitz, Ori Rottenstreich, and Ronen Tamari. “Helix: A Fair Blockchain Consensus Protocol Resistant to Ordering Manipulation”. In: *IEEE Trans. Netw. Serv. Manag.* 18.2 (2021), pp. 1584–1597.

[90] Lin Zhao, Sourav Sengupta, Arijit Khan, and Robby Luo. “Temporal Analysis of the Entire Ethereum Blockchain Network”. In: *Proceedings of the Web Conference 2021* (2021). DOI: 10.1145/3442381.3449916.

[91] Liyi Zhou, Kaihua Qin, and Arthur Gervais. “A2MM: Mitigating Frontrunning, Transaction Reordering and Consensus Instability in Decentralized Exchanges”. In: *CoRR* abs/2106.07371 (2021). arXiv: 2106.07371. URL: https://arxiv.org/abs/2106.07371.

[92] Liyi Zhou, Kaihua Qin, Christof Ferreira Torres, Duc Viet Le, and Arthur Gervais. “High-Frequency Trading on Decentralized On-Chain Exchanges”. In: *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*. IEEE, 2021, pp. 428–445.

[93] [Online; accessed 9. Nov. 2022]. Mar. 2022. URL: https://nakamotoinstitute.org/static/docs/b-money.txt.

[94] *A Beginner's Guide to Understanding the Layers of Blockchain Technology - Blockchain Council*. [Online; accessed 28. Mar. 2023]. May 2022. URL: https://www.blockchain-council.org/blockchain/layers-of-blockchain-technology.

[95] *Bit Gold | Satoshi Nakamoto Institute*. [Online; accessed 9. Nov. 2022]. Nov. 2022. URL: https://nakamotoinstitute.org/bit-gold.

[96] *Bitcoin Hashrate Chart - BTC Hashrate 243.56 EH/s*. [Online; accessed 15. Nov. 2022]. Nov. 2022. URL: https://www.coinwarz.com/mining/bitcoin/hashrate-chart.

[97] *Bitcoin price today, BTC to USD live, marketcap and chart | CoinMarketCap*. [Online; accessed 9. Nov. 2022]. Nov. 2022. URL: https://coinmarketcap.com/currencies/bitcoin.

[98] Agostino Capponi, Ruizhe Jia, and Ye Wang. "The Evolution of Blockchain: from Lit to Dark". In: *arXiv preprint arXiv:2202.05779* (2022).

[99] Brad Chase and Ethan MacBrough. *XRP Github code for Transaction Set Canonical Ordering*. 2022. URL: https://github.com/XRPLF/rippled/blob/e32bc674aa2a035ea0f05fe43d2f301b203f1827/src/ripple/app/misc/CanonicalTXSet.cpp.

[100] Michele Ciampi, Muhammad Ishaq, Malik Magdon-Ismail, Rafail Ostrovsky, and Vassilis Zikas. "FairMM: A Fast and Frontrunning-Resistant Crypto Market-Maker". In: *Cyber Security, Cryptology, and Machine Learning - 6th International Symposium, CSCML 2022*. 2022, pp. 428–446.

[101] *Decentralized Exchange - Limitations*. [Online; accessed 11. Oct. 2022]. Oct. 2022. URL: https://xrpl.org/decentralized-exchange.html.

[102] *FITTER documentation fitter 1.4.0 documentation*. Apr. 2022. URL: https://fitter.readthedocs.io/en/latest.

[103] XRP Ledger Foundation. *Reserves | XRPL.org*. [Online; accessed 14. Sep. 2022]. Sept. 2022. URL: https://xrpl.org/reserves.html.

[104] XRP Ledger Foundation. *subscribe | XRPL.org*. [Online; accessed 16. Sep. 2022]. Sept. 2022. URL: https://xrpl.org/subscribe.html.

[105] Lioba Heimbach and Roger Wattenhofer. “Eliminating Sandwich Attacks with the Help of Game Theory”. In: *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security* (2022).

[106] Aanchal Malhotra. *0030 XLS 30d: Automated Market Maker on XRPL #78*. [Online; accessed 05. oct. 2022]. July 2022. URL: https://github.com/XRPLF/XRPL-Standards/discussions/78.

[107] *Maximum Number of Peers | XRPL.org*. [Online; accessed 9. Sep. 2022]. Sept. 2022. URL: https://xrpl.org/set-max-number-of-peers.html.

[108] Conor McMenamin, Vanesa Daza, and Matthias Fitzi. “FairTraDEX: A Decentralised Exchange Preventing Value Extraction”. In: *IACR Cryptol. ePrint Arch.* (2022), p. 155. URL: https://eprint.iacr.org/2022/155.

[109] *Peer Crawler | XRPL.org*. Jan. 2022. URL: https://xrpl.org/peer-crawler.html.

[110] Julien Piet, Jaiden Fairoze, and Nicholas Weaver. “Extracting Godl [sic] from the Salt Mines: Ethereum Miners Extracting Value”. en. In: *arXiv:2203.15930 [cs]* (Mar. 2022). arXiv: 2203.15930. URL: http://arxiv.org/abs/2203.15930.

[111] CoW Protocol. *CoW Protocol - Batch Auctions*. [Online; accessed 05. oct. 2022]. Oct. 2022. URL: https://docs.cow.fi/overview/batch-auctions.

[112] The Penumbra Protocol. *The Penumbra Protocol - ZSwap*. [Online; accessed 07. oct. 2022]. Oct. 2022. URL: https://protocol.penumbra.zone/main/zswap.html.

[113] *RPOW - Reusable Proofs of Work*. [Online; accessed 9. Nov. 2022]. Mar. 2022. URL: https://nakamotoinstitute.org/finney/rpow/index.html.

[114] *Running an XRP Ledger Validator*. Apr. 2022. URL: https://xrpl.org/blog/2020/running-an-xrp-ledger-validator.html.

[115] Emrah Sariboz, Gaurav Panwar, Roopa Vishwanathan, and Satyajayant Misra. “FIRST: FrontrunnIng Resilient Smart ConTracts”. In: *CoRR* abs/2204.00955 (2022). DOI: 10.48550/arXiv.2204.00955. arXiv: 2204.00955. URL: https://doi.org/10.48550/arXiv.2204.00955.

[116] *Today’s Top 100 Crypto Coins Prices And Data | CoinMarketCap*. [Online; accessed 15. Nov. 2022]. Nov. 2022. URL: https://coinmarketcap.com/coins.

[117] *UNL XRP Ledger Foundation*. [Online; accessed 5. Oct. 2022]. Oct. 2022. URL: https://foundation.xrpl.org/unl.

[118] Ben Weintraub, Christof Ferreira Torres, Cristina Nita-Rotaru, and Radu State. “A Flash(bot) in the Pan: Measuring Maximal Extractable Value in Private Pools”. In: *CoRR* abs/2206.04185 (2022).

[119] xpring-eng. *rippled-network-crawler*. Jan. 2022. URL: https://github.com/xpring-eng/rippled-network-crawler.

[120] *XRP Network Snapshots*. Apr. 2022. URL: https://drive.google.com/drive/folders/1SY4IemcQsr0FCiagLOdSyNlOYmQ6SLdg.

[121] *xrpl-py*. [Online; accessed 7. Oct. 2022]. Oct. 2022. URL: https://pypi.org/project/xrpl-py.

[122] *Accounts | XRPL.org*. [Online; accessed 7. Mar. 2023]. Mar. 2023. URL: https://xrpl.org/accounts.html.

[123] *Bitnodes*. [Online; accessed 4. Jan. 2023]. Jan. 2023. URL: https://bitnodes.io.

[124] *Consensus | XRPL.org*. [Online; accessed 24. Mar. 2023]. Mar. 2023. URL: https://xrpl.org/consensus.html.

[125] *David "JoelKatz" Schwartz on Twitter*. [Online; accessed 17. Apr. 2023]. Apr. 2023. URL: https://twitter.com/JoelKatz/status/1646649714417487872.

[126] *Home | XRPL.org*. [Online; accessed 3. Feb. 2023]. Feb. 2023. URL: https://xrpl.org.

[127] *Interledger Protocol (ILP): Open and Inclusive Payments*. [Online; accessed 3. Apr. 2023]. Jan. 2023. URL: https://interledger.org.

[128] Team Mudrex. “A Beginner’s Guide to the Layers of Blockchain Architecture”. In: *Mudrex Blog* (Feb. 2023). URL: https://mudrex.com/blog/layers-of-blockchain-explained.

[129] *Payment Channels | XRPL.org*. [Online; accessed 17. Apr. 2023]. Apr. 2023. URL: https://xrpl.org/payment-channels.html#payment-channels.

[130] *Proof-of-stake (PoS) | ethereum.org*. [Online; accessed 3. Feb. 2023]. Feb. 2023. URL: https://ethereum.org/en/developers/docs/consensus-mechanisms/pos.

[131] *Ripple: The Most (Demonstrably) Scalable Blockchain - High Scalability* -. [Online; accessed 17. Apr. 2023]. Apr. 2023. URL: http://highscalability.com/blog/2017/10/2/ripple-the-most-demonstrably-scalable-blockchain.html.

[132] Vytautas Tumas, Sean Rivera, Damien Magoni, and Radu State. "Topology Analysis of the XRP Network". In: *38th ACM SIGAPP Symposium on Applied Computing (SAC'23)*. 2023.

[133] Xrplf. *Squelch Peer Limit*. [Online; accessed 5. Jan. 2023]. Jan. 2023. URL: https://github.com/XRPLF/rippled/blob/develop/src/ripple/overlay/ReduceRelayCommon.h#L47.

[134] Anna D Broido and Aaron Clauset. "Scale-free networks are rare". In: (). DOI: 10.1038/s41467-019-08746-5. URL: https://doi.org/10.1038/s41467-019-08746-5.

[135] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, and Bobby Bhattacharjee. "Measurement and Analysis of Online Social Networks". In: *Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement*, pp. 29–42.

[136] Hani Salah, Stefanie Roos, and Thorsten Strufe. "Characterizing Graph-Theoretic Properties of a Large-Scale DHT: Measurements vs. Simulations". In: (). URL: https://www.p2p.tu-darmstadt.de/.

[137] *Visa Operational Performance Data, Q4 2020*. URL: https://s1.q4cdn.com/050606653/files/doc_financials/2020/q4/Visa-Inc.-Q4-2020-Operational-Performance-Data.pdf.

[138] Vytautas Tumas. *It came to me once in a dream.* 2023.