NLAC: A Self-Maintained Trust Overlay for the XRP Ledger

Flaviene Scheidt de Cristo*, Arno Geimer*, Radu State*
* University of Luxembourg, SnT, 29, Avenue J.F Kennedy, L-1855 Luxembourg
Email:{flaviene.scheidt, arno.geimer, radu.state}@uni.lu

***Abstract*—This paper presents NLAC, a framework for creating, managing, and maintaining trusted Unique Node Lists (UNLs) for the XRP Ledger. NLAC consists of three modules that automate the generation of UNLs, including membership management, classification, and grouping. The framework uses layered Token-Curated Registries (TCRs) and an optimization algorithm to ensure the dispersion of authority within the UNL space. NLAC improves the safety and liveness of the XRP Ledger by achieving maximum trust in the UNL space.**

***Index Terms*—XRPL, unstructured p2p, trust overlay, blockchain**

I. Introduction

Since the release of Bitcoin [1] several blockchains have been proposed in different shapes and flavors. Being one of the first, the XRP Ledger (XRPL) is today well established in this middle, with a market capital of $25B$[1]. Its unique consensus mechanism [2] also makes the XRPL a faster and less resource-intensive system, breaking away from the traditional *Proof of Work*. However, there is a discussion about how much *permissionless* and *distributed* the validation process actually is.

The XRP Ledger Consensus Protocol (XRP LCP) is considered a *Byzantine-fault Tolerant* type of consensus, using quorums of trusted subnets to validate ledger versions. In this context, a validator node must declare a *Unique Node List* (UNL), comprising a list of peers trusted to not collude to defraud the ledger. Although every node has the power to choose its own trusted list, in reality, some scenarios may cause the ledger to fork or stall [3] and a minimum overlap between two or more lists is necessary to ensure *safety* and *liveness* of the network. By safety, we mean that nothing bad will happen, while liveness ensures that something good will eventually happen [3].

In this work, we propose NLAC (*nested, layered, automatic, continuous*), a framework not only for automatically generating single trusted UNLs, but also for creating, managing and maintaining the entire UNL space. NLAC comprises three modules that manage membership, classification, and grouping of nodes into trusted lists without human intervention by using layered *Token-Curated Registries* (TCRs) and optimization to achieve maximum trust on the UNL space while guaranteeing better dispersion of authority.

[1]According to https://cryptoslate.com/coins/ on 4/april/2023

979-8-3503-2687-1/23/$31.00 ©2023 IEEE

II. Background

A. The XRP Ledger Consensus Protocol

In the XRP LCP, trust relies upon the so-called *Unique Nodes Lists*, encompassing a static set of validators believed not to collude to manipulate or break the consensus process. In this scenario, trust is said to be collective, which means that there is no individual trust in validators. Instead, we expect the entire UNL set to behave in a non-byzantine way [2]. To achieve this goal, each node declares its trust in a subnetwork of validators represented by a UNL, the agreement between these nodes being the source of truth on the ledger.

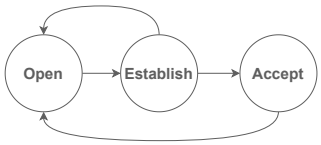

Fig. 1. Simplified State Machine for the XRP LCP with the three main phases and its transitions.

The XRP LCP works as a replicated state machine, as shown in Figure 1. In a simplified way, the consensus round starts with the *open* state, when every node opens a new version of the ledger locally. The next phase is called *establish* and comprises the voting process. First, the nodes aggregate all transactions received asynchronously during the previous round into a set called *proposal*. Proposals are broadcast through the network. Nodes receive these proposals, comparing their own sets solely with those received from their trusted peers. If there are disjoint sets - called *disputes* -, the nodes analyze whether the disputed transactions are present in the majority of the proposals received from their UNLs. In the affirmative case, the transaction is included in a new proposal. The process repeats until 80% of the UNL agrees on a set of transactions. Each node then applies the agreed set to the local open ledger in the next phase: *accept*. This threshold is called the *trust threshold*, representing the number of honest nodes required to keep the ledger safe while making progress.

B. The Trust Overlay

We call *trust overlay* the entire space of trust relationships between nodes established by the UNLs declared. Certain overlap conditions must be met between trusted sets to

guarantee the liveness, safety, and performance of the ledger. With a loosely connected trust overlay, validators would take longer to agree on a final ledger version due to clustering and the difficulty of transactions reaching distant nodes. A highly connected network, however, has the disadvantage of high message overhead, increasing latency, and, as a result, landing on the same problem of increasing the time needed to agree on closing a ledger or even increasing the possibility of a stall or fork [4].

Four primary works address the issue of the minimum intersection area between UNLs required to ensure safety and liveness. The first whitepaper released [2] presented the following formulation for the minimum overlap required ($w_{i,j}$) between two UNLs i and j:

$$|UNL_i \cap UNL_j| \geq w_{i,j} max(|UNL_i|, |UNL_j|) \, \forall i, j$$

This work concludes $w_{i,j} >= 0.2$, which means that the overlap should be greater than 20% of the size of the largest UNL, assuming $0.2 * n_{total}$ to be the minimum UNL size, being n_{total} the total number of nodes participating in the network. Later, an independent analysis by Armknecht et al. [5] showed that the 20% overlap may not be sufficient to reach consensus, increasing the minimum required overlap to 40%.

The current whitepaper [6] was released in 2018, stating that the intersection area between two UNLs should be superior to 90%, also pointing out a specific case where even with a 99% overlap, the ledger progress could stall. This work was the first to propose the use of a unique UNL. Another independent analysis by Christodoulou et al. [7] tackles the UNL overlapping problem using an empirical approach, showing that the minimum UNL overlap can be relaxed when there are less than 20% malicious nodes present. This work suggests space for optimizations on the minimum UNL overlap, also suggesting the necessity of a malicious node estimator to relax the overlap.

III. Problem Definition

Based on the studies presented, the XRP Ledger Foundation curates and maintains a main UNL, advising all participants to put their trust in this list [8]. Given these guidelines, as of February 2023, the trust overlay of the XRPL network comprises a core of 34 validators connected in a complete graph. This structure and governance method leads to some considerations about the dispersion of authority in the current implementation of the XRPL.

Vergne [9] presents a discussion about the concept of authority dispersion by breaking it into two definitions: *decentralization* and *distribution*. While *decentralization* is defined as the dispersion of coordinated communications, *distribution* is characterized by the diffusion of the decision-making process among stakeholders. Applying those definitions to the scenario proposed, decentralization tells us about how the network is structured and how the communication between nodes occurs, while distribution is about who has the power to make decisions.

The XRPL network is built upon an unstructured p2p system, duly considered *decentralized* since there is no central authority coordinating the way information is exchanged between participants. However, it does not have robust means to dynamically rearrange its trust overlay. The solution adopted in the event of a partial outage is the *Negative UNL* (nUNL) [10], which puts trusted validators that are believed to be malfunctioning in a list of nodes that will be ignored during the consensus process. The nUNL is designed to handle limited disruptions, and the addition and removal of nodes from the main UNL still require manual intervention. This notion opposes what is commonly agreed on in terms of decentralization, derived from the work of Baran [11], which points out the ability of a network to delegate routing decisions under adverse conditions. The trust overlay lacks the ability of self-arrangement, leaving the decision to reroute the trust in the hands of human actors.

There are two instances in which the XRPL network presents low degrees of distribution; first, on the trust overlay itself, with the core of 34 nodes that do not delegate the decision-making and listen only to themselves. This closed system allows non-UNL nodes solely to transmit transactions and to relay proposals and validations generated by trusted validators. The second instance is on the curation of the main UNL, relying on a central authority to decide who are the trustable nodes and how many of them are necessary to keep the network alive and safe while maintaining good performance.

We can then argue that the trust overlay of the XRPL has a low dispersion and could benefit from higher decentralization. In addition to the overlapping issue, these characteristics also arise from the lack of appropriate tools and techniques that allow the automatic and safe generation and maintenance of trusted validator lists. Such techniques would support the implementation of a model closer to what is conceptually described in the XRP LCP specifications [3], providing higher authority dispersion on the consensus process.

On top of that, nodes in the XRPL also do not have incentives to be part of the main UNL and behave correctly. Unlike other consensus mechanisms, such as Proof-of-Work and Proof-of-Stake [12], the XRPL does not issue rewards for validating ledgers. The only incentive a node has to become a validator is to help keep the network secure and the ledger progressing without forks. However, the requirements to become a trusted validator are expensive in terms of computational resources [13] [14]. That being said, even trusted nodes have no incentive to continue to behave in the best interest of the ledger.

To date, there has been only one proposal for automatic UNL generation. Ripple+ [15], between other improvements brings a mechanism to select UNLs based on a structure of core and leaf nodes. The solution, however, implements changes to the core of the consensus process, delegating authority to the called core nodes. Our work aims to propose a framework for the generation and maintenance of UNLs on

top of the already existing consensus algorithm.

IV. Proposal

A. Properties

1) Nested: Since UNL overlap is still an open topic in XRP LCP research and a crucial requirement for the trust overlay, we propose a solution that allows **nesting**. When several lists of nodes are picked from a random graph, representing each set as a linear structure may not satisfy the overlapping requirement. And more than that, it gives no control over the number of nodes at the intersection of such sets.

A nested structure as a tree is more suitable for this representation, as we may represent every UNL as a tree and the set of all UNLs as a forest. Overlapping this forest gives us a directed acyclic graph (DAG) as shown in Figure 2. This structure provides flexibility in choosing the intersection size between two or more UNLs and the general percentage of overlap of the entire space. More details are given in the next section.

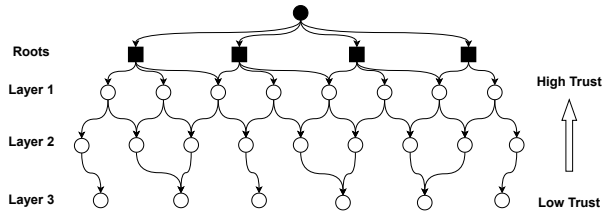

Fig. 2. DAG formed after grouping the nodes into lists represented by trees

2) Layered: The nodes in the XRPL network have no direct incentive to become validators [16]. Consequently, there is no mechanism to incentivize validators to behave in the best interest of the network once they are added to the main UNL. We propose a reputation-based system comprising **layers** of trust. This structure is the foundation for developing an awarding system that gives monetary incentives to nodes to grow their good reputations.

3) Autonomous: The curation and maintenance of the main UNL are manual, with nodes sending their candidacy for manual consideration. We need a solution able to self-curate to ensure more reliability and greater authority dispersion and autonomy. We then need a solution that can control the amount of overlap between the lists and the mobility between layers without human intervention. To tackle this issue, we propose the use of Token-Curated Registries (TCRs) [17]. TCRs are lists curated in a decentralized way, using tokens to reward curators and incurring monetary penalties for low-quality content and bad curation.

TCRs allow us to have an automatic and transparent process for generating, curating, and maintaining UNLs. It also gives us tools to control layers, using some smarter implementations, such as Nested and Layered TCRs [18] [19]. Another advantage is the possibility of implementing a rewarding system based not only on good behavior but also on good curation.

4) Continuous: The XRPL is built on top of an unstructured p2p network. As such, it comprises a dynamic system with nodes joining and leaving at any rate. Participants may also suffer from malicious intrusions, bad network conditions, and a variety of other byzantine behaviors. To account for this dynamic scenario, UNLs must adapt flexibly without compromising safety and liveness.

Nodes must be continuously observed and easily removed from trusted positions, providing the trust overlay the ability to self-arrange. We propose the use of continuous TCRs, giving token holders the ability to challenge validators that perform below the threshold of their layers [20].

B. Architecture

To fulfill the desired properties described above, we propose a framework based on the architecture shown in Figure 3. This section briefly describes each component and the interaction between them. We further expand on how they work and their concepts in the three following sections.

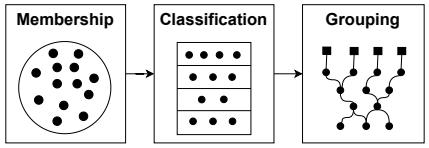

Fig. 3. NLAC Modules Architecture

The first component, called *Membership*, comprises a layer in which the validators are selected to be part of the *UNL space*. The UNL space encompasses all validators considered trustworthy by the network. Currently, the XRPL network already has the UNL space created, with 34 nodes manually selected to be part of the trusted set. In this work, we propose the use of *Token-Curated Registries* [17] to create the UNL space without human interference.

After defining the membership in the set, we move on to the next component: *Classification*. The idea is to classify all validators present in the UNL space into trust levels according to a score obtained by their peers. The nodes at higher levels are said to be more reliable than the nodes at lower levels. These higher-level nodes have fewer chances to behave in a byzantine way.

Having the UNL space classified in layers of trust, we need to group validators into lists to guarantee the dispersion of authority, hence the *Grouping* module. NLAC organizes lists as trees, taking advantage of the trust layers generated in the above component. These trees overlap each other, forming a *Directed Acyclic Graph* (DAG) in the form of a forest, as shown in Figure 2, which means that some nodes will be part of two or more lists. To ensure reliability, highly trustable nodes are near the roots, and less trustable nodes stay in the leaves.

V. Membership

A. Token-Curated Registries

In its most naïve form, TCRs are simple publicly curated lists. Listees have no direct monetary reward, but indirect

gains for figuring on high-quality lists. Take the example of a list of the best European universities in a certain field. Figuring on this list may ensure the universities a higher number of applications and higher investments in the particular field.

Each list usually has an associated token. To participate in the curation, one needs to possess a certain amount of tokens. Token holders can vote and challenge candidacies. To apply to the list, a user needs to stake a certain amount of tokens, which are held in a pool associated with the candidate and are untouched until the candidacy process is over.

During the candidacy period, any token holder can challenge the entry if they judge the candidate as not suitable for the list. To issue a challenge, the holder must stake tokens within the pool. All holders can vote to accept or reject the challenge. If the majority votes to accept the challenge, the challenger receives the staked tokens back, plus a reward for good curation. The candidate, for its part, loses its staked tokens and is not included in the list. However, if the holders vote to reject the challenge, the challenger is the one who loses the stake. In this situation, the conclusion is the same as if no challenge has been issued during the candidacy period: the candidate receives their tokens back and gets included in the list.

B. UNL Membership

To figure in the space of trusted validators, a node must stake a minimum amount of tokens within a token pool; Then a candidacy period starts. During this period, the node is evaluated by its peers, who will construct the first reliability score. The proposals formed by this node are only observed for scoring and will not be taken into consideration until the node is included in the NLAC structure.

At the end of the evaluation period, the upper layer nodes can challenge the candidate if the guidelines [13] were not met or if the reliability score is lower than a certain threshold. All the nodes that were observing the candidate can vote to reject or accept the challenge. Note that it is not reasonable to have all validators in the trust overlay evaluating a candidate. This approach would require candidates to transmit their messages to the entire network, causing a high message overhead. The proposal for NLAC is that only a subset of validators observe and evaluate the candidate.

If not challenged, the candidate receives the staked tokens back and is included as a leaf, considering the balance of the tree, so UNLs have approximately the same size. If a challenge is issued, the appointed curators vote to determine the outcome. In case of dismissal, the challenger loses their tokens; the candidate receives their tokens back and is included as a leaf. In case of acceptance, the challenger receives their tokens plus a reward, and the candidate is not included on any list, losing the stake.

After inclusion, the new listee will continue to be observed by the appointed peers. These peers may change as the tree adapts to the conditions of the network. If the validator starts presenting byzantine behavior, if the guidelines cease to be followed, or in case of complete disconnection, any appointed peer may issue a challenge. Acceptance of the challenge means complete removal of the validator from the DAG, which means that there is no downward movement. A node in a higher position has more to lose than a node present at a lower level, guaranteeing that highly trusted validators will continue to act honestly on behalf of the ledger.

VI. Classification

The Classification module serves not only as a necessary step for guaranteeing minimal levels of trust in each UNL but also as a mechanism for rewarding trustable nodes. The idea is to employ a Layered TCR [18] to incentivize the nodes to behave honestly to be promoted to higher levels, resulting in monetary rewards for curating lower levels.

In its current implementation, the XRPL network employs a *Reliability Measurement* mechanism to identify faulty nodes that should be added to the nUNL. This measurement comprises the percentage of times a validator issued a validation that is agreed upon by the network to be true, considering the last 256 ledger versions. Each node computes the reliability score for each of its peers using the following equation [10]:

$$Reliability = V_a \div 256$$

Being V_a the total number of validations received from a given peer that a node judges to be true considering its ledger versions. When the reliability score of a peer falls below 50%, the node can submit a proposal to add the said peer to the nUNL. The network must reach a consensus on adding or removing validators from the nUNL.

Although this mechanism works well for temporary disruptions, it does not permanently remove or add validators to the UNL space, its sole purpose is to tell the network to temporarily ignore *proposals* and *validations* coming from faulty nodes. We propose an extension of the nUNL, in which we use a measurement similar to the reliability score to classify nodes into layers of trust, with more trustable nodes placed into higher layers.

The movement between layers can happen only upward to guarantee that nodes that become untrustable for a prolonged time are automatically removed from the UNL space, so nodes have an extra incentive to behave in the best interest of the ledger. The upward movement happens in two instances, first when a node applies to be promoted to higher levels, and second when a high-level node is removed or promoted, and there is a node in the immediate level below that fulfills the minimum score required.

When a node applies for promotion, it must stake tokens within the pool and go through the same process described in the previous section. In the event that another node is removed or promoted, another node can propose a validator that it believes to be the best candidate for promotion. Again, by staking tokens within the pool and going through the curation process. In the same way, if the score of a node falls below the threshold set for the layer it sits on, a node

on the levels above can propose its removal by staking tokens and going through the same process.

The reason for not automatically promoting nodes as soon as they reach the minimum score required for a given layer is to promote rewards for nodes on higher levels to curate the UNL space. Every proposal to modify the space in any way requires tokens to be staked, and, safe from unchallenged additions, rewards to be issued. This method rewards good curation and monetizes the participation of nodes in the UNL. It also provides incentives for nodes to behave correctly to maintain higher levels, where they can receive rewards for curating lower levels.

The number of levels and minimum score required, as well as a more in-depth study on the most suitable way to score nodes, is not the focus of this work. Further work on this topic is still necessary, considering the underlying characteristics of the XRPL network.

VII. Grouping

The final module comprises the most important part of the NLAC framework. It is where trustable validators are combined into lists with the goal of better distributing the decision power while keeping the maximum possible trust in the system. To achieve that, we employ optimization methods to maximize the total trust of the system, considering each layer of trust generated in the previous module.

Figure 2 shows the structure that we aim to achieve after optimizing the set. The black circle represents the root of the forest from which all lists derive. In the illustration, we show four lists, each of them rooted in what we represent as black squares. Each trusted validator is represented by a white circle. By traversing each tree, we obtain the UNLs we seek to represent. The entire forest shows how the lists overlap each other to create a tightly connected space while maintaining a certain degree of dispersion.

There are three layers of trust represented, with the higher levels closer to the root. The closer a node is to the leaves, the lower its reliability score. To avoid low-trust nodes being the majority on a list or being in too many lists at once, NLAC limits the maximum number of low-trust nodes in each UNL, meaning that the number of leaf nodes should also be limited, so we can avoid the scenario of low-trust nodes massively skewing the total trust of the system.

A. Mathematical Model

We generated a mathematical model for grouping the validators according to the trust layers fulfilling some conditions modeled as constraints while trying to achieve the maximum total trust in the system. We modeled the structure representing the UNL memberships as the matrix X with

$$x_{ij} = \begin{cases} 1 & \text{if } node_i \in UNL_j \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Thus, the row i represents the node i and the column j represents the UNL j, $i \in \{1, ..., N\}$ and $j \in \{1, ..., M\}$ with N the number of nodes in the UNL space and M the number of UNLs to be generated by the optimizer. Let K be the maximum number of layers generated in the previous module. Consider the trust vector y with $y_i \in \{1, ..., K\}$, where node i belongs to trust level k.

Maximizing the total trust of the system amounts to finding X_{opt} with

$$X_{\text{opt}} = \underset{X \in \{0,1\}^{N \times M}}{\arg\max} ||y^T X||_2 \tag{2}$$

To simplify the model, we do not employ the reliability score generated internally by each node for their peers because different nodes may have different scores for each validator. The average score of a node is already reflected in the level they occupy, which is agreed upon by the entire network.

The trivial solution for this objective function is X_{opt} being the matrix of ones. This, however, does not fulfill the dispersion requirement imposed previously. For this reason, we impose the following constraints:

First, we want to enforce a minimum size U for each UNL:

$$||Xe_j||_1 \geq U \qquad \forall j \in \{1, ..., M\} \tag{3}$$

with $||\ .\ ||_1$ being the L^1-norm and $e_j \in \mathbb{R}^M$ the j-th basis vector.

We want to guarantee that any two UNLs have a minimum overlap P:

$$(Xe_p)^T Xe_q = e_p^T X^T Xe_q \geq P \quad \forall\ p, q \in \{0, .., M\} \tag{4}$$

Considering that all nodes have the same voting power, to guarantee that low trust nodes will not be part of too many UNLs and skew the power balance, we limit the number of UNLs a node can be part of based on the trust level it is part of. Let L_k be the parameter that identifies the maximum number of UNLs a node at the trust level k can be part, with $k \in \{0..K\}$.

$$||e_{j_k}^T X||_1 \leq L_k, \qquad j_k \in \{1, ..., N\} \tag{5}$$

Another important limitation is the maximum number of nodes in a given layer a UNL can have. Let Q_k be the maximum number of nodes of trust level k the UNL j may have, then:

$$||v_k^T Xe_j||_1 \leq Q_k \qquad \forall k \in \{1, ..., K\} \tag{6}$$

where

$$(v_k)_i = \begin{cases} 1 & \text{if Node } i \text{ belongs to trust level } k \\ 0 & \text{otherwise} \end{cases}$$

B. Experimental Evaluation

We tested[2] the above formulation using Pyomo [21] with the cplex solver [22] on a Core™ i7-8665U CPU@1.90GHz × 8 machine with 16GB of RAM running Ubuntu 22.04.2 LTS. We fixed the values of $U = N * (1/3)$ and $K = 4$,

[2]Code available at https://github.com/FlavScheidt/NLACOptimization

TABLE I
TRUST OPTIMIZATION EVALUATION

N	M	P	Total Trust	Constraints	Variables	Time
24	8	80%	280	121	193	0.05s
24	16	90%	280	193	385	0.05s
32	16	80%	436	209	513	0.39s
32	24	90%	436	281	769	4.38s
64	16	80%	712	273	1025	0.55s
64	32	90%	872	417	2049	18.47s

also distributing the nodes randomly through the layers. The values for the vectors L and Q were $L = [16, 8, 4, 2]$ and $Q = [12, 8, 4, 2]$. These values were empirically obtained on a *testnet* of 24 nodes and 8 UNLs.

Table I shows the results of six combinations of M, N, and P, considering that these values can change depending on the state of the network. The goal of the optimization is to group N validators into M lists respecting the previous constraints while maximizing the trust of the system. The table also shows how many constraints and variables were generated by the model and the solution time.

The total trust of the system naturally grows as a function of N, however, it does not seem to be affected by M when $N < 64$. The total solution time is also not affected by M when $M < 24$. It is important to note that for $N = 64$ and $M = 32$ the solution time may be longer than is feasible for a distributed solution based on the XRP LCP. However, when $M > 2$ the complex scenario needs a deeper analysis, as shown previously in Section II. That being said, our tool provides means for maximizing trust, but some parameters require a more in-depth analysis.

VIII. CONCLUSION & FUTURE WORK

Overall, NLAC represents an important step forward in the development of the XRP Ledger. By providing a framework for automatic curation and self-arrangement, NLAC enhances the unique features of the XRP Ledger Consensus Protocol, while addressing key issues related to trust and dispersion of authority, as well as delineating a mechanism for implementing a rewarding system on the network.

NLAC comprises three main modules, all with tunable parameters, leaving space for further improvements as the network evolves. These modules were constructed based on four desirable properties derived from the characteristics of the XRP Ledger trust overlay: *nesting*, *layering*, *autonomy*, and *continuity*. The first module employs Token-Curated Registries to establish membership in the UNL space, followed by the second module, which uses layered Token-Curated Registries to create levels of trust. These levels are then used in the last module, where nodes are grouped into different lists, following guidelines to achieve maximum trust while guaranteeing a better dispersion of authority.

However, future work is to be done on the subject. NLAC still lacks a more detailed analysis of its behavior in a production unstructured p2p network. Having tunable parameters, further analysis of the ideal values for each network configuration is also necessary.

ACKNOWLEDGMENT

We thankfully acknowledge the support from the RIPPLE University Blockchain Research Initiative (UBRI) for our research.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: www.bitcoin.org

[2] D. Schwartz, N. Youngs, and A. Britto, "The ripple protocol consensus algorithm," 2014. [Online]. Available: https://ripple.com/files/ripple_consensus_whitepaper.pdf

[3] I. Amores-Sesar, C. Cachin, and J. Mićić, "Security analysis of ripple consensus," 2020. [Online]. Available: https://arxiv.org/abs/2011.14816

[4] W. Hao, J. Zeng, X. Dai, J. Xiao, Q. S. Hua, H. Chen, K. C. Li, and H. Jin, "Towards a trust-enhanced blockchain p2p topology for enabling fast and reliable broadcast," *IEEE Transactions on Network and Service Management*, vol. 17, pp. 904–917, 6 2020.

[5] F. Armknecht, G. O. Karame, A. Mandal, F. Youssef, and E. Zenner, "Ripple: Overview and outlook," vol. 9229. Springer Verlag, 2015, pp. 163–180.

[6] B. Chase and E. MacBrough, "Analysis of the xrp ledger consensus protocol," 2018. [Online]. Available: https://arxiv.org/abs/1802.07242

[7] K. Christodoulou, E. Iosif, A. Inglezakis, and M. Themistocleous, "Consensus crash testing: Exploring ripple's decentralization degree in adversarial environments," *Future Internet*, 2020. [Online]. Available: www.mdpi.com/journal/futureinternet

[8] "Run rippled as a validator," accessed: 2022-07-28. [Online]. Available: https://xrpl.org/run-rippled-as-a-validator.html

[9] J.-P. Vergne, "Decentralized vs. distributed organization: Blockchain, machine learning and the future of the digital platform," *Organization Theory*, vol. 1, p. 263178772097705, 10 2020.

[10] "Negative unl," accessed: 2023-02-14. [Online]. Available: https://xrpl.org/negative-unl.html

[11] P. Baran, "On distributed communications networks," *IEEE transactions on Communications Systems*, vol. 12, no. 1, pp. 1–9, 1964.

[12] V. Buterin, "What proof of stake is and why it matters," *Bitcoin Magazine*, vol. 26, 2013.

[13] "The foundation unique node list," accessed: 2023-03-31. [Online]. Available: https://foundation.xrpl.org/unl/

[14] "System requirements," accessed: 2023-03-31. [Online]. Available: https://xrpl.org/system-requirements.html

[15] C. Ma, Y. Zhang, B. Fang, H. Zhang, Y. Jin, and D. Zhou, "Ripple+: An improved scheme of ripple consensus protocol in deployability, liveness and timing assumption," *CMES - Computer Modeling in Engineering and Sciences*, vol. 130, pp. 463–481, 2022.

[16] "Introduction to consensus," accessed: 2022-09-13. [Online]. Available: https://xrpl.org/intro-to-consensus.html

[17] M. Goldin, "Token-curated registries 1.0," 2017, accessed: 2022-09-13. [Online]. Available: https://medium:com/@ilovebagels/token-curated-registries-1-0-61a232f8dac7

[18] M. Lockyer, "Token curated registry (tcr) design patterns," 2018, accessed: 2022-10-19. [Online]. Available: https://hackernoon:com/token-curated-registry-tcr-design-patterns-4de6d18efa15

[19] D. de Jonghe and F. Gong, "The layered tcr," 2018, accessed: 2022-10-19. [Online]. Available: https://blog.oceanprotocol.com/the-layered-tcr-56cc5b4cdc45

[20] S. de la Rouviere, "Continuous token-curated registries: The infinity of lists," 2017, accessed: 2023-02-15. [Online]. Available: https://shorturl.at/bAISZ

[21] M. L. Bynum, G. A. Hackebeil, W. E. Hart, C. D. Laird, B. L. Nicholson, J. D. Siirola, J.-P. Watson, and D. L. Woodruff, *Pyomo–optimization modeling in python*, 3rd ed. Springer Science & Business Media, 2021, vol. 67.

[22] I. I. Cplex, "V12. 1: User's manual for cplex," *International Business Machines Corporation*, vol. 46, no. 53, p. 157, 2009.