A Formal Approach for the Analysis of the XRP Ledger Consensus Protocol

Lara Mauri[1][a], Stelvio Cimato[1][b] and Ernesto Damiani[1,2][c]
[1]*Computer Science Department, Università degli Studi di Milano, Milan, Italy*
[2]*EBTIC - Khalifa University of Science and Technology, Abu Dhabi, U.A.E.*

Keywords: XRP Ledger, Ripple, Cryptographic Protocol, Consensus, Distributed Ledger.

Abstract: Distributed ledger technology is envisioned as one of the cornerstones of promising solutions for building the next generation of critical applications. However, there is still quite a bit of confusion and hype around the real security guarantees this technology offers. This is especially due to the fact that for the vast majority of existing blockchain-based consensus protocols it is really hard to find sufficiently detailed documentation that fully captures their behavior. A number of recent papers have formalized the behavior of Bitcoin-like protocols in order to rigorously study the security and privacy properties of their underlying structure, but surprisingly very little work has been devoted to the formalization of distributed ledger systems using BFT-like approaches. In this work, we focus on *XRP Ledger*, better known as *Ripple*, and take the first steps towards the complete formalization of its consensus protocol. To this end, we have investigated all the existing documentation and analyzed its source code. We present a formal description of its consensus protocol for every step. Furthermore, we provide an accurate view of its security guarantees in terms of safety and liveness and show how to increase the desired tolerance by changing the value of specific protocol parameters.

1 INTRODUCTION

Although the classical problem of achieving mutual agreement in the presence of faulty nodes (Pease et al., 1980) has been extensively studied in the literature from different perspectives (e.g. resiliency, network connectivity, communication complexity, running time and channel reliability), interest in consensus protocols has risen quite significantly over the last few years because of their application to various novel settings such as blockchain systems. The novelty of blockchain technology is a powerful combination of well-known research results taken from distributed computing, cryptography and game theory that together create a system providing a trustworthy service to parties who otherwise have no reason to trust each other. The term *blockchain* is self-explanatory and denotes a distributed ledger (i.e. a public data structure) where inputs called transactions are aggregated in the form of blocks which are added one after another. A copy of the ledger is locally maintained by all the nodes, which operate in a peer-to-peer communication setting and continuously send and validate transactions in an attempt to update the state of the shared ledger. In principle, any participant can propose the addition of a new block to the blockchain, but this addition has to be approved by the other peers to ensure its legitimacy.

The idea behind what is today known as the *Nakamoto consensus* (or *ledger consensus*) was first conceptualized in 2008 with the birth of the Bitcoin payment system (Nakamoto, 2008) and refers to the consensus mechanism allowing the nodes in the Bitcoin network to decide which next block to include into the ledger. Specifically, the ultimate goal of the consensus protocol is to ensure that parties have a unique common view of the ledger even in the presence of possible conflicting inputs and arbitrary Byzantine behaviors of some network nodes. In order to guarantee consistency between potentially different versions of the ledger, Bitcoin relies on the concept known as *proof-of-work* (PoW), which requires a party to invest computational power to affect the behavior of the system. However, the extraordinary amounts of energy it demands have raised concerns over its sustainability and environmental im-

[a] https://orcid.org/0000-0002-4024-1015
[b] https://orcid.org/0000-0003-1737-6218
[c] https://orcid.org/0000-0002-9557-6496

Mauri, L., Cimato, S. and Damiani, E.
A Formal Approach for the Analysis of the XRP Ledger Consensus Protocol.
DOI: 10.5220/0008954200520063
In *Proceedings of the 6th International Conference on Information Systems Security and Privacy (ICISSP 2020)*, pages 52-63
ISBN: 978-989-758-399-5; ISSN: 2184-4356

pact. These limitations motivated an interesting research direction that proposes the adoption of a new class of consensus protocols based on the use of alternative resources (Bano et al., 2017; Wang et al., 2018) – among them, the *proof-of-stake* (PoS) paradigm is the most implemented so far. Such developments, combined with recent efforts to leverage the core features of blockchains for novel use cases in both financial and non-financial domains (Rawat et al., 2019), have led to a significant increase in proposals for blockchain-based solutions, with new software applications and services appearing daily. But this rush to quickly enter the market poses serious risks to the entire blockchain ecosystem since vulnerabilities of insufficiently tested code can be exploited with significant consequences due to the decentralized nature and irreversibility of the technology (Saad et al., 2019).

According to Halpin and Piekarska (2017), an emerging field that needs further research is the study of the security and privacy properties of the structure underlying blockchain-based systems. In this context, the first formal security analysis of the fundamental principles behind the Nakamoto consensus was proposed by Garay et al. (2015). In their work, the authors presented an abstraction of the Bitcoin protocol in synchronous networks, which they referred to as the *Bitcoin backbone*, and proved that it satisfies certain security properties. Later, Pass et al. (2017) extended the analysis to asynchronous networks, and further refinements of the initial model of computation were presented in subsequent works (Garay et al., 2017, 2018). While a number of other papers have studied the security of several PoW- and PoS-based blockchains in a rigorous manner (Pass and Shi, 2017; Badertscher et al., 2017; Daian et al., 2016; Kiayias et al., 2017), it is worth noting that for the vast majority of existing blockchain protocols it is really hard to find sufficiently detailed information on the functioning of the consensus mechanism they use (though there is a lot of work presented online in the form of white papers, pre-prints and blog posts). Moreover, the majority of current research is focusing on studying blockchains consistent with the so-called *permissionless* setting, whereas much less attention has been paid to the formalization of distributed ledger systems operating in *permissioned* settings, which embrace aspects of traditional *Byzantine-fault tolerant* (BFT) consensus protocols. In contrast to permissionless blockchains such as Bitcoin, permissioned ones are operated by known entities and only a restricted group of nodes can participate in the process for determining the correct update of the ledger (Cachin and Vukolic, 2017).

Motivated by this emerging new era of research, in this work we focus our attention on *XRP Ledger* – better known as *Ripple* (Ripple Labs Inc., a) –, the third-largest digital currency in market capitalization after Bitcoin and Ethereum. The XRP Ledger protocol is a distributed payment system that uses a unique consensus protocol which sets it apart from previous approaches. Based on the idea of subjective trust assumptions, in Ripple every participant is free to specify which nodes it believes will behave correctly and which ones it considers faulty by means of so-called *Unique Node Lists* (or *UNLs*). The fact that the XRP Ledger protocol does not rely on a process of mining to verify transactions has many advantages, in terms of transaction speed, compared to other consensus algorithms (Mauri et al., 2018). Indeed, while for Bitcoin-like protocols transactions are confirmed after an hour on average, in the XRP Ledger network the settlement process takes around 5 seconds, with a throughput of up to 1500 transactions per second.

Although mentioned in several surveys on the consensus mechanisms that govern the most popular blockchains (Xiao et al., 2019; Gramoli, 2017), the Ripple protocol is often described in a vague and not very clear way. By now the original white paper (Schwartz et al., 2014) is deprecated, and available documentation about how the consensus process works is restricted to the developer portal (Ripple Labs Inc., b) and a recent analysis provided by the creators of Ripple themselves (Chase and MacBrough, 2018). Even if the latter presents a detailed explanation of the Ripple algorithm and derives conditions for its safety and liveness, there remains a degree of uncertainty around many aspects of the protocol and, in any case, not all steps have actually been described. Furthermore, to date the only existing peer-reviewed analysis of the XRP Ledger consensus protocol (Armknecht et al., 2015) was conducted on the original white paper and showed that some specifications were flawed.

Our Contributions. In light of the above, we believe that having a clear and accurate description of the XRP Ledger consensus protocol is fundamental not only to gain in-depth understanding of its behavior, but also to increase trust in the foundational principles of the algorithm and identify its vulnerabilities. To this end, we have carefully investigated all the existing documentation relating to XRP Ledger and, by directly analyzing its source code (Ripple Labs Inc., c), we have created an in-depth description of the consensus protocol for every step. In particular, our work contributes in the following aspects: (1) taking the first steps towards the complete formalization of the XRP Ledger Consensus Protocol; (2) providing an accurate view of the current security guarantees of the protocol in terms of safety and liveness; (3) showing how the inherent correlation between the safety and liveness tolerances, the validation quorum and the UNLs overlapping size can be leveraged to increase either liveness or safety.

Organization. The remainder of the paper is organized as follows. Section 2 provides the necessary background to understand the consensus mechanism powering the XRP Ledger network. Section 3 presents our formalization of the different phases of the XRP Ledger Consensus Protocol, expressed by means of a set-theoretic notation and mathematical formulas. Section 4 reports and clarifies results from a previous analysis with respect to safety and liveness properties and extends it by showing that the value of some protocol parameters can be changed to guarantee more liveness or safety. Section 5 concludes and proposes directions for future research.

2 BACKGROUND ON THE XRP LEDGER NETWORK

2.1 Basic Concepts and Terminology

In the XRP Ledger network, the collaborative process by which the shared replicated transaction system is updated and kept consistently synchronized is known as the *XRP Ledger Consensus Protocol* (or *XRP LCP*, for short). The primary objective of each participant involved in the XRP LCP is to advance the chain of validated ledgers by reaching agreement on a new set of transactions to apply to the prior ledger. The overall consensus process, which can be viewed as a progression through three phases – *collection, deliberation* and *validation* – is controlled by the so-called *validators*. Unlike simple tracking nodes, which are uniquely responsible for relaying transactions from clients throughout the network, participants acting as validators perform the additional task of proposing and revising *candidate sets of transactions* to be processed.

Several permissioned blockchain platforms rely on classical BFT consensus protocols, which typically require every node to know the entire set of peers participating in the consensus process. However, one of the major challenges for those protocols that prevents their wider adoption in blockchain systems is their scalability in terms of the number of nodes (Vukolić, 2016). Conversely, the XRP Ledger protocol does not rely on a strict notion of membership and not every node has to trust all others. This gives the XRP Ledger network a clear advantage over BFT-based blockchains because it is more scalable in terms of the number of nodes. In order to achieve consensus, validators share information about their candidate transaction sets by exchanging signed messages (whose format will be described in detail later on), but nodes only consider messages sent by specific validators they believe will behave honestly. In particular, each participant individually chooses which validators it will listen to when making decision about the next ledger, and such choice is declared in the form of a list, referred to as the *Unique Node List* (UNL), which is locally maintained by each validator; validators belonging to the UNL are called *trusted*.

A fundamental component of the first phase of the consensus protocol is represented by the *open ledger*, which is the ledger to which all pending transactions are applied. When certain specific conditions are met, the ledger is *closed*, and the closure implies that new transactions are no longer accepted for inclusion in the new version of the ledger. In essence, the set of transactions included in the closed ledger constitutes the first *position* and reflects the validator's initial belief of which transactions should be considered for the next ledger. Nodes are equipped with a public/private key pair. Digital signatures are used to check that submitted transactions are correct, i.e. authorized to do a specific set of actions, but also serve as a means to authenticate the issuers of the messages received. The fact that all protocol communications are signed makes it possible to reliably identify validators in the peer-to-peer network even if messages are relayed by untrusted nodes.

As mentioned above, validators communicate by means of messages in an attempt to reach network agreement. One of such messages is represented by the so-called *proposals*. At the start of the consensus process, each node decides whether to operate as an active or passive validator. A *proposing* validator is a normal participant taking on its own position and sending proposals to other peers. Conversely, an *observing* validator updates its position, but does not propose it to its peers. The first proposal issued by a validator corresponds to its initial position, so proposals can be seen as subsequent positions taken by nodes as the network progresses. As we will see later, the concept of *dispute* is strictly connected to that of proposal. It is used to denote an individual transaction that is either not included in a validator's proposal or not included in a trusted validator's proposal.

In addition to the closed ledger, we can distinguish two other types of ledger representing subsequent evolutions, in terms of content validity and finality, of the initial ledger version under consideration. Specifically, the ledger resulting from the completion of deliberation, known as the *last closed ledger*, is the most recent ledger on which, from the point of view of a single validator, the network reached agreement. This ledger version is not immediately considered final. Indeed, a ledger is *fully validated* only once it passes a certain validation threshold. It is only at that point that it is considered authoritative and irrevocable, meaning it becomes part of the permanent public history and its content is truly reliable.

Lastly, a *validation* is a signed message containing the hash of the last closed ledger, whose purpose is to let trusted validators know which particular ledger was built by the other validators and come to a common conclusion about which last closed ledger should be considered authoritative (i.e. declared fully validated).

2.2 Informal Protocol Description

The XRP LCP proceeds in rounds and consists of three different phases that are continually repeated to process groups of transactions. The typical execution of a single ledger round goes from collection to deliberation to validation. Below we provide a brief description of each phase:

- *Collection*: During this phase, validators collect new incoming transactions, apply the well-formed ones to their open ledger and subsequently relay them to the other peers. When certain conditions are met, each validator determines the open ledger period should end and closes its ledger.
- *Deliberation*: Validators begin an iterative process in which they try to establish consensus by exchanging proposals with their trusted validators. Starting from their own position, they continuously change their view of the transaction set by adding or removing disputed transactions from their proposal and, after each update, they check the percentage of trusted validators agreeing with them. When a validator sees a supermajority agreement, it declares that consensus has been reached. Then, it creates a new last closed ledger by applying the consensus transaction set to the last validated ledger and issues a validation.
- *Validation*: Participants determine whether to consider the outcome of deliberation (i.e. their last closed ledger), fully validated by comparing the hashes received from their trusted peers. The decision is taken on the basis of the percentage of agreement on the newly-created ledger. Only when a quorum of validations for the same ledger is reached, that ledger is deemed final.

3 FORMALIZING THE XRP LCP

3.1 Preliminaries

In the XRP Ledger network, the shared global ledger is actually a series of *ledger versions* (or *ledgers*). A ledger is any quintuple of the form $L = \langle sn, h, h_{-1}, ts, T\rangle$ where sn is the sequence number of the ledger, h is the unique identifying hash of the ledger's contents, h_{-1} is the h value of the previous ledger version this ledger was derived from, ts is a timestamp, and T is a set of transactions. The hash serves as a unique identifier for the ledger and its contents, whereas the sequence number identifies the order in which ledgers occur in the ledger chain. The very first ledger, known as the *genesis ledger*, has sequence number 1; each successive ledger has a sequence number one greater than that of the preceding one. We say a tuple $L = \langle sn, h, h_{-1}, ts, T\rangle$ is valid with respect to a predecessor (fully validated) ledger $L' = \langle sn', h', h'_{-1}, ts', T'\rangle$ iff:

1. $h_{-1} = h'$;
2. $sn = sn' + 1$;
3. $|\Sigma_L| \geq q_v$, where $|\Sigma_L|$ is the support of the ledger L and q_v is the minimum percentage of participation required to reach network agreement.

Below we introduce the notation used in the formalization of the XRP LCP provided in Section 3.2, as well as the timing assumptions.

3.1.1 Notation

We denote by N the universe of nodes of the peer-to-peer XRP Ledger network. Nodes play different roles: client applications submit transactions to server nodes, which are differentiated in tracking nodes and validators. The consensus process is exclusively driven by the latter. Since only validators actively participate, check and validate all transactions taking place in the system, we believe that for the purposes of formalization it is meaningful to consider only the subset $N_v \subset N$ of participants, where N_v denotes exactly the nodes acting as validators. As already mentioned, during the consensus process, validators only evaluate proposals and validations issued by their trusted nodes, discarding those received from validators not belonging to their UNL. For any node $i \in N_v$, $UNL_i = \{u : u \in N_v, F(u)\}$ denotes the set of all nodes u for which $F(u)$ is true, where $F(u)$ means that validator i trusts u. Also, we use $n_i = |UNL_i|$ to denote the size of node i's UNL. Giving as a fact that the individual choice to include (or omit) a given transaction in the next ledger is influenced only by the trusted nodes' messages, we decide not to directly specify UNL membership every time (in our formalization, we will specify this dependency only in the context of the last phase). Deliberation and validation phases are parameterized by q_c and q_v, respectively. In particular, q_c specifies the *consensus quorum* (i.e. the minimum number of validators having issued the same proposal needed to declare consensus), whilst q_v − see the third condition required for ledger validity − represents the *validation quorum* (i.e. the minimum number of validations needed to fully validate a ledger). Both these parameters are a function of a constant k and n_i, where n_i is the previously defined size of UNL_i. We let $\tilde{L}_i$, $\tilde{L}_i^c$ and L_i' denote the open ledger, the closed ledger and the last closed ledger of node i, respectively. Moreover, we use $\hat{L}$ to denote the last fully validated ledger. tx is used to represent a single transaction under consideration by consensus. Any transactions excluded from a node's proposal are added to its local queue we denote by T_Q. For notational simplicity, we suppose the existence of another set T_I which contains all received new transactions. Also, we use P_i to denote the node i's proposal of candidate transactions and the symbol θ to indicate the threshold for including a given transaction in the proposal P_i. Lastly, D_i denotes the node i's set of disputed transactions, whereas σ_i denotes a validation issued by i. Table 1 provides a summary of the notation used throughout this paper (some parts however will be slightly modified to suit the context).

Table 1: Summary of notation.

N_v	The set of validators
i	A validator in the network
UNL_i	i's Unique Node List
n_i	The size of UNL_i
tx	A single transaction
T	A set of transactions
T_Q	The set of queued transactions
T_I	The set of new transactions
q_c	The consensus quorum
q_v	The validation quorum
$\tilde{L}_i$	i's open ledger
$\tilde{L}_i^c$	i's closed ledger
L_i'	i's last closed ledger
$\hat{L}$	The last fully validated ledger
P_i	i's proposal of transactions
D_i	i's set of disputed transactions
σ_i	i's validation

3.1.2 Timing

An important aspect that must be considered when designing any consensus protocol is the ability of parties to reach a certain degree of synchronization during the protocol execution. In the XRP Ledger network, the protocol execution is driven by a heartbeat timer which allows nodes to advance the consensus process. In practice, at regular intervals each validator checks whether the necessary conditions to move to the next phase of the consensus protocol are met. Although each validator can begin a new round of consensus at arbitrary times based on when the initial round started and the time taken to apply the transactions, the transition to both deliberation and validation phases can only occur at the end of the heartbeat timer.

Furthermore, each node maintains an internal clock that it uses when calculating its own *close time*, i.e. the time at which the ledger has been closed (see Sections 2.1 and 3.2.1 for details). In practice, at the closure of the ledger (which reflects the end of collection phase) each validator uses its current time as the initial close time estimate and includes this value in its initial position. Later, this approximate time is rounded based on a dynamic resolution. As a result, each validator uses the consensus process not only to reach agreement with its trusted nodes about the set of transactions, but also attempts to agree on a common time for the closure of the ledger. Validators update their close time positions in response to the effective close time included in their trusted nodes' positions, and in this way, they can derive a common close time for the ledger without the need to synchronize their clocks. When there is no clear majority of trusted validators agreeing on a close time, nodes agree to disagree on an actual close time. Even though in this case the network has no official close time, the effective close time of the ledger is set to 1 second past the close time of the previous ledger.

Although reaching agreement on the effective close time is part of the XRP LCP, in order to make our formalization more readable, we decided to include in it only some fundamental protocol timing parameters.

3.2 In-depth Protocol Description and Its Formalization

In Ripple, the ledger chain structure starts with the genesis ledger and ends with the last fully validated ledger. Generally, given the prior fully validated ledger $\hat{L}$ and a set T of new candidate transactions, the execution of the XRP Ledger Consensus Protocol determines an output ledger $\hat{L}'$ which, together with the previous validated ledgers, forms the ledger history. As introduced in the foregoing section, the XRP LCP is an asynchronous round-based protocol consisting of three phases. The diagram of Fig. 1 shows the output expected from each phase from the perspective of an individual node $i \in N_v$: *(a)* the output of the collection phase is represented by i's initial proposal P_i^0, which contains the starting position taken by the validator itself before considering any trusted validator position; *(b)* at the end of the second phase, i builds a new last closed ledger L_i'; *(c)* the validation phase establishes which last closed ledger, amongst those proposed by all participants, must be considered the authoritative one ($\hat{L}'$).

We now proceed with the formalization of the three main consensus phases. Note that in our formalization we follow the flow depicted in Fig. 1. That is, we consider the perspective of a single validator $i \in N_v$, which makes decisions only by listening to its unique node list UNL_i. In order to formalize the behaviour of the XRP LCP we referred to the current implementation of the core XRP Ledger server, called `rippled`, whose source code is available in the GitHub repository (Ripple Labs Inc., c).

3.2.1 Collection

The first stage of the process is a quiescent period allowing the validator i to create an individual perception of the state of the network. This perception is referred to as the open ledger. The node's open ledger is filled with the candidate transactions which failed to be included in the last round (i.e. those queued in T_Q), as well as newly submitted ones. i's open ledger $\tilde{L}_i$ contains the set of transactions T:

$$T = T_Q \cup T_I, \;\; T \in \tilde{L}_i \tag{1}$$

Under normal circumstances, the closure of the ledger occurs after a predetermined minimum time $t_{minclose}$ has elapsed and only if the open ledger contains at least one transaction. Thus, i closes its open ledger if the following holds:

$$T \neq \emptyset \wedge (t_{open} \geq t_{minclose}), \;\; T \in \tilde{L}_i \tag{2}$$

where t_{open} is used to denote how long the ledger has been open. We let $\tilde{L}_i^c$ indicate that i's open ledger has been closed (superscript letter c). In other cases, the protocol either *(1)* closes the ledger if more than half of the proposers have closed the ledger or validated the last closed ledger, or *(2)* postpones the closure to the end of a certain idle interval so that it is more likely to include some transactions in the next ledger. Regardless of the condition which led to the closure of the ledger, once the pre-close phase is completed, i establishes its initial position on the basis of the transactions included in the newly closed ledger. Then, i proposes it to its trusted validators in the form of a proposal, i.e. a signed message containing the transactions it believes should be included in the next ledger (recall from Section 2.1 that a node not operating in a proposing mode maintains its position internally without sharing it with peers):

$$P_i^0 = \{tx : \; tx \in T, \;\; T \in \tilde{L}_i^c\} \tag{3}$$

Here the notation used for the proposal slightly differs from that given in Section 3.1.1; we introduced the superscript 0 to denote that P_i is the initial proposal shared by i. Like ledgers, also proposals have a sequence number that is incremented each time a validator updates the transaction set contained therein.

As easy to guess, different nodes may receive different unconfirmed transactions due to network delays. As this variation in arrival time allows for each potential new ledger version to be different for each node, immediately after sharing its initial position, i considers all the proposals received from its trusted validators and creates a dispute for each transaction discovered not to be in common with the peer's position under consideration. We define the validator i's dispute set D_i as:

$$\begin{aligned} D_i = \{tx : \; & (tx \in P_i^r \wedge tx \notin P_{j \neq i}^r) \\ & \vee (tx \notin P_i^r \wedge tx \in P_{j \neq i}^r)\} \end{aligned} \tag{4}$$

where r refers to one of the successive proposals issued by i during deliberation (at this time, $r = 0$). Also, the validator keeps track of each peer's vote on each disputed transaction (voting in favour of a disputed transaction simply means believing that such transaction should be included in the next ledger). For each $tx \in D_i$, we use $v(tx)$ to denote the support given to tx by i's trusted validators:

$$v(tx) = |j \neq i : \; tx \in P_j^r| \tag{5}$$

where, as before, r is currently 0 (in a more advanced stage of the process, i.e. during deliberation, P_j^r will refer to the most recent proposal issued by j).

Unless they are part of a peer's proposal, transactions received after the closure of the ledger are not considered until the next round.

3.2.2 Deliberation

In order to achieve consensus on the specific set of transactions to be processed next, i begins an iterative process during which it issues updated proposals (i.e. updates of its initial position), which are modified based on the support obtained by each individual transaction. A heartbeat timer is the key ingredient that drives the consensus process forward: it is only on timer calls, which occur at a regular frequency, that i adjusts and issues new proposals (see Section 3.1.2).

Going into detail, the mechanism allowing differences between peers' proposals to converge is based

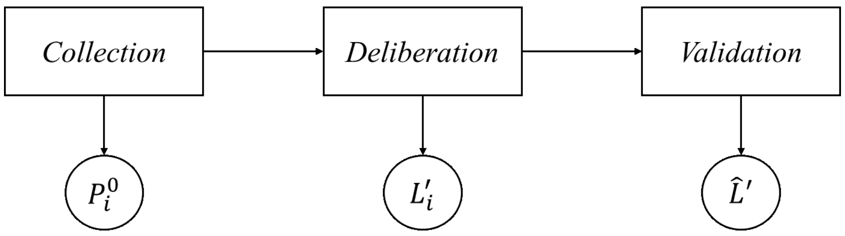

Figure 1: Output of each phase from the perspective of a single validator *i*.

on a majority voting scheme. The evaluation of the achieved convergence degree is performed taking the value of a certain threshold θ as a reference. In turn, the value θ depends on a parameter *d* which expresses the percentage of time that consensus is expected to take to converge. Specifically, *d* is a function of the previous round duration, which corresponds to the duration of the last deliberation phase (measured from closing the open ledger to accepting the consensus result) and the duration of the deliberation phase for the current consensus round:

$$\begin{aligned} d &= f(t_{(deliberation-1)},\, t_{deliberation}) = \\ &= \frac{t_{deliberation} \cdot 100}{max(t_{(deliberation-1)},\, t_{minconsensus})} \end{aligned} \tag{6}$$

where $t_{minconsensus}$ (currently, 5 seconds) expresses the minimum amount of time to consider that consensus was reached in the previous round.

As deliberation proceeds, *i* changes its candidate transaction set in response to what its trusted validators propose. As a result, *i* may either add a disputed transaction to its current set if the percentage of trusted validators that have proposed the same transaction in their most recent proposal exceeds the aforementioned threshold θ or, otherwise, remove it. New proposals are formalized as follows:

$$\begin{aligned} P_i' = \{tx :\ & (tx \in P_i^r \wedge tx \notin D_i) \\ & \vee (tx \in D_i \wedge (v(tx) \geq \theta))\}. \end{aligned} \tag{7}$$

The current implementation uses an initial threshold for inclusion of 0.5. This means that if a particular disputed transaction is supported by half of the validators or more, *i* agrees to include it in its set. On the contrary, a transaction that, at this early stage, does not have the support of at least 50% of the trusted nodes, is removed from *i*'s position. Omitted transactions are added to the transaction queue T_Q and considered again for inclusion in the next ledger version. Therefore, the queue is updated whenever a transaction is removed from a proposal:

$$T_Q' = T_Q \cup \{tx :\ tx \in D_i \wedge (v(tx) < \theta)\}. \tag{8}$$

The specific values of θ are subject to change, in accordance to the XRP Ledger implementation. Depending on the value of the function *d* (defined in Eq. 6), θ can currently assume one of the following values:

$$\begin{cases} 0.5 & d < 50 \\ 0.65 & 50 \leq d < 85 \\ 0.7 & 85 \leq d < 200 \\ 0.95 & otherwise \end{cases} \tag{9}$$

After changing its transaction view, *i* broadcasts its new proposal (increasing the related sequence number) and determines whether consensus has been reached − in case *i* has not changed its set of transactions, no new proposal needs to be issued. Consensus can be declared only if there is a supermajority support for each of the transactions the candidate set contains. Formally, the support of a proposal P_i^r is expressed as follows:

$$v(P_i^r) = |\{j \neq i :\ P_j^r = P_i^r\}|. \tag{10}$$

Actually, consensus is reached when the following three conditions are met: *(a)* a certain minimum amount of time has elapsed since deliberation began, *(b)* at least 3/4 of the previous round proposers are participating or the current deliberation phase is longer (of a given minimum amount of time) than the deliberation phase of the previous round, *(c) i*, together with its trusted peers, has reached the percentage threshold q_c above which consensus can be declared. However, for the purpose of formalization, the really meaningful condition is the third. Hence, we assume *i* declares consensus reached when the following holds:

$$v(P_i^r) \geq q_c, \ \ q_c = k \cdot n_i \tag{11}$$

where n_i is the number of trusted nodes belonging to UNL_i (see Section 3.1.1) and $k = 0.8$ in the current implementation. Once consensus is reached, *i* builds a new last closed ledger L_i' by adding the agreed-upon set of transactions to the prior fully validated ledger $\hat{L}$:

$$L_i' = \hat{L} \cup \{tx \in P_i^r :\ v(P_i^r) \geq q_c\} \tag{12}$$

and then, it broadcasts its resulting ledger in the form of a signed message σ_i containing the identifying hash of this ledger. At this point, the round is completed and *i* now builds a new open ledger on top of the last closed ledger. As previously said, the open ledger represents the basis of any validator's initial proposal and hence, the first transactions to be added to the new open ledger are those held over from the previous consensus round. Next, all valid transactions that in the previous round were received after the ledger was closed are applied. In practice, the protocol starts a new round of consensus before the third phase, i.e. validation, ends. Each participant begins a new collection phase concurrently, preparing its proposal for the next ledger version while the consensus process related to the prior ledger version is ongoing. Accordingly, at any given time, the process is characterized by a series of in-progress open ledgers, individual participants' last closed ledgers that have not been fully validated and historical ledgers that have been fully validated.

3.2.3 Validation

At the end of deliberation, as seen above, validators independently build what they believe the next state of the ledger should be. Due to latency in propagating transactions throughout the network, different validators may compute different last closed ledgers. These ledger versions have the same sequence number, but different hash values that uniquely identify their contents (see Section 3.1). It is important to note that the sequence number and the hash correlate 1:1 only for fully validated ledgers. Thus, for a given sequence number, only one ledger version can ever be fully validated. In order to converge on a universal status of the ledger and, consequently, resolve the above differences, i checks how many trusted validators have built a last closed ledger equal to its own by comparing its validation, i.e. the hash of the ledger it computed, with the hashes received from its peers; among all the most recent validations, i considers only those with the greatest sequence number. We denote by $\Sigma_{L_i'}$ the set of trusted validators that published the same validation hash as the validator i:

$$\Sigma_{L_i'} = \{j \in UNL_i : \sigma_j = \sigma_i\}. \qquad (13)$$

Based on these validations, i recognizes whether the previous consensus round succeeded or not. In particular, the node declares its last closed ledger L_i' fully validated in the event that a supermajority agreement on the calculated hash is reached. Therefore, if i detects that it is in the majority, having built a ledger that received enough matching validations to meet the validation quorum q_v, it considers the ledger fully validated. Formally, L_i' is declared fully validated iff:

$$|\Sigma_{L_i'}| \geq q_v, \;\; q_v = k \cdot n_i \qquad (14)$$

where, as for the consensus quorum q_c (cf. Eq. 11), the constant k is 0.8. Conversely, if i is in the minority, it cannot consider its ledger instance fully validated. Instead, it has to find the supermajority ledger, i.e. the one with the sufficient number of validations (and highest sequence number), and accept it as the new fully validated ledger:

$$\hat{L}' = L_x' : \; (x = i \lor x \in UNL_i) \land (|\Sigma_{L_x'}| \geq q_v). \qquad (15)$$

Lastly, in case for a given sequence number no last closed ledger meets q_v, no ledger is declared fully validated and i switches to a strategy allowing it to determine the preferred last closed ledger for the next consensus round (see (Chase and MacBrough, 2018) for an overview on this strategy).

To summarize, the three outputs depicted in Fig. 1 correspond to Eq. 3, 12 and 15 respectively.

4 XRP LEDGER PROPERTIES

4.1 Safety and Liveness

One of the fundamental problems in distributed computing is ensuring that in a system where processors exchange information, all correct processes make a decision and eventually reach a common understanding or agreement even if one or more processors have failed. There are several flavors of the consensus problem (Pease et al., 1980), but all variations are subject to conditions similar to the following:

- *Validity*: If every process begins with the same initial value v, then the final decision of a non-faulty process must be v;
- *Agreement*: Final decisions by non-faulty processors must be identical;
- *Termination*: Every non-faulty process must eventually decide.

These conditions are intended to capture two crucial properties, namely *safety* and *liveness*. The former, which derives from the conjunction of validity and agreement, traditionally guarantees that something bad will never happen. On the other hand, liveness guarantees that something good will happen eventually, and derives from the termination condition. In the context of the XRP Ledger network, the above definitions can be refined as follows:

- *Safety*: If an honest node fully validates a ledger L, then all honest nodes cannot fully validate a contradictory ledger $L' \neq L$;
- *Liveness*: If an honest node broadcasts a valid proposal P to all honest nodes, then P will eventually be accepted by all nodes and included in a fully validated ledger.

The phenomenon called *fork* is the biggest threat to the ability of a distributed ledger system to operate properly and produce results that can be relied upon. In the XRP Ledger network, the occurrence of a fork corresponds to the situation in which *two honest validators fully validate conflicting ledgers, i.e. ledgers having the same sequence number, but different identifying hash value*.

As already mentioned, the XRP LCP features a layered notion of trust. The network is divided into subsets of nodes that are collectively trusted to not collude in an attempt to defraud the other peers, and each node has complete discretion in selecting its own UNL. Since validators have influence only over nodes configured to trust them, the presence of a certain honest validator in more UNLs directly implies a higher influence by that validator on the process of determining the next ledger state. As a result, protocol safety relies on the fact that the majority of the validators act honestly and it is strongly dependent on the degree of intersection between the UNLs of every pair of nodes. The original Ripple whitepaper (Schwartz et al., 2014) provided an initial analysis of the overlap condition required to ensure consistency, coming to the conclusion that the needed minimum overlap was roughly $1/5$ of the UNL size. Later, an independent analysis (Armknecht et al., 2015) proved that the above condition was insufficient and suggested that the correct bound was $> 2/5$ of the UNL size. To be precise, they showed that in order to ensure the absence of any fork, the intersection set size between the UNL of any two validators needs to be $> 40\%$ of the maximum UNL size. A recent work by Chase and MacBrough (2018) provided a further analysis of the XRP Ledger Consensus Protocol and also derived new conditions for its safety and liveness, changing the expectation for the overlap requirement. In the remainder of this section, we show the major findings of Chase and MacBrough which will be useful in the subsequent section. Prior to this, we present the model used to analyze the safety and liveness of the protocol.

The network is modeled as if an adversary is in full control of at most t_i nodes in UNL_i for any validator i. The nodes controlled by the adversary are called Byzantine and can deviate from the protocol by performing at least one of the following actions: *(i)* not responding to messages; *(ii)* sending incorrect messages; *(iii)* sending differentiated messages to different nodes. On the contrary, we consider as honest any node that follows exactly the prescribed XRP LCP. The honest proportion is equal to $n_i - t_i$. Due to the FLP Impossibility Result (Fischer et al., 1985), safety and liveness cannot be simultaneously guaranteed by any consensus algorithm in the presence of arbitrary asynchrony and Byzantine nodes. In (Chase and MacBrough, 2018), the authors assumed a weak form of asynchrony in order to prove that the system is able to not fall in a state where some honest nodes can never fully validate a new ledger. This last assumption, however, does not seem to be sufficient to guarantee that the system cannot get stuck even in networks where two UNLs disagree only by few nodes. In this regard, the paper showed an example where even with 99% UNL overlap and no Byzantine faults, the system fails to successfully apply the preferred branch strategy, consequently maintaining a ledger chain with two distinct branches (in other words, the nodes are unable to determine a preferred chain of ledgers to converge on). It follows that, actually, only in the restricted case in which the network consists of a single agreed-upon UNL with an arbitrary number of extra nodes, the XRP LCP cannot get stuck.

The most relevant contribution by Chase and MacBrough was the re-analysis of the overlap condition required to ensure safety. According to them, the XRP LCP guarantees fork safety if for every pair of nodes i, j the following holds:

$$O_{i,j} > \frac{n_j}{2} + n_i - q_v + t_{i,j} \qquad (16)$$

where $O_{i,j} = |UNL_i \cap UNL_j|$, $q_v = k \cdot n_i$ (cf. Eq. 14) and $t_{i,j} = min\{t_i, t_j, O_{i,j}\}$ is the maximum number of allowed Byzantine faults in $UNL_i \cap UNL_j$ – here we have slightly changed the original notation in order to adapt it to the one used in our formalization. Assuming 80% validation quorum (q_v) (as prescribed in the actual implementation) and 20% faults ($t_{i,j}$), the condition in essence requires roughly $> 90\%$ UNL agreement.

4.2 Analysis

In order to provide a more accurate view of the security provisions of the XRP LCP in terms of safety and liveness, here we make some observations about the three key parameters of the protocol, i.e. UNL overlap, validation quorum and tolerated Byzantine nodes, considering also the discussion recently appeared on the Ripple GitHub repository (Wilson, 2018). We investigate the values $O_{i,j}$, q_v and $t_{i,j}$ displayed in Eq. 16 and in the following we show how safety and liveness tolerances are related to each other and vary according to the assumptions made on the above parameters.

To this end, we find it convenient to introduce two additional parameters, namely μ_i and μ_j, which denote the size of the set of surplus nodes for UNL_i

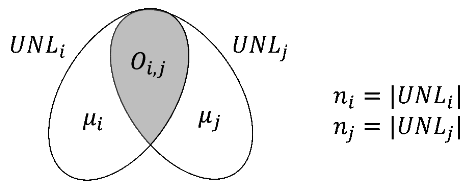

Figure 2: μ_i and μ_j are the set sizes of surplus nodes in UNL_i and UNL_j.

and UNL_j, respectively, that is the nodes in UNL_i that are not in common with UNL_j and viceversa: $\mu_i = n_i - O_{i,j}$, $\mu_j = n_j - O_{i,j}$ (Fig. 2). Moreover, we denote by t_s the safety fault tolerance and by t_l the liveness tolerance of the system, i.e. the maximum number of Byzantine nodes the XRP LCP tolerates in order to guarantee safety and liveness, respectively.

Given a pair of nodes i, j with their respective UNLs, and assuming that $n_i < n_j$ and both UNLs have the same percentage of faulty nodes, $t_{i,j} = min\{t_i, t_j, O_{i,j}\} = t_i$. By substituting $O_{i,j}$ with $n_i - \mu_i$ in Eq. 16, we get the following inequality:

$$t_s < -\frac{n_i}{2} - \frac{\mu_i}{2} - \frac{\mu_j}{2} + q_v. \tag{17}$$

On the other hand, the liveness tolerance t_l is given by:

$$t_l < n_i - q_v. \tag{18}$$

In general, a consensus protocol providing results that can be relied upon is preferable, rather than one that is able to progress in the presence of faulty nodes but, at the same time, reports impaired results that could undermine consistency. As a result, in order to obtain a validation quorum q_v whose safety fault tolerance t_s meets or exceeds the liveness tolerance t_l, we can use $t_s \geq n_i - q_v$ and get the following:

$$n_i - q_v < -\frac{n_i}{2} - \frac{\mu_i}{2} - \frac{\mu_j}{2} + q_v$$

$$q_v > \frac{3n_i}{4} + \frac{\mu_i}{4} + \frac{\mu_j}{4}. \tag{19}$$

4.2.1 Unique UNL

Let us now consider the simplest case where the XRP Ledger network consists of a single agreed-upon UNL. In this case, $n_i = n_j = n$, and, consequently, $\mu_i = \mu_j = 0$. By Eq. 19, we obtain $q_v > (3/4)n$, meaning that when there is 100% agreement on participants, reaching a 75% validation quorum is sufficient to fully validate a ledger. Therefore, in Eq. 16 we have $n > n/2 + n - (3/4)n + t_i$, from which we obtain $t_s (= t_i) < (1/4)n$, whereas from Eq. 18 we obtain $t_l < n – (3/4)n = (1/4)n$. As a result, in case $UNL_i = UNL_j$ and $q_v = 0.75n$, both the tolerances are $0.25n$. This means that when less than 25% of trusted nodes are Byzantine, the protocol functions properly.

Keeping valid the assumption $UNL_i = UNL_j$, now we show how t_s and t_l vary when q_v is equal to $0.8n$, as required by the current XRP LCP specification. In this case, the value of the two tolerances are no longer equal: $t_s < (3/10)n$, and $t_l < (1/5)n$. Compared to the previous case in which $q_v = 0.75n$, here the liveness tolerance is lower than the safety fault tolerance, and this implies that the system ability to not get stuck is slightly weakened.

4.2.2 Overlapping UNLs

So far the analysis has focused on the circumstance where the network is composed of a unique UNL. However, the validation quorum increase takes on greater significance in the context in which the UNLs of any two nodes i, j do not completely overlap, i.e. when at least one of the two parameters μ_i and μ_j is > 0. In this regard, we now turn our attention to the total non-overlapping component resulting from the sum of the two individual surpluses μ_i and μ_j of Eq. 17:

$$\mu_i + \mu_j < -n + 2q_v - 2t_s. \tag{20}$$

Let us now consider two cases in which we set the validation quorum first to 80% and then to 90% of the nodes. In the former case, $q_v = 0.8n$ and assuming $t_s = 0.2n$, we can safely allow up to a non-overlap of $0.2n$. In the latter case, $q_v = 0.9n$ and assuming $t_s = 0.1n$, the maximum non-overlap we can safely allow is $0.6n$. Thus, as the above bounds show, as q_v increases, the degree of freedom of any node in the choice of validators to trust, in turn, increases.

To conclude, if we want the system to have a little more liveness, we can increase the sizes of the surplus node sets μ_i and μ_j. Recalling that t_l depends on the quantity $n_i - q_v$, as the values μ_i and μ_j increase, we obtain a higher validation quorum (cf. Eq. 19), and hence the maximum number of allowed Byzantine nodes to guarantee liveness increases. Let us consider an overlap of 90%, i.e. a non-overlap of 10%. If the safety fault tolerance is $0.2n_i$, from Eq. 16 we obtain $q_v > 0.75n_i$, and $t_l < 0.25n_i$. In contrast, if the safety fault tolerance is $0.25n_i$, we obtain $q_v > 0.8n_i$, and hence, $t_l < 0.2n_i$. From this analysis, it emerges that safety and liveness tolerances, validation quorum, and UNLs overlapping set size are strictly correlated, and it is possible to tune these parameters according to the desired properties the system needs to satisfy.

Currently, if no configuration changes are made, each node adopts the default and recommended UNL provided by Ripple. This essentially implies that no disagreement on the participants in the network is allowed, since all the nodes listen to a single list of validators. Accordingly, the XRP LCP is really able to guarantee that the network cannot get stuck as long as the number of Byzantine nodes within the system is limited.

5 CONCLUSIONS

Understanding the fundamental mechanisms and security properties of the structure underlying blockchain-based protocols is becoming increasingly important. Thus, detailed analyses and formalizations of existing distributed ledger systems are crucial to prove the correctness of the algorithms and gain confidence that they achieve their goals.

In this paper, we have presented a formalization of the XRP Ledger Consensus Protocol, whose functioning has been studied in a rather superficial way so far. To the best of our knowledge, our work is the first one that describes the phases of the Ripple protocol in such great detail, trying to avoid ambiguities in defining its behavior. We have included the analysis of two key security properties, namely safety and liveness, based upon which the efficacy of the protocol can be determined. Furthermore, we have shown that the correlation between some protocol parameters can be leveraged to meet a desired liveness/fault tolerance. Our work represents the first step towards the complete analysis of the XRP Ledger Consensus Protocol and opens several research directions. For instance, our set-theoretic approach could serve as the basis for the construction of an extended formalism that is capable of capturing the behaviour of the protocol by means of a strictly formal language. Also the use of different model-driven approaches and automated verification tools can greatly benefit from our specification efforts and facilitate the comprehensive description of the protocol. For the immediate future, we plan to extend our work considering the formalization of additional security properties and using different models of computation.

REFERENCES

Armknecht, F., Karame, G. O., Mandal, A., Youssef, F., and Zenner, E. (2015). Ripple: Overview and Outlook. In Conti, M., Schunter, M., and Askoxylakis, I., editors, *Trust and Trustworthy Computing*, pages 163–180, Cham. Springer International Publishing.

Badertscher, C., Maurer, U., Tschudi, D., and Zikas, V. (2017). Bitcoin as a Transaction Ledger: A Composable Treatment. In Katz, J. and Shacham, H., editors, *Advances in Cryptology – CRYPTO 2017*, pages 324–356, Cham. Springer International Publishing.

Bano, S., Sonnino, A., Al-Bassam, M., Azouvi, S., McCorry, P., Meiklejohn, S., and Danezis, G. (2017). Consensus in the Age of Blockchains. *CoRR*, abs/1711.03936.

Cachin, C. and Vukolic, M. (2017). Blockchain Consensus Protocols in the Wild. *CoRR*, abs/1707.01873.

Chase, B. and MacBrough, E. (2018). Analysis of the XRP Ledger Consensus Protocol. *CoRR*, abs/1802.07242.

Daian, P., Pass, R., and Shi, E. (2016). Snow White: Provably Secure Proofs of Stake. Cryptology ePrint Archive, Report 2016/919.

Fischer, M. J., Lynch, N. A., and Paterson, M. S. (1985). Impossibility of Distributed Consensus with One Faulty Process. *J. ACM*, 32(2):374–382.

Garay, J., Kiayias, A., and Leonardos, N. (2015). The Bitcoin Backbone Protocol: Analysis and Applications. In Oswald, E. and Fischlin, M., editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 281–310, Berlin, Heidelberg. Springer Berlin Heidelberg.

Garay, J., Kiayias, A., and Leonardos, N. (2017). The Bitcoin Backbone Protocol with Chains of Variable Difficulty. In Katz, J. and Shacham, H., editors, *Advances in Cryptology – CRYPTO 2017*, pages 291–323, Cham. Springer International Publishing.

Garay, J. A., Kiayias, A., Leonardos, N., and Panagiotakos, G. (2018). Bootstrapping the Blockchain, with Applications to Consensus and Fast PKI Setup. In Abdalla, M. and Dahab, R., editors, *Public-Key Cryptography – PKC 2018*, pages 465–495, Cham. Springer International Publishing.

Gramoli, V. (2017). From Blockchain Consensus Back to Byzantine Consensus. *Future Generation Computer Systems*.

Halpin, H. and Piekarska, M. (2017). Introduction to Security and Privacy on the Blockchain. EuroS&P 2017 - 2nd IEEE European Symposium on Security and Privacy, Workshops.

Kiayias, A., Russell, A., David, B., and Oliynykov, R. (2017). Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. In Katz, J. and Shacham, H., editors, *Advances in Cryptology – CRYPTO 2017*, pages 357–388, Cham. Springer International Publishing.

Mauri, L., Cimato, S., and Damiani, E. (2018). A Comparative Analysis of Current Cryptocurrencies. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy - Volume 1: ICISSP*, pages 127–138. INSTICC, SciTePress.

Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System.

Pass, R., Seeman, L., and Shelat, A. (2017). Analysis of the Blockchain Protocol in Asynchronous Networks. In Coron, J.-S. and Nielsen, J. B., editors, *Advances in Cryptology – EUROCRYPT 2017*, pages 643–673, Cham. Springer International Publishing.

Pass, R. and Shi, E. (2017). The Sleepy Model of Consensus. In Takagi, T. and Peyrin, T., editors, *Advances in Cryptology – ASIACRYPT 2017*, pages 380–409, Cham. Springer International Publishing.

Pease, M., Shostak, R., and Lamport, L. (1980). Reaching Agreement in the Presence of Faults. *J. ACM*, 27(2):228–234.

Rawat, D. B., Chaudhary, V., and Doku, R. (2019). Blockchain: Emerging Applications and Use Cases. *CoRR*, abs/1904.12247.

Ripple Labs Inc. (a). Ripple. https://ripple.com/. Last checked on Oct, 2019.

Ripple Labs Inc. (b). XRP Ledger Dev Portal. https://xrpl.org/index.html. Last checked on Oct, 2019.

Ripple Labs Inc. (c). Ripple Source, GitHub repository. https://github.com/ripple/rippled/tree/develop/src/ripple. Last checked on Oct, 2019.

Saad, M., Spaulding, J., Njilla, L., Kamhoua, C. A., Shetty, S., Nyang, D., and Mohaisen, A. (2019). Exploring the Attack Surface of Blockchain: A Systematic Overview. *CoRR*, abs/1904.03487.

Schwartz, D., Youngs, N., and Britto, A. (2014). The Ripple Protocol Consensus Algorithm. *Ripple Labs Inc. White Paper*.

Vukolić, M. (2016). The Quest for Scalable Blockchain Fabric: Proof-of-work vs. BFT Replication. In Camenisch, J. and Kesdoğan, D., editors, *Open Problems in Network Security*, pages 112–125, Cham. Springer International Publishing.

Wang, W., Hoang, D. T., Xiong, Z., Niyato, D., Wang, P., Hu, P., and Wen, Y. (2018). A Survey on Consensus Mechanisms and Mining Management in Blockchain networks. *CoRR*, abs/1805.02707.

Wilson, B. (2018). Raise quorum / increase fault tolerance. https://github.com/ripple/rippled/issues/2604. Last checked on Oct, 2019.

Xiao, Y., Zhang, N., Lou, W., and Hou, Y. T. (2019). A Survey of Distributed Consensus Protocols for Blockchain Networks. *CoRR*, abs/1904.04098.