xx network

White Paper
xx consensus

December, 2019

The information provided in this white paper pertaining to xx network SEZC ("Praxxis" or the "Company"), the xx Coin (the “Coins”), its business assets, strategy and operations is for general informational purposes only and is not a formal offer to sell or a solicitation of an offer to buy any Coins, securities, options, futures, or other derivatives related to securities in any jurisdiction and its content is not prescribed by securities laws. Information contained in this white paper should not be relied upon as advice to buy or sell or hold Coins or securities or as an offer to sell Coins. This presentation does not take into account nor does it provide any tax, legal or investment advice or opinion regarding the specific investment objectives or financial situation of any person. While the information in this presentation is believed to be accurate and reliable, Praxxis and its agents, advisors, directors, officers, employees and shareholders make no representation or warranties, expressed or implied, as to the accuracy of such information and Praxxis expressly disclaims any and all liability that may be based on such information or errors or omissions thereof. Praxxis reserves the right to amend or replace the information contained herein, in part or entirely, at any time, and undertakes no obligation to provide the recipient with access to the amended information or to notify the recipient thereof.

Neither we nor any of our representatives shall have any liability whatsoever, under contract, tort, trust or otherwise, to you or any person resulting from the use of the information in this presentation by you or any of your representatives or for omissions from the information in this presentation. Additionally, the Company undertakes no obligation to comment on the expectations of, or statements made by, third parties in respect of the matters discussed in this presentation.

This whitepaper contains forward looking statements, including among other things, statements concerning the distribution of xx Coins, and other statements identified by words such as “could,” “expects,” “intends,” “may,” “plans,” “potential,” “should,” “will,” “would,” or similar expressions and the negatives of those terms. Forward-looking statements are not promises or guarantees of future performance, and are subject to a variety of risks and uncertainties, many of which are beyond our control. Actual results could differ materially from those anticipated in such forward-looking statements as a result of various risks and uncertainties, which include, without limitation, market risks and uncertainties and the satisfaction of losing conditions for a distribution of xx Coins. Forward-looking statements speak only as of the date hereof, and, except as required by law, Praxxis undertakes no obligation to update or revise these forward-looking statements.

Contents

1 INTRODUCTION

Since the introduction of Bitcoin [16], many have attempted to develop a decentralized platform capable of providing a global-scale payments system outside of the control of the institutions of society. However, these platforms are burdened by large fees, slow payment confirmation, and high energy consumption, preventing the necessary user adoption to fully replace centralized systems. No current blockchain platform simultaneously achieves decentralization, speed, privacy, and long-term security against threats such as quantum computers.

The Praxxis Technical Paper introduces, the **xx blockchain**, a new approach providing an unprecedented combination of **speed**, **security** and **scalability**. The core component of a decentralized blockchain is the consensus algorithm that facilitates agreement between nodes in the network. The xx blockchain relies on a novel **quantum-secure** consensus algorithm, **xxBFT** (also referred to as **xx consensus**), that achieves linear authenticator complexity and single block finality. xxBFT is scalable to thousands of nodes while maintaining low block latency, high performance and constant-sized proofs of finality. xxBFT operates with**low energy consumption** and is truly **egalitarian**, granting every node equal operating time and rewards.

The xx blockchain, supported by the xxBFT consensus algorithm, sets out to create the first quantum-secure and truly decentralized platform providing a global-scale payments system and restoring the balance of power between large organizations and digital citizens.

In this technical paper, we start by setting the goals and assumptions for the xx blockchain, and present its key features and novelties. We will then describe, in detail, the operation of the xxBFT consensus algorithm, followed by an examination of its properties. This will include a brief description of how to securely initialize the xx blockchain network. Finally, we offer an overview of related work in the blockchain space, and summarize the key innovations presented in this document.

2 GOALS & ASSUMPTIONS

The xxBFT consensus algorithm aims to achieve linear authenticator complexity, while guaranteeing safety under asynchrony and liveness under partial synchrony. Furthermore, xxBFT should provide single block finality and achieve optimistic responsiveness during optimal network conditions. xxBFT and the xx blockchain have been designed to achieve the following goals:

- ***Safety Goal***: No two correct nodes commit to different blocks for the same round.
- ***Liveness Goal***: All correct nodes eventually commit a block that updates the ledger.
- ***Validity Goal***: The probability that a correct node commits a block containing an invalid transaction is negligible.
- ***Optimistic Responsiveness Goal***: A correct leader, under perfect network synchrony, must be able to drive consensus to a decision as fast as allowed by actual message delays.

ASSUMPTIONS

We assume that all the nodes in the network have authenticated channels, which were previously established in a quantum-secure manner during network initialization. Furthermore, nodes don't rely on any standard public key cryptographic primitives which are vulnerable to quantum computing. In pursuit of this, all signatures are hash-based and all communications are encrypted using symmetric ciphers.

The network operates under the standard Byzantine Fault Tolerant (BFT) assumption, where byzantine nodes may fail arbitrarily or behave maliciously, trying to subvert the network. We assume that the number of nodes exhibiting byzantine behavior is at most $1/3$ of the network. In distributed systems literature this assumption is expressed as: in order to tolerate f byzantine nodes, the size of the network must be at least $n = 3f + 1$ [17].

To model the network we use the *partial synchrony model* as described in [7], which states that there is an unknown Global Stabilization Time (GST), after which two honest nodes are able to communicate in a known bounded time. Moreover, we assume that the network may fail to deliver messages, delay them, duplicate them, or even deliver them out of order.

ADVERSARIAL MODEL

We assume a global adversary capable of simultaneously coordinating all malicious nodes in the network. The adversary can isolate honest nodes from the rest of the network, provided that the BFT assumption is not violated. This adversary can also view the state of every honest node at any time and can instantly modify the state of all adversarial nodes accordingly. The goal of the adversary is to utilize all the capabilities at its disposal to cause the most damage to the decentralized platform.

The adversary is able to eavesdrop, forward, and delete messages between honest nodes. However, due to the use of authenticated channels, the adversary is not able to modify, replay, or inject new messages, without detection. Furthermore, the adversary is computationally bounded, but has access to quantum computing capabilities and can break all the cryptographic primitives that rely on the hidden subgroup problem, such as the (Elliptic Curve) Discrete Logarithm Problem and the factorization of large primes. Finally, the adversary is unable to subvert symmetric primitives as these are easily adaptable to resist quantum attacks.

3 OVERVIEW

We propose a novel ”bi-stable” leader-based BFT consensus framework facilitating a binary decision using ”optimistic path” and ”fallback” procedures. When the network is strongly synchronous, an ”optimistic path” aims to achieve responsive agreement on a correct leader's block proposal while guaranteeing safety. However, when byzantine behavior causes the network to start losing synchrony, consensus will rely on a ”fallback” mechanism, which maintains safety under asynchrony but guarantees liveness under partial synchrony, after GST.

This framework is based on five primary approaches that enable xxBFT to achieve its primary performance goals while remaining quantum-secure:

NETWORK INITIALIZATION

Network initialization is a process through which the initial nodes establish symmetric keys with each other, collectively generate an unmanipulatable random seed, and sign the genesis block. We propose one novel mechanism to achieve this using a physical, but virtually auditable, event. Prior to this event, nodes are required to commit to a tree of hash values, publishing the images of each value but keeping the preimages private.

Each node is responsible for bringing a number of these preimages to the event. Some of these preimages are shared with other nodes in attendance in order to generate symmetric keys used for quantum-secure authenticated communication channels. Another set of these preimages are combined with preimages from all other attendees to collectively generate an unmanipulatable random value that will be used to seed the randomness generated for every subsequent block. Finally, the attending nodes all sign the genesis block of the network, locking in the random seed and the initial network membership and coin tree.

COMMITTED RANDOMNESS

Committed randomness is a mechanism to generate unmanipulatable randomness every block using a publicly verifiable random value that is published by the block producer (BP). This published random value is hashed with the chain of prior randoms extending back to the random seed generated during Network initialization. As a result, a new network random is contained on every block and can be used to seed the selection algorithms, for example to choose the BP and execute endorser sampling for each round of consensus.

Since the randoms have been committed to in advance, the value revealed by the BP cannot be modified because it is verifiable by all nodes in the network, and would be rejected otherwise. Therefore, a malicious BP cannot influence the network random in order to gain any advantage, for example by finding a value that always selects malicious BPs. Hence, the network random is an unmanipulatable value and can be safely used to prevent any manipulation of the selection algorithms.

We refer to the network random as being probabilistically unpredictable. This property results from the chaining of BP random reveals, and it prevents an organized attack on the BP or endorsers of subsequent consensus rounds. The current BP will always know the upcoming leader before the rest of the network, allowing a malicious leader to know ahead of time if the next BP is also malicious. If this happens, the adversary can predict scheduling two rounds in advance, instead of one. Given the BFT assumption, the probability of a malicious node becoming the BP is at most $\frac{1}{3}$. This way, the probability of n consecutive rounds with a malicious leader is given by $\frac{1}{3}^n$, which is an exponential decrease. This probability distribution ensures that an honest node is highly likely to be selected as the BP every few rounds, releasing a random unknown to the adversary and re-establishing the unpredictability of the network random.

ENDORSER SAMPLING

Endorser sampling is an algorithm that selects a constant-sized, randomly chosen subset of nodes from the network to act as endorsers for each round of consensus. These endorsers are the only nodes in the network responsible for validating and endorsing the block of transactions. This provides a major advantage when compared to classical BFT consensus algorithms: most nodes in the network are not required to receive and validate transactions, allowing them to preserve bandwidth and computational power to execute tasks other than consensus.

The selection of the endorser sample can be subject to attacks if not carefully designed. For example, if the selection algorithm depends on some value contained in a block, a malicious leader can influence the selection of the sampled nodes by ”mining” the block until a favorable sample containing an overwhelming number of malicious nodes is found. If this happens, the malicious endorsers can try to subvert the network by sending out fake endorsements.

In a decentralized network, the endorser sampling approach can only be made safe if the algorithm that selects the endorsers is truly unpredictable and unmanipulatable. In xxBFT this may be achieved using the committed randomness generated in a prior round.

EFFICIENT FALLBACK

Anytime malicious behavior or an unreliable network disrupts a round of consensus, a fallback mechanism is triggered. While nearly all BFT consensus mechanisms have some manner of fallback, they all generally suffer from increased communication or authenticator complexity as compared to the “optimistic path”. xxBFT is distinguished in that authenticator complexity remains linear even when a fallback is triggered.

There are generally two types of fallback in xxBFT depending on how the round fails. If the block producer (BP) is unresponsive or proposes an invalid block then a fallback to produce an empty block is triggered, selecting a new BP for the next round. If the network fails to receive a quorum of endorsements from the endorser sample then a separate fallback attempts to commit the same block by selecting a new endorser sample.

QUANTUM-SECURE GROUP ENDORSEMENTS

The combination of endorser sampling and committed randomness can be used to create a non quantum-secure consensus algorithm, relying on, for example, standard elliptic curve public key cryptography and BLS signature aggregation. However, the consensus algorithm utilized in the xx blockchain must be quantum-secure. There are different types of quantum-secure signature schemes, for example, lattice-based, code-based, super-singular isogenies and hash-based. Of these, we believe that hash-based signatures, such as the Winternitz OTS$^+$ scheme [11], provide properties of interest to the design of the xxBFT consensus algorithm. The size of these signatures is acceptable when compared to other quantum-secure schemes, even if much larger than standard public key cryptography signatures.

However, since no signature aggregation is possible using WOTS$^+$, the size of proof of finality certificates, which are produced by xxBFT to prove agreement on a block, would still be very large. This is a problem for users that need to receive those proofs using mobile devices. The xxBFT algorithm must then be able to produce compact proofs of block finality.

Hence, we introduce a hash-based proof designed to allow xxBFT to issue a compact proof of finality containing the results of a consensus round. In our scheme, nodes issue compact one-time signatures that can be probabilistically verified and, as a group, provide a final quantum-secure state.

The scheme is parameterized by the number of nodes that are required to participate and the desired security level. Based on the parameters, nodes create a series of separate random bit strings to sign a specific number of bytes and each corresponding checksum. Unlike the WOTS+ scheme, each individual byte has a corresponding checksum ladder.

An important distinguishing factor of our construction is that each node in the set signs a sequence of bytes

tied to their own secret key elements, as opposed to having the entire set sign different parts of one same hash. Once they have done so, other nodes in the network can check the collective proof.

4 xxBFT

We propose xxBFT, the first quantum-secure BFT consensus algorithm with linear authenticator complexity in block producing rounds. We start by describing how xxBFT combines the three previously presented novelties in order to build an optimized optimistic path. We then outline the fallback procedure, which allows consensus to operate correctly even if the optimistic path fails to achieve agreement.

OPTIMISTIC PATH

The optimistic path leverages the unmanipulatable selection of an endorser sample in order to quickly and safely achieve network agreement on the block proposal. Furthermore, it utilizes quantum-secure group endorsements to provide compact proofs of block finality. It consists of four phases: *propose*, *validate*, *confirm* and *commit*, as shown in Figure 3.

▷ PROPOSE

Every $block$ is first constructed by the BP who is chosen randomly and unmanipulatably by a prior leader's committed random. The BP verifies all the transactions and their signatures, organizes the transactions into a Merkle tree, and then constructs and signs both a $transactionBatch$ and a smaller $stateUpdate$. The $stateUpdate$ verifiably contains all the same transactions as the $transactionBatch$ but, to preserve network bandwidth, does not include the large quantum-secure signatures of each transaction. The $block$ is composed by: the $blockHeader$, the committed random, and the Merkle roots of the $transactionBatch$ and $stateUpdate$. The $blockHeader$ will simply contain the block number, the hash of the previous block, and the new network random. Figure 1 shows the structure of the $block$.

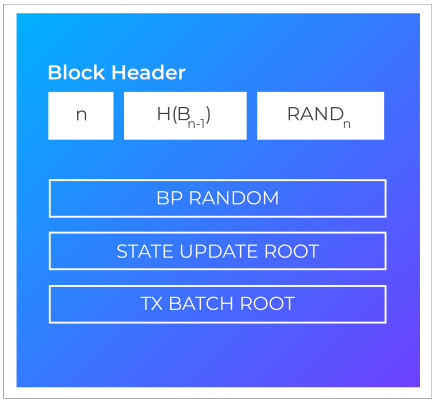

Figure 1: Block Structure

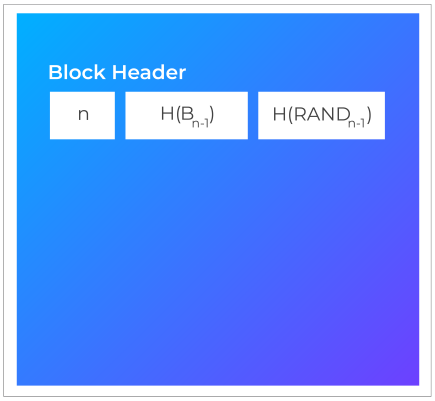

Figure 2: Empty Block Structure

The BP gossips the block proposal to the network, consisting of the $block$ and the $stateUpdate$. The network random contained in the $block$ is an accumulation of all the randomness that has been revealed in the lifetime of the network. One simple way of performing this accumulation is to hash the current BP's committed random with the previous round's network random. Each node in the network verifies the BP's committed random and network random contained in the $block$, allowing them to deterministically compute the endorser set, containing a constant number of nodes, say 100. In parallel, the BP gossips the $transactionBatch$ only to the endorsers.

▷ VALIDATE

The selected endorsers know about their role upon receiving the block proposal from the BP. Then, after receiving the $transactionBatch$, endorsers validate all the transactions against the current state of the ledger. If all are valid, endorsers sign the $block$ hash using a WOTS+ signature and broadcast it to the network. Otherwise, if a single bad transaction is detected, the BP is deemed malicious and endorsers can broadcast the transaction and a Merkle proof to the network. Together with the signed $block$ by the BP, this information is sufficient to prove to any node that the BP acted maliciously. This allows nodes to

abandon the optimistic path and start operating on the fallback procedure.

▷ CONFIRM

All nodes in the network wait to receive signatures from a threshold of the endorsers, say 65%, allowing them to form a Quorum Certificate (QC). This is a data structure that expresses the combined opinion of the endorser set. An $endorsementQC$ guarantees, with overwhelming probability, that all the transactions are valid. Once a node receives an $endorsementQC$ it can be confident that the block is valid, without needing to verify any transactions. The node then creates a $confirm$ signature and gossips it back to the endorsers, letting them know that it has seen an $endorsementQC$.

All endorsers wait to receive a $2f + 1$ network majority quorum of $confirm$ signatures on the block proposal, forming a $confirmQC$. This allows endorsers to be certain that no forks will occur and that the network is ready to commit the block. Endorsers will then broadcast to the network a compact $commit$ endorsement on the $block$ hash, which certifies that it should be committed.

▷ COMMIT

Each $commit$ endorsement on its own is insecure, but a threshold of them certifying the same $block$ hash, form an unforgeable, quantum-secure group endorsement. Once any node in the network has received this threshold of $commit$ endorsements, it can gather them into a $commitQC$ and the $block$ is considered to have reached finality. Using this novel cryptographic structure, xxBFT can provide compact, constant-sized and quantum-secure proofs of block finality.

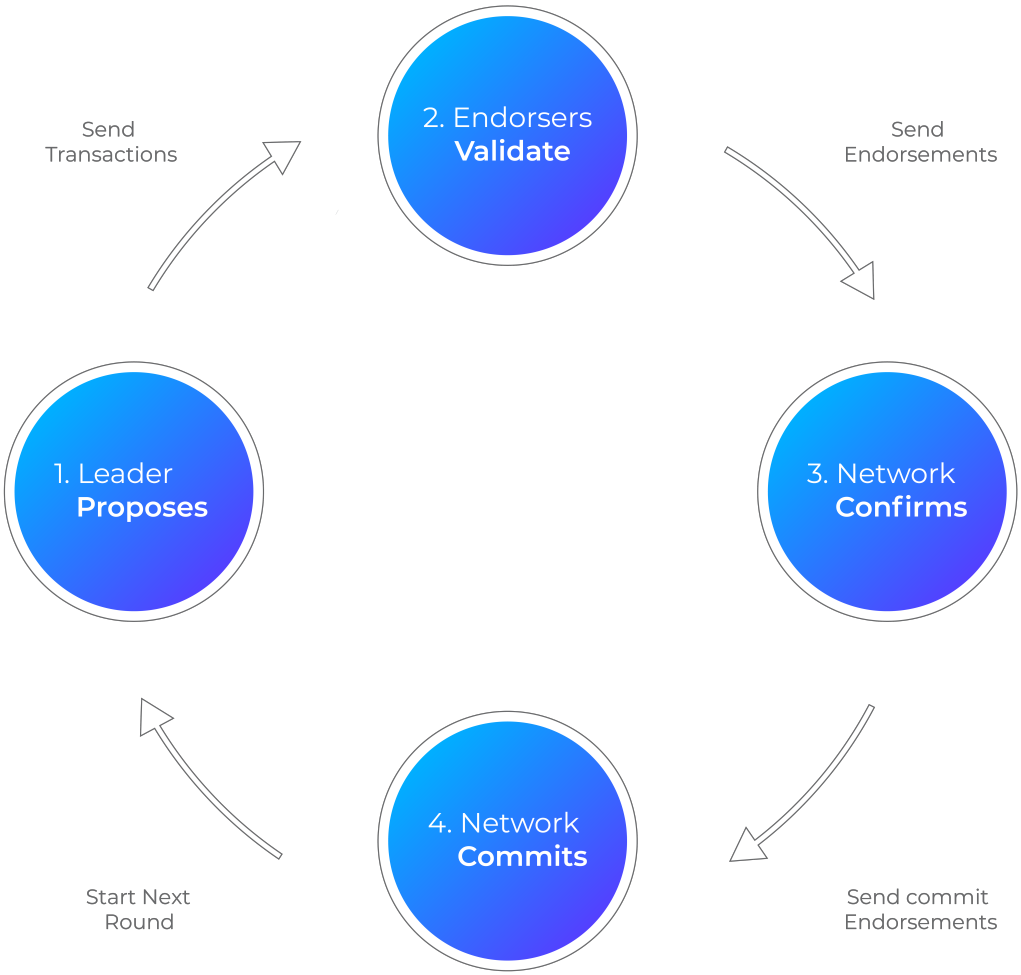

Figure 3: Optimistic Path State Machine Diagram

FALLBACK

When the network starts experiencing synchrony issues, the optimistic path might get stuck and nodes running xxBFT need to activate the fallback procedure. The transition from optimism to pessimism needs to be carefully designed in order to maintain safety. Furthermore, this transition mechanism should accommodate a variety of scenarios that can cause the optimistic path to halt at the different phases of the algorithm. Without loss of generality we assume that the transition mechanism is part of the fallback procedure. While the optimistic path can only decide on a block proposal, depending on the situation, the fallback might lead to an empty block instead.

▷ EMPTY BLOCK

The empty block is a different type of block than the one proposed by a BP during the regular operation of consensus. It contains no information but still has a block header, which includes a fresh network random that is computed by hashing the previous network random. This way, the empty block is deterministic, which allows the utilization of a leader-less algorithm, simplifying the overall design of the fallback mechanism. The empty block structure can be seen in Figure 2.

The fallback procedure of xxBFT is divided into two paths, one that results in an empty block, and another that commits a block proposed by the leader of the consensus round. We call these the empty block path and non-optimistic block path. Nodes can enter the fallback mechanism at either path, depending on the situation. Both paths are designed in the same way, using two phases of endorsement, where the whole network is required to participate in order to reach a decision. The two phases draw from classical BFT algorithms, where nodes start by gossiping $prepare$ signatures to each other. Once a node receives a quorum of $\frac{2}{3} + 1$ prepares from the network, it will form a $prepareQC$. This signifies that a majority of the network is ready to agree on the block, allowing the node to broadcast a $commit$ signature. This is an actual full quantum-secure signature, as opposed to the $commit$ endorsement sent by endorsers in the optimistic path. Finally, when a node receives $\frac{2}{3} + 1$ commits from the network, it forms a $commitQC$ and commits either the proposed block or an empty block, depending on the path it has followed.

While waiting for a $prepareQC$ in the empty block path, nodes can ”change their mind” if they receive the block proposal, send out a $prepare$ signature on the block, and then wait for a $prepareQC$ certifying the block proposal. This connection between both paths of the fallback procedure might improve latency in some scenarios. For example, if $\frac{2}{3}$ of the network is waiting for the last signature needed to form a $prepareQC$ for the block proposal, a node that switches paths allows the network to finally make progress.

Conversely, if a majority of the network is already prepared to commit an empty block, nodes that are waiting for a $prepareQC$ on the block proposal can move back to the empty block path if they receive a $prepareQC$ for the empty block. However, this case is slightly different, since the node can broadcast a $commit$ signature for the empty block. In some cases, this signature might be the last needed to have the network form a $commitQC$, reaching a final agreement on the empty block.

We now analyze the different scenarios where the fallback procedure is activated, which can happen during three phases of the optimistic path: *propose*, *confirm* and *commit*.

▷ BYZANTINE BLOCK PRODUCER

We start by examining the situation where the optimistic path gets stuck during the *propose* phase. This might happen if the BP is malicious and is trying to delay consensus, or if an honest leader is offline or faulty. In these situations we want to move to the next leader as soon as possible while ensuring that a block is still produced. To address this, all nodes have a timeout that is triggered when no block proposal is received from the current BP in the required time. When this happens, the node enters the fallback procedure via the empty block path. Eventually, $\frac{2}{3} + 1$ of the network will timeout, allowing consensus to be reached on the empty block for the current round.

▷ BYZANTINE ENDORSERS

In the second situation we examine a possible optimistic path deadlock during the *confirm* phase. This can happen if the selected endorser sample contains a large number of malicious nodes, which refuse to send out their endorsements, and an $endorsementQC$ is never reached. Another timeout is needed to handle this case, with a longer wait time than the first one. As in the first scenario, when this timeout is triggered, nodes enter the fallback procedure via the empty block path.

However, malicious endorsers can endorse the block, but then refuse to broadcast their commit endorsements. This means that the optimistic path can also get stuck during the *commit* phase. Thus, a third timeout is needed to make progress, with an even longer wait time than the previous two. In this case, once the timeout is triggered, the node enters the fallback procedure via the non-optimistic block path. This is possible since some nodes in the network have an $endorsementQC$, which is sufficient to know that the block proposal is valid and all efforts should be made to commit this block.

One could argue that this third timeout is necessary in the case the described situation happens, but in practice never occurs. If during the *confirm* phase an adversary controls all the malicious endorsers and has the possibility to make the optimistic path fail, then it will never pass on this opportunity and let the *commit* phase fail instead. The block committed by the fallback procedure and the optimistic path would be the same. Hence, the adversary doesn't gain anything by simply delaying the decision on that block, while he can instead lead consensus to commit an empty block, causing the most damage to the network.

▷ PARTITION ATTACK

Now, we analyze the impact of a large scale partition attack, executed by an extremely sophisticated adversary that controls up to $1/3$ of the nodes in the network. By allowing only one of the partitions to receive a block proposal, a second partition will timeout and fallback to the empty block path. This can cause a consensus deadlock where the two sets of nodes are waiting for $commitQC$s for both an empty block and a regular block. If a timeout were added to these waiting states then it would be possible for this adversary to force both an empty block and a regular block to be committed, effectively breaking safety. This way an alternative solution is required.

▷ BYZANTINE RESYNC PROCEDURE

According to the partial synchrony assumption, the partition attack has to eventually end, allowing the network to restore synchrony. Once this happens, nodes will be stuck in the last phases of the two different paths of the fallback procedure. However, since all honest nodes can now communicate with each other, they can share all information they have about their state and realize that a partition attack must have happened. This way nodes can enter a byzantine resync procedure, which is simply a two phase full network consensus last effort to commit the block proposal of the current round. An in-depth analysis on this partition attack is detailed in Section 5 - Analysis.

▷ ALTERNATIVE FALLBACK

We introduce an alternative approach to the fallback mechanism which achieves linear authenticator complexity based on a second endorser sample. In the scenario where the optimistic path fails, a second randomly selected endorser set is chosen to carry out consensus in the same way as the optimistic path. This endorser set can be computed using a deterministic function or through the use of a cryptographic sortition mechanism. The latter adds an additional unpredictability layer as it implies that nodes have some type of secret key which allows them to independently calculate whether or not they have been selected to be part of the second endorser set. In contrast, the first approach using the deterministic function, could simply perform a random shuffle on the network membership list and choose the endorser set based on the shuffled list. In the event of failure in the second endorser sample, the network keeps repeating the same procedure, selecting a new endorser set until one is able to reach agreement.

5 ANALYSIS

In this section we analyze the scalability of xxBFT through the use of an endorser sample, provide the safety and liveness trade-offs of the system, and present the performance of the system against different worst case scenarios.

COMPLEXITY AND SCALABILITY

In most leader-based BFT-family consensus protocols, the performance metric of interest is authenticator complexity. This measures the number of signatures communicated between all nodes during a round of consensus. To achieve a scalable decentralized network it is of paramount importance that block finality doesn't increase significantly, otherwise performance will degrade heavily. Since each node has limited bandwidth, the overall network bandwidth increases at the same rate as the size of the network. This way, it is necessary that the authenticator complexity scales at the same or a lower rate.

The xxBFT consensus has four major communication phases, which we analyze in order to demonstrate how linear authenticator complexity is achieved, considering that the endorser set size E is kept constant, while the network $\mathcal{N}$ grows in size.

Communications Phase	Number of Communications	Size of Each Communication	Total Authenticator Complexity
BP gossips $block$ to Network	1 to $N \Rightarrow O(\mathcal{N})$	Constant = 1	$O(\mathcal{N})$
BP gossips $stateUpdate$ to Network	1 to $N \Rightarrow O(\mathcal{N})$	Bounded by $transactionBatch$ = 1	$O(\mathcal{N})$
BP gossips $transactionBatch$ to Endorsers	1 to $E \Rightarrow O(E)$	Bounded by $transactionBatch$ = 1	$O(E)$
Total Complexity			$O(\mathcal{N} + \mathcal{N} + E) = O(\mathcal{N})$

Table 1: BP gossips the $block$, $stateUpdate$, and $transactionBatch$

Communications Phase	Number of Communications	Size of Each Communication	Total Authenticator Complexity
Endorsers gossip $endorsement$ to Network	E to $\mathcal{N} \Rightarrow O(E \cdot \mathcal{N})$	Constant = 1	$O(E \cdot \mathcal{N})$
Total Complexity			$O(\mathcal{N})$

Table 2: Endorsers gossip $endorsement$ to the Network

Communications Phase	Number of Communications	Size of Each Communication	Total Authenticator Complexity
Network gossip $confirm$ to the Endorsers	E to $\mathcal{N} \Rightarrow O(E \cdot \mathcal{N})$	Constant = 1	$O(E \cdot \mathcal{N})$
Total Complexity			$O(\mathcal{N})$

Table 3: Network gossips $confirm$ to the Endorsers

Communications Phase	Number of Communications	Size of Each Communication	Total Authenticator Complexity
Endorsers gossip $commit$ to Network	$\mathcal{N}$ to $E \Rightarrow O(E \cdot \mathcal{N})$	Constant = 1	$O(E \cdot \mathcal{N})$
Total Complexity			$O(\mathcal{N})$

Table 4: Endorsers gossip $commit$ to the Network

As seen in Table 1, 2, 3 and 4, endorser nodes send a constant number of signatures, but are the only ones that receive signatures from the whole network. This means that the authenticator complexity of endorsers is linear. On another hand, since the endorser set size is constant, all other nodes in the network only need to receive and send a constant number of signatures, achieving constant authenticator complexity. When analysing complexity, the worst case always prevails, resulting in theoretical linear authenticator complexity for the optimistic path of xxBFT. However, in practical terms, the constant authenticator complexity of the majority of the network results in good scalability. This suggests that xxBFT scales sub-linearly if the nodes, when selected to be endorsers, have the bandwidth necessary to support receiving signatures from the whole network. Theoretical simulations have supported this claim showing that increasing the size of the network from 100 to $100,000$ nodes results in an increase in block confirmation time of less than two seconds.

xxBFT PROPERTIES ANALYSIS

From previous work on BFT consensus algorithms, we know that the safety, liveness and validity properties are always guaranteed if up to 1/3 of the network is byzantine. However, this only applies when all nodes in the network participate in consensus.

▷ ENDORSER SAMPLING: A NUMBERS GAME

When a sample is selected from the network to endorse blocks, it is important to analyse the impact on the safety, liveness, and validity properties of the system. In order to provide confidence in the xx blockchain, the xxBFT consensus algorithm parameters should be selected in an optimal way, while keeping the probability of breaking these properties negligible.

When sampling is performed, and since xxBFT is an egalitarian algorithm, all nodes have equal probability of being selected to be an endorser. Let the following parameters be defined:

- $\mathcal{N}$, the size of the network
- E, the size of the endorser set (sample)
- q, the quorum fraction of endorsements needed to approve a block
- h, the fraction of honest nodes in the network
- $\mathcal{X}$, the random variable counting the number of honest nodes sampled

Endorser selection is effectively a random sample of the network with multiple draws, without replacement, i.e., from $\mathcal{N}$, we choose E nodes, where each can only be selected one time. This means that the random variable $\mathcal{X}$ follows a hypergeometric distribution, with the probability mass function given by:

$$P(\mathcal{X} = k) = \frac{\binom{h \cdot \mathcal{N}}{k} \times \binom{(1-h) \cdot \mathcal{N}}{E-k}}{\binom{\mathcal{N}}{E}}$$

In order to break safety, it is sufficient that malicious endorsers can convince two disjoint sets of honest endorsers to accept different blocks (malicious nodes can endorse both blocks). This can be achieved by an adversary that can execute a partition attack, where the two sets of honest endorsers cannot communicate

with each other.

Let the number of honest nodes in an endorser set be $\mathcal{G}$, and the number of malicious nodes $\mathcal{B}$. We can calculate the point of failure, i.e., when safety is broken, for a given endorser set using the following equations:

$$E = \mathcal{G} + \mathcal{B} \ ; \ q \cdot E = \mathcal{B} + \frac{\mathcal{G}}{2}$$

Solving for $\mathcal{G}$ in terms of q and E, we compute the point of failure as $\mathcal{G} = 2 \cdot (1 - q) \cdot E$. This means that the probability of safety failure due to endorser sampling can be computed by the cumulative distribution function (CDF) of the random variable $\mathcal{X}$, evaluated at the point of failure:

$$P_{sf} = CDF_{\mathcal{X}}(2 \cdot (1 - q) \cdot E) = \sum_{k=0}^{2\cdot(1-q)\cdot E} \frac{\binom{h\cdot\mathcal{N}}{k} \times \binom{(1-h)\cdot\mathcal{N}}{E-k}}{\binom{\mathcal{N}}{E}}$$

In order to break liveness, it is sufficient that malicious nodes refuse to endorse any block, leading consensus to halt for the duration of the round. This means that the probability of liveness failure can be calculated in the same way as the probability of safety failure, simply computing the hypergeometric CDF at $\mathcal{G} = q \cdot E - 1$:

$$P_{lf} = CDF_{\mathcal{X}}(q \cdot E - 1) = \sum_{k=0}^{q\cdot E-1} \frac{\binom{h\cdot\mathcal{N}}{k} \times \binom{(1-h)\cdot\mathcal{N}}{E-k}}{\binom{\mathcal{N}}{E}}$$

In order to break validity, it is necessary that malicious endorsers can form a quorum without needing any endorsements from honest nodes. If this happens, they can effectively convince any node in the network that a batch of transactions is valid. The probability of validity failure can also be computed using the hypergeometric CDF, evaluating it at $\mathcal{G} = (1 - q) \cdot E$:

$$P_{vf} = CDF_{\mathcal{X}}((1 - q) \cdot E) = \sum_{k=0}^{(1-q)\cdot E} \frac{\binom{h\cdot\mathcal{N}}{k} \times \binom{(1-h)\cdot\mathcal{N}}{E-k}}{\binom{\mathcal{N}}{E}}$$

▷ THE SAFETY / LIVENESS TRADE-OFF

CAP theorem [10] is a well known result stating that a consensus algorithm cannot guarantee both safety and liveness during periods of asynchrony in a distributed network. This means that when there is a partition of the network, a consensus algorithm design should prioritise one of safety or liveness. BFT algorithms tend to prioritise safety, meaning they only guarantee progress (liveness) during periods of partial synchrony, but they hold safety while the network is asynchronous, never allowing forks to occur. Bitcoin's Proof-of-Work (PoW) based consensus is an example of the opposite: liveness is prioritised, as progress can be made even during a network partition, by allowing forks. This means that safety, in the classical sense, is not held, since different nodes commit different blocks at the same height. Any forks are then resolved with an extra protocol, which in the case of Bitcoin is as simple as the longest chain wins, since that will be the chain with the largest amount of work performed.

▷ SAFETY ANALYSIS

The xxBFT consensus algorithm, like many others in the BFT family, prioritises safety over liveness, meaning that safety needs to hold at all times, in order to not have to deal with forks in the blockchain. By employing endorser sampling it becomes difficult to avoid forks, since in order to make the safety failure probability negligible, liveness needs to be sacrificed. However, this trade-off is only relevant if the consensus algorithm is confined to the sample, without any participation from the rest of the network. This provides the motivation for including a consensus phase where full network participation is required.

xxBFT operates an optimistic path with three endorsement phases: *validate*, *confirm* and *commit*. During the *validate* phase, endorser nodes send a block endorsement to the entire network. A node will then enter the *confirm* phase once it has seen a quorum of endorsements. It will then send back a block confirmation signature to the endorsers. Finally, endorser nodes that have seen a network majority of block confirmation signatures will enter the *commit* phase. Endorsers then send a block commit signature to every node in the network, which upon seeing a quorum of commits will accept the block as final.

We now analyse the behavior of xxBFT when a full network partition is in effect. Without loss of generality, we can assume that all the honest endorsers are evenly distributed between both network partitions. A malicious BP can create two valid blocks and send one to each partition. Both partitions will be able to successfully complete the *validate* phase, since it is very easy to obtain two quorums by having malicious endorsers sign both blocks. However, only one of the partitions will contain a network majority, meaning that it will also complete the *confirm* phase. Honest endorsers in the other partition will never see a majority of confirm signatures, so they will never send out a commit signature.

If malicious endorsers don't send out their commit signatures, both partitions will experience a timeout of the optimistic path, and start executing a full network two phase BFT consensus to try and commit the two different blocks. However, only one partition has a network majority, ensuring only one of the blocks can reach finality. If malicious nodes in the network don't send their signatures, consensus in both partitions will get stuck, until the partition is gone. But, when this happens, all honest nodes in the network will have enough proof that the BP tried to create a fork and he can be expelled from the network.

This analysis proves that the *confirm* phase ensures safety is always preserved in xxBFT, if and only if the endorsers wait for a network majority of confirm signatures. This majority quorum is 2/3+1 for the classic BFT assumption of up to 1/3 byzantine nodes in the network. The quorum can be reduced if the byzantine nodes assumption is relaxed.

▷ LIVENESS ANALYSIS

Most BFT consensus algorithms include the concept of a view change, which happens when a leader is not able to drive consensus in a round, and nodes move to the next round, or view. This can happen, for example, if the leader is malicious and simply doesn't propose a block, or due to network asynchrony. Since there is a rotating leader in all of these BFT consensus algorithms, when a view change happens, a new leader will attempt to drive consensus to an agreement. View change is sufficient in order to preserve liveness, as it guarantees that, under partial synchrony, eventually a correct leader will be able to drive consensus to a decision.

In the xxBFT consensus algorithm there is no view change, meaning that rounds are never skipped, and a block is committed for every round. Instead, we have the concept of an empty block, which doesn't contain transactions, and is deterministic, meaning that no leader is needed to drive a round that produces an empty block. In order to agree on an empty block, a two phase full network BFT consensus in necessary.

When a node reaches a timeout, instead of doing a view change, it will broadcast a $prepare$ signature on the empty block. When a node sees a majority of $prepare$ signatures it can form a $prepareQC$ for the empty block, and it will broadcast a $commit$ empty block signature. If enough nodes timeout, eventually a $commitQC$ is formed and the empty block is committed for that round.

Under partial synchrony, the empty block guarantees liveness is not violated since eventually an honest

node will be selected to be BP and will be able to drive consensus to commit a block containing transactions, instead of an empty block.

However, if the network is asynchronous due to an active partition attack by an adversary, there is a possibility that nodes will be stuck in different consensus states when partial synchrony is restored, which could violate liveness.

▷ WORST CASE PARTITION ATTACK

We will first describe how this attack occurs and then explain how the byzantine resync procedure prevents consensus from halting, meaning that liveness is guaranteed.

To execute this attack, an adversary needs to partition the network in two disjoint sets of honest nodes, each containing 1/3 of the total number of nodes. Without loss of generality, we require that one of the partitions contains an honest BP and enough honest endorsers in order to form a quorum. Let this partition be *setB*. The last remaining honest node, *nodeC*, needs to be able to receive communications from the other partition, *setA*, but not be allowed to send messages to any other honest nodes.

With this attack in place, *setB* will see the block proposal, a quorum of endorsements for that proposal, but not enough endorser $commit$ signatures, meaning that nodes in *setB* will enter the fallback procedure via the non-optimistic block path. They broadcast a $prepare$ signature for the block, with all nodes in *setB* seeing 1/3 of those signatures, and being stuck waiting for a majority. Meanwhile, *setA* and *nodeC* will timeout, broadcasting a $prepare$ signature for the empty block. Nodes in *setA* will see 1/3 of $prepare$ signatures but *nodeC* sees 1/3+1, since only outbound communications from *nodeC* are blocked. At this point in time, all byzantine nodes, which account to the remaining 1/3 of the network, send a $prepare$ signature for the empty block to *nodeC*, allowing it to form a $prepareQC$. Then, *nodeC* will broadcast a $commit$ signature for the empty block, which is not received by any other honest nodes, and will be stuck waiting for a network $commitQC$ for the empty block.

Now, the adversary stops the network partition, allowing *setA* and *setB* to communicate again, but keeping *nodeC* partitioned. Nodes in *setA* will receive the block proposal and $endorsementQC$ from *setB*, which allows them to move to the non-optimistic block path, sending a $prepare$ signature for the block. This way, at this point in time, 2/3 honest nodes are one extra signature away from forming a $prepareQC$ for the block. This signature is sent by one of the byzantine nodes, allowing 2/3 honest nodes to make progress, broadcast $commit$ signatures for the block, and getting stuck waiting for a network $commitQC$ for the block, still only needing one extra signature.

Finally, the adversary allows *nodeC* to communicate with the rest of the network, technically meaning that partial synchrony has been restored. Now, 2/3 honest nodes and *nodeC* have conflicting $prepareQC$ signatures for an empty block and a regular block, meaning that at least one byzantine node must have signed two distinct $prepare$ signatures. However, 1/3 honest nodes have also signed twice, since they entered the fallback procedure via the empty block path and then moved to the non-optimistic block path. This means that, while there is no way to pinpoint any malicious nodes during this attack, it is highly likely that all honest nodes will know that some active attack took place. This allows nodes to enter the byzantine resync procedure, which is a last resort effort to commit a block with transactions. This procedure is a two phase network wide consensus, which from the BFT assumption ensures liveness is guaranteed.

▷ IMPACT OF ENDORSER SAMPLING

While endorser sampling doesn't directly compromise liveness, it does affect how often a consensus round fails, producing an empty block. Even during periods of perfect network synchrony, an honest BP might not be able to drive consensus to commit a regular block. If the selected endorser sample doesn't have enough honest nodes validating transactions and endorsing the block, the endorsement phase will fail, causing nodes in the network to enter the fallback procedure and start trying to reach consensus on an empty block.

▷ VALIDITY ANALYSIS

Traditionally, in most BFT consensus algorithms and blockchain platforms, every node in the network has access to all transactions, as these are shared via a mempool or other kind of gossip protocols. This means that when a block is proposed containing a batch of transactions, every node in the network will validate all of them against their local record of transactions, before accepting that block. Even in protocols that have a subset of the network endorsing a block, this still applies, as everyone has the transactions. This means that on these platforms validity failure is never an issue.

On the other hand, in xxBFT, the batch of transactions is only sent to the endorser sample, meaning that there is always a probability of validity failure. We will analyze how to minimize this probability in order to make it negligible, ensuring that in practice a bad transaction will never be accepted.

As previously mentioned, when an endorser sample is taken from the network, there will be a probability that enough malicious nodes are selected and can form an endorsement quorum. If this happens, validity is broken, since the quorum of malicious nodes can endorse a block with any fake/bad transactions, which other nodes in the network will accept as valid. Furthermore, this can only happen if the BP is also malicious, as an honest BP would correctly not include any bad transactions in its block proposal.

An adversary aiming to break validity will need to carry out an active attack. First, the adversary needs to wait for a favorable endorser sample that contains a quorum of malicious nodes, and also be in control of the BP of the given round, so that bad transactions can be included in the block. Then, he needs to fully partition all the honest endorsers from the network, so that they don't receive any transactions. This is needed since, otherwise, any honest endorser would have proof that malicious endorsers and the BP signed a block containing bad transactions, which is sufficient to have them expelled from the network.

The probability of breaking validity is directly influenced by the size of the endorser sample, the quorum needed to accept an endorsement as valid, and the percentage of byzantine nodes in the network. Relaxing the byzantine nodes assumption from 1/3 to a smaller value will decrease the probability of breaking validity. However, in order to stay in line with other BFT consensus algorithms, we maintain the 1/3 byzantine nodes assumption. Selecting a larger endorser sample will lead to a smaller probability of breaking safety, as will increasing the quorum. However, these increases negatively impact performance and liveness of the network. A larger endorser sample will decrease performance, as it will take longer to send a transaction batch to more endorsers, and a higher quorum needed to approve a block will decrease liveness in perfect network synchrony. In the next section we will analyze how to make validity failures negligible, while maintaining good performance and liveness.

▷ IMPLEMENTATION PARAMETERS

We now choose parameters for xxBFT and calculate the probabilities of liveness and validity failures. First, we define the network size as being arbitrarily large, allowing us to upper bound probabilities of failure for endorser sampling. This upper bound exists since the hypergeometric distribution asymptotically tends to the binomial distribution when the network size is much larger than the endorser sample. In practice we set $\mathcal{N}$ to 10 million nodes, which is much larger than any sensible value of E.

Then, we use the standard BFT assumption for the number of honest nodes in the network, meaning that h is 2/3. After running various simulations we believe that acceptable values for the endorser sample and quorum percentage have been achieved: $E = 200$ and $q = 0.60$.

In order to better represent the validity failure probability, we will use the Mean Time To Failure ($MTTF$) metric, which can be calculated in years as:

$$MTTF = \frac{B_t}{3.154 \times 10^7 \times P_{vf}}$$

where, B_t is the expected block size in seconds, which we conservatively set to 2.

With the chosen parameters and computing the expressions presented in previous sections, we achieve a probability of validity failure of **3.7129x10^{-15}**, which is equivalent to a $MTTF$ of around **17 million years**. In terms of liveness failure, the probability of endorser sampling causing an empty block under perfect network synchrony is approximately **2%**. As previously analyzed, endorser sampling has no negative impact on safety.

6 NETWORK INITIALIZATION

We introduce a setup phase for a quantum-secure blockchain platform. The goal of this protocol is to produce a Genesis Block containing the list of network members, the initial random seed, and monetary value in the system. The described setup must eliminate the possibility of eavesdropping or tampering, even by a quantum-capable adversary.

NODE PLACARDS

Nodes involved in the setup phase create a set of values and organize them in a hash tree called the Placard Tree. This tree's root is the node's identifier (or pseudonym) in the network.

The placard tree values of each individual node comprises the following branches:

- **Random Seed**: individual value to be used for the Decentralized Random Number Generation.
- **Random Values**: set of random bit strings to be revealed throughout the operation of the platform. Nodes should reveal one of these values every round they are selected to be a Block Producer.
- **WOTS+ Signing Keys**: to ensure quantum-security of the platform, nodes use hash-based signatures to sign every block they should endorse.

DECENTRALIZED RANDOM NUMBER GENERATION

To securely create a random seed for the system, each node provides their individual contribution by opening their individual commit. Nodes who fail to do so are ejected from the network. Once this reveal process is completed, the random number generation is finalized. The resulting seed should serve as a reliable and trusted RNG. For further protection, one may add a cryptographic wrapper [1] to reduce the impact of untrusted randomness. This way, nodes attempting to influence the random number generation by revealing malicious values see their efforts fail as the wrapper mitigates this problem.

GENESIS BLOCK

After the system random is generated, nodes proceed to sign the genesis block. The process is considered complete when at least $2f + 1$ of nodes publish their signatures (which can also take place during a specific timeline). The Genesis Block should contain the random seed associated with the setup phase, along with the list of all token addresses (quantum-secure public keys), and a network signature attesting to the integrity of the block data. Once the network obtains this final signature on the Genesis block, the network initialization is officially completed.

PHYSICAL EVENT DETAILS

Network initialization can be performed as a physical event, where participants must be able to physically open a digital commit (e.g., by showing a QR code containing the value that was committed to) and sign a genesis block. One way to do this is through the use of a table with specially allocated slots for each of the signers.

At arrival, participants prepare their corresponding random value contribution and place it in a verifiable and transparent container so that everyone can check their elements are properly stored. Nodes, during the random number generation phase, securely place a card containing a QR code with their corresponding random value. Once the process is completed, the cards can be flipped in a secure and verifiable way such that the final network random cannot be tampered with. To sign the Genesis block, nodes can follow the same procedure and use the same slots to place their signature bits.

SUMMARY

This section introduced a mechanism to initialize a quantum-secure blockchain platform. If desired, potential nodes can perform the network initialization in the form of an in-person event, following the sequence of steps described above along with a possible symmetric key establishment to introduce confidential channels in the network, thus limiting the power of the adversary.

7 RELATED WORK

Although the seminal paper on Bitcoin appeared in 2008, most of the underlying technological ideas have been present both in the cryptography and distributed systems fields many years earlier.

Proof-of-Work (PoW) is usually associated with Dwork and Naor [8] because of their proof-of-computation design to fight junk mail in 1992. However, the term was formally introduced in [12] and the first proposal involving proof of computational work dates back to the early 70s. In 1974, Ralph Merkle [15] introduced a public-key cryptosystem where a receiving party must perform a certain amount of computational work to prove that it found a solution to a cryptographic puzzle.

In 1979, Chaum [2] introduced a design where multiple suspicious parties are able to securely establish a distributed system capable of collectively agreeing on incoming transactions. Since the network membership set is known, this can be considered the first **permissioned blockchain** design since in every round, nodes maintain a hash chain that contains all the previous consensus states, thus creating a distributed chain of records.

Byzantine Consensus ensures that a distributed system is able to agree on a specific set and order of transactions to be processed even in the presence of network failures where certain network nodes may arbitrarily fail or provide conflicting information to different parts of the network. Protocols of this type are called **Byzatine Fault Tolerant (BFT)** and most of them can only tolerate up to 1/3 of byzantine failures in the network, as proven in [17].

Digital cash, created by Chaum [5], introduced the idea of digital coins that relied on public-key cryptography [3] for security. In this system, randomly generated serial numbers signed by the private key of a central authority (e.g., bank) could be spent in a private manner. An important result that derived from this work is the notion of **Double Spending** [4], where users attempt to spend the same digital money more than once. This is now a core concept behind any platform that deals with digital money.

More than two decades later, Nakamoto [16] introduced **Bitcoin**, a cryptocurrency that relies on PoW to maintain a decentralized ledger. In contrast with other distributed systems, Bitcoin nodes are able to join or leave the network at will and do not require knowledge of the membership set, thus making it the first **permissionless blockchain**. However, this feature results in a change of paradigm in the distributed systems realm: safety changes from deterministic to probabilistic. As a result, when transacting in the network, users should typically wait 4-6 blocks until considering a transaction final.

PoW poses as an effective **Sybil resistance** [6] mechanism against attackers attempting to create multiple identities in the network. Platforms like Bitcoin, require network participants to perform a significant amount of computational work. Therefore, attempting to perform substantial computational work under different pseudonyms at the same time quickly turns out to be a very expensive attack.

To avoid high energy consumption from the constantly used computational power of the network and to provide faster finality, blockchain platforms such as [9][14][13] rely on **Proof-of-Stake (PoS)** instead. PoS implies that the more monetary value users have in the system, the more influence they have on the consensus protocol, which implies that theoretically, an attacker must spend a significant amount of money to initiate an attack at a scale large enough to maliciously influence the consensus.

Algorand [9] is a PoS system that introduces a novel consensus protocol that uses a **random sample** of the network to reach a faster agreement on new blocks. Unlike most PoS platforms, Algorand's consensus protocol does not employ a **slashing** mechanism where nodes, if caught performing malicious actions in the network, see their stake slashed.

REFERENCES

[1] Akhmetzyanova, L., Cremers, C., Garratt, L., Smyshlyaev, S. V., and Sullivan, N. Limiting the impact of unreliable randomness in deployed security protocols. Cryptology ePrint Archive, Report 2018/1057, 2018.

[2] Chaum, D. *Computer Systems established, maintained and trusted by mutually suspicious groups.* Electronics Research Laboratory, University of California, 1979.

[3] Chaum, D. Blind signatures for untraceable payments. In *Advances in cryptology* (1983), Springer, pp. 199–203.

[4] Chaum, D. Achieving electronic privacy. *Scientific American 267*, 2 (1992), 96–101.

[5] Chaum, D., Fiat, A., and Naor, M. Untraceable electronic cash. In *Conference on the Theory and Application of Cryptography* (1988), Springer, pp. 319–327.

[6] Douceur, J. R. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems* (London, UK, UK, 2002), IPTPS '01, Springer-Verlag, pp. 251–260.

[7] Dwork, C., Lynch, N., and Stockmeyer, L. Consensus in the presence of partial synchrony. *Journal of the ACM (JACM) 35*, 2 (1988), 288–323.

[8] Dwork, C., and Naor, M. Pricing via processing or combatting junk mail. In *Annual International Cryptology Conference* (1992), Springer, pp. 139–147.

[9] Gilad, Y., Hemo, R., Micali, S., Vlachos, G., and Zeldovich, N. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles* (2017), ACM, pp. 51–68.

[10] Gilbert, S., and Lynch, N. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *Acm Sigact News 33*, 2 (2002), 51–59.

[11] Hülsing, A. W-OTS+ — shorter signatures for hash-based signature schemes. In *Lecture Notes in Computer Science (LNSC)* (2013), vol. 7918, pp. 173–188.

[12] Jakobsson, M., and Juels, A. Proofs of work and bread pudding protocols. In *Secure Information Networks*. Springer, 1999, pp. 258–272.

[13] Kiayias, A., Russell, A., David, B., and Oliynykov, R. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference* (2017), Springer, pp. 357–388.

[14] Kwon, J., and Buchman, E. A network of distributed ledgers. *Cosmos, dated* (2018), 1–41.

[15] Merkle, R. C. Secure communications over insecure channels. *Commun. ACM 21*, 4 (1978), 294–299.

[16] Nakamoto, S. Bitcoin: A peer-to-peer electronic cash system.

[17] Pease, M., Shostak, R., and Lamport, L. Reaching agreement in the presence of faults. *Journal of the ACM (JACM) 27*, 2 (1980), 228–234.

APPENDIX

A. xxBFT PSEUDO CODE

Algorithm 1 - Consensus Optimistic Path

for all rounds **do**
▷ PROPOSE phase
as a Block Producer
BUILD *block*
SEND *stateUpdate* to Network
SEND *block* to Network
SEND *stateUpdate* + *transactionBatch* to Endorsers
as all Network
WAIT *block*
as the next BP
DECODE *transactionBatch*
WAIT *block*

if Network does not receive *block* **then**
Consensus Fallback Empty Block Path
end if

▷ VALIDATE phase
as a Block Producer
SEND *stateUpdate* + *transactionBatch* to Endorsers
SEND *stateUpdate* to Network
as an Endorser
WAIT *stateUpdate* + *transactionBatch*
DECODE *transactionBatch*
VALIDATE *stateUpdate*
SEND *blockEndorsement* to Network
as all Network
WAIT *stateUpdate*
WAIT *blockEndorsement* QC
as the next BP
DECODE *transactionBatch*
WAIT *stateUpdate*

if Network does not receive *blockEndorsement* QC **then**
Consensus Fallback Empty Block Path
end if

▷ CONFIRM phase
as an Endorser
    WAIT $2/3 + 1$ of *blockConfirmation*
as all Network
    SEND *blockConfirmation* to Endorsers
as the next BP
    BUILD *nextTransactionBatch* + *nextStateUpdate*

▷ COMMIT phase
as an Endorser
    SEND *blockCommitment* to Network
as all Network
    WAIT *blockCommitment* QC
    COMMIT *block*
as the next BP
    BUILD *nextTransactionBatch* + *nextStateUpdate*

if Network does not receive *blockCommitment* QC **then**
    Consensus Fallback Non-optimistic Block Path
end if

end for

Algorithm 2 - Consensus Fallback Empty Block Path

for all rounds **do**
▷ CONFIRM phase
as all Network
    SEND *emptyBlockConfirmation* to Network
    WAIT *emptyBlockConfirmation* QC
as the next BP
    DECODE *transactionBatch*

▷ COMMIT phase
as all Network
    SEND *emptyBlockCommitment* to Network
    WAIT *emptyBlockCommitment* QC
    COMMIT *EmptyBlock*
as the next BP
    BUILD *nextTransactionBatch* + *nextStateUpdate*
end for

Algorithm 3 - Consensus Fallback Non-optimistic Block Path

for all rounds **do**
    ▷ CONFIRM phase
    as all Network
        SEND *blockConfirmation* to Network
        WAIT *blockConfirmation* QC
    as the next BP
        DECODE *transactionBatch*

    ▷ COMMIT phase
    as all Network
        SEND *blockCommitment* to Network
        WAIT *blockCommitment* QC
        COMMIT *Block*
end for

B. xxBFT FLOWCHART

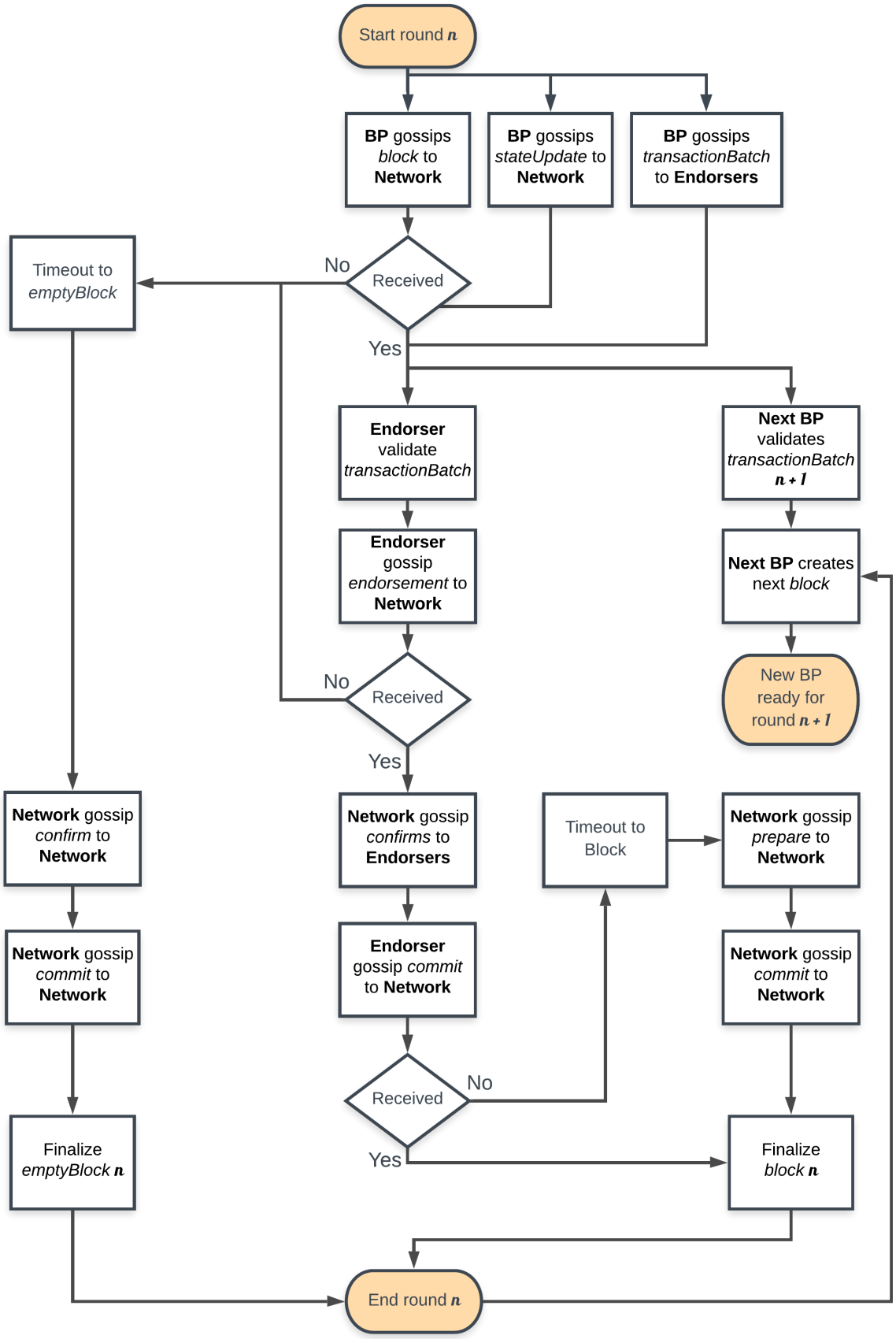

Figure 4: xx BFT Flowchart

C. xxBFT STATE MACHINE DIAGRAMS

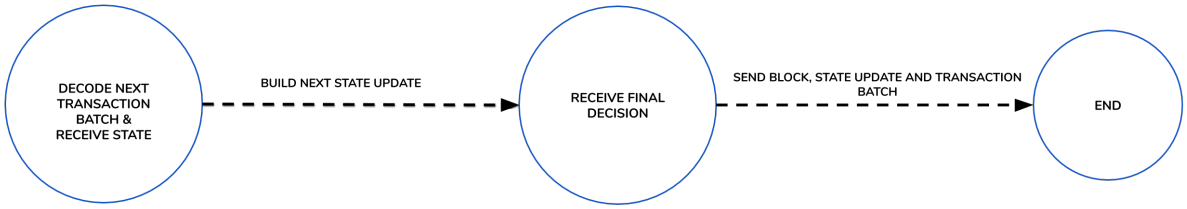

Figure 5: BP State Machine

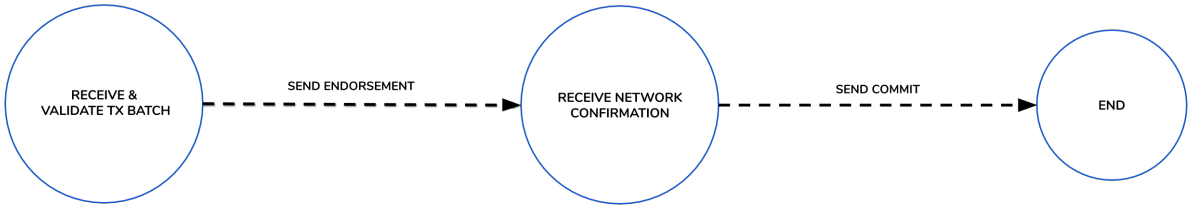

Figure 6: Endorser State Machine

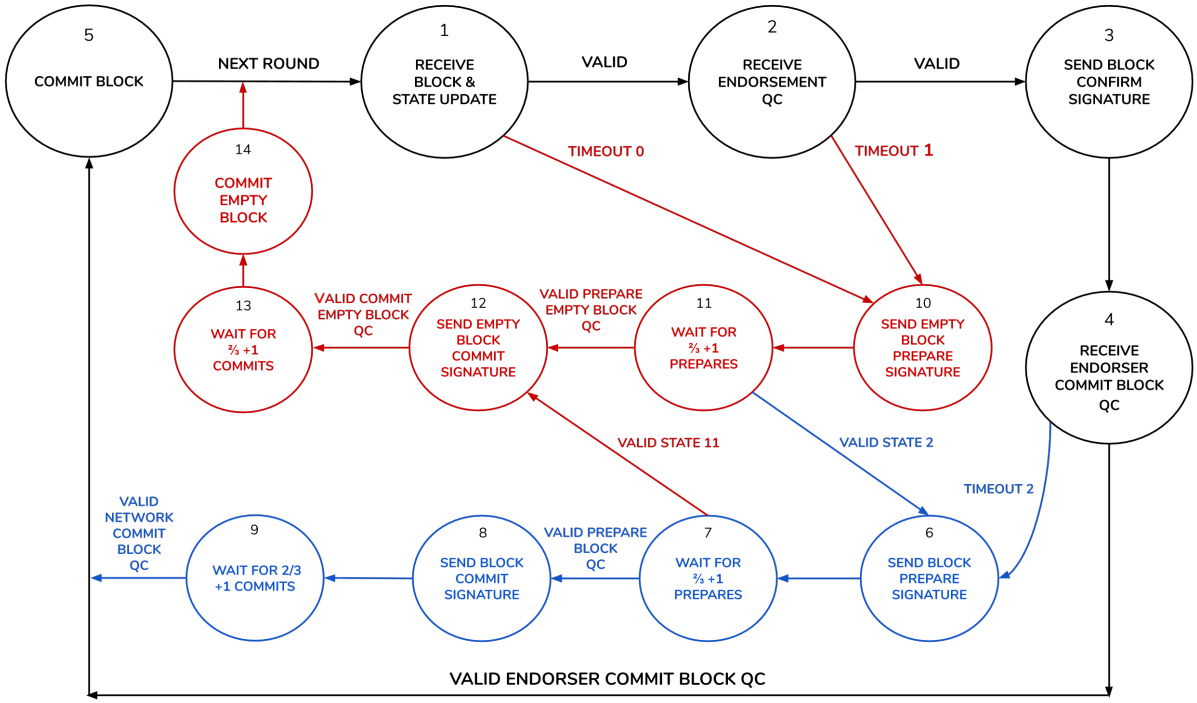

Figure 7: Network State Machine