Consensus Chronicles: A Multivariate Exploration of Blockchain Consensus Algorithms

Garima Goyal and Sagar Ghimre

May 23, 2024

Abstract

As the inexorable march of database and communication technologies propels us forward, the demand for system availability and autonomy has become an inescapable imperative. Distributed systems offer a tantalizing solution, but harbor a formidable challenge: how to distribute data and programs in a manner that optimizes performance while upholding the sacrosanct principles of security and availability.

In this crucible of complexity, blockchain research has emerged as a vanguard of computer science, heralding the potential to revolutionize the very notion of trust in distributed systems. At its core, a blockchain is a decentralized peer-to-peer network, a digital realm where data is enshrined across a vast network of servers, forging a collective trust among its denizens. **Fabric**, a modular and extensible open-source system hosted by the Linux Foundation's Hyper-ledger project, stands as the first truly extensible blockchain platform for running distributed applications. Supporting modular consensus protocols tailored to diverse use cases and trust models, Fabric introduces an entirely novel blockchain design, revamping how the technology copes with non-determinism, resource exhaustion, and performance attacks. Departing from existing platforms reliant on domain-specific languages or cryptocurrencies, Fabric empowers distributed applications to be written in standard, general-purpose programming languages, without systemic dependency on a native cryptocurrency. This paradigm shift is realized through a portable notion of membership, seamlessly integrating with industry-standard identity management. The disruptive potential of this technology transcends its origins, pervading domains as diverse as e-commerce systems, high-volume transaction markets, asset management, and exchanges. By distributing data and enabling trust between nodes, blockchain technology offers an innovative solution for secure and available distributed systems. This paper elucidates Fabric's architecture, the rationale underpinning its design decisions, and its prominent implementation aspects, while delineating its distributed application programming model.

In a landscape where distributed consensus systems lack formal analysis and concrete evaluation criteria, this work proposes a novel framework for assessing the effectiveness of distributed consensus methods under economic considerations. Deriving challenges in proof-of-stake consensus, this paper advances the formal analysis of this highly dynamic field, where much work is committed outside traditional academia.

1 Introduction

Blockchain technology has emerged as a revolutionary force, enabling secure and transparent transactions across a distributed network of untrusted parties. At the core of many popular cryptocurrency blockchains like Bitcoin and Ethereum lies the Proof-of-Work (PoW) consensus mechanism. PoW involves miners solving computationally intensive cryptography puzzles to validate new blocks, with successful miners receiving rewards. This process establishes network-wide consensus on the canonical blockchain by having nodes build upon the longest chain of valid blocks.

However, as blockchains scale to industrial deployments with geographically distributed nodes, network delays can significantly impact performance and security. Delays increase the likelihood of temporary forks emerging as miners work on divergent chain views, leading to wasted computational work on orphaned blocks. This slows the overall blockchain growth rate and transaction processing. Crucially, delays also heighten the vulnerability to adversarial attacks aimed at double-spending or rewriting history, even if the attacker controls less than the commonly assumed 51

To address these challenges, advanced analytical models have been developed to capture the stochastic dynamics of blockchain evolution under network delays. These models provide insights into the long-term growth rate, occurrence of forks, and the amount of wasted mining effort as a function of the delay parameter. The models enable setting appropriate ”difficulty-of-work” targets to achieve desired performance. Moreover, they quantify the adverse impact of delays on blockchain security, precisely characterizing the probability of successful attacks for given attacker hashing power and confirmation thresholds.

At the forefront of this innovation is Hyper-ledger Fabric, an open-source blockchain platform designed for enterprise-grade applications. Developed under the Linux Foundation’s Hyper-ledger umbrella, Fabric introduces a groundbreaking architecture that addresses the limitations of traditional blockchains, offering unparalleled resilience, flexibility, scalability, and confidentiality.

Fabric departs from the conventional ”order-execute” paradigm, pioneering an innovative ”execute-order-validate” approach. This novel design enables parallel execution of transactions, accommodates non-deterministic operations, and respects application-specific trust assumptions. By separating the transaction flow into distinct phases – execution, ordering, and validation – Fabric ensures robust consensus mechanisms tailored to diverse use cases.

Complementing Hyper-ledger Fabric’s cutting-edge capabilities is IBM’s comprehensive blockchain solutions suite. IBM Blockchain Platform, built on Fabric, empowers enterprises to develop, deploy, and operate blockchain networks with unmatched security, performance, and governance. Leveraging decades of expertise in enterprise computing, IBM provides a secure, production-ready environment for blockchain applications, enabling seamless integration with existing systems and processes.

Furthermore, **IBM’s blockchain offerings extend beyond Fabric**, encompassing a range of innovative technologies and services. The IBM Blockchain Platform Starter Plan allows organizations to explore blockchain capabilities through a managed cloud service, while the IBM Blockchain Platform Enterprise Plan delivers a fully-managed, production-ready blockchain environment. Additionally, IBM Blockchain Services provide consulting, implementation, and support services, ensuring successful blockchain adoption and integration. Together, Hyper-ledger Fabric and IBM’s blockchain solutions are driving digital transformation across industries, enabling secure and transparent transactions, streamlining business processes, and fostering collaboration among diverse stakeholders. With their robust architectures, flexible deployments, and comprehensive support, these technologies are poised to revolutionize the way enterprises operate, fostering trust, transparency, and efficiency in an increasingly connected global economy.

2 Hyperledger Fabric Architecture

Hyperledger Fabric ushers in a paradigm shift in blockchain architectures with its innovative execute-order-validate model, revolutionizing the transaction processing flow. This groundbreaking design tackles long-standing limitations and ushers in a new era of resilience, flexibility, and performance for enterprise blockchains.

During the execute phase, as depicted in the diagram(Figure 1), transactions are processed in parallel across only a subset of peers called ”endorsing peers,” eschewing the conventional approach of executing every transaction on every node. These endorsing peers run the smart contract code (chaincode) and simulate the transaction, generating an endorsed transaction response. This parallel endorsement is governed by a flexible endorsement policy that aligns with the trust assumptions of the specific smart contract. Policies can mandate endorsement from a pre-defined set of peers or require a minimum number of endorsements, unlocking performance gains and accommodating non-deterministic operations. The ordering phase decouples the consensus mechanism from the transaction execution, delegating the establishment of a total order to a modular ordering service. As illustrated in the diagram, the endorsed transactions from various peers are collected and ordered by the ordering service using pluggable consensus protocols like Crash Fault-Tolerant Raft or Byzantine Fault-Tolerant SMaRt. This stateless service seamlessly integrates well-established distributed system toolkits. Separating ordering from execution enables tailoring the consensus layer to diverse trust models. In the validation phase, each peer, including the endorsing peers, meticulously validates the ordered transactions against the endorsement policy before committing state updates to their respective ledgers (blocks + world state). This deliberate design prevents insidious race conditions that could arise from speculative execution, ensuring robust consistency despite the parallel architecture.

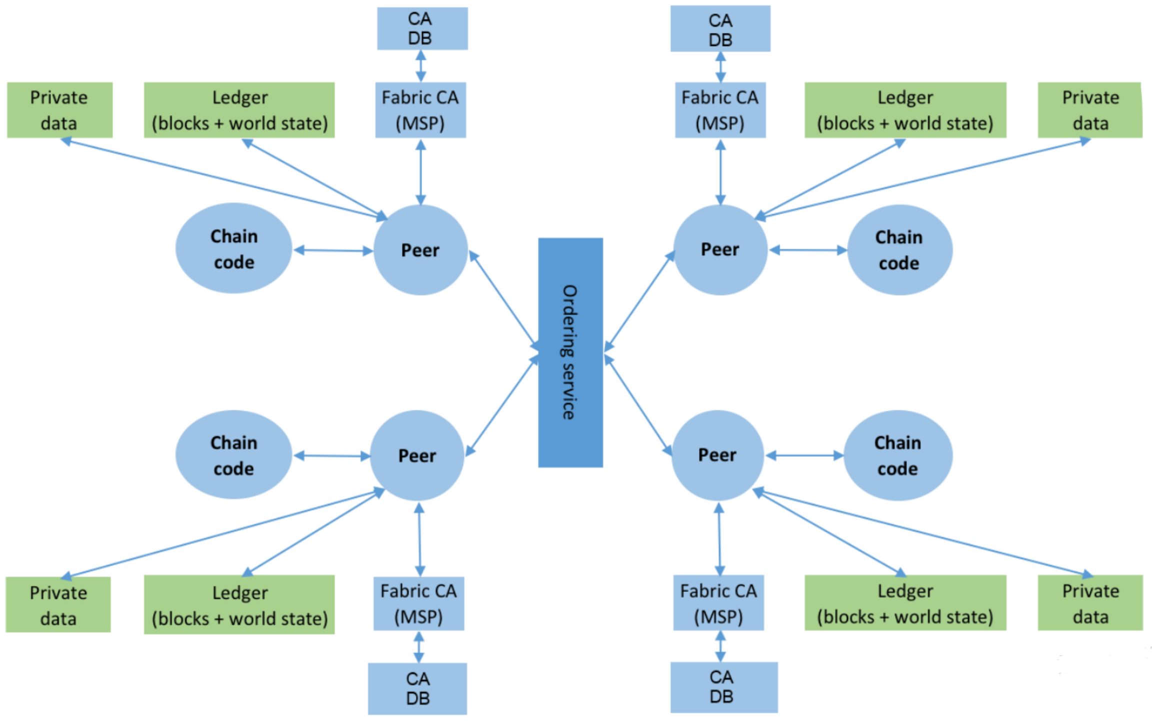


Figure 1: Hyperledger Fabric Architecture

Fabric's architectural innovations blend passive primary-backup replication, reminiscent of middleware-based database replication, with the principles of active replication employed in Byzantine fault-tolerant systems. The endorsement propagation from endorsing peers mimics passive replication, while the ordering service facilitates active parallel execution – a hybrid paradigm purpose-built for untrusted environments.

Smart contracts, termed "chaincode," benefit from the flexibility of standard programming languages like Go, Node.js, and Java. They are deployed within isolated Docker containers on peers, ensuring secure execution within the blockchain context while prohibiting direct access to ledger state databases. Dockerized deployment streamlines chaincode lifecycle management across globally distributed networks. Complementing the core architecture are auxiliary components like the membership service provider (Fabric CA (MSP)) for managing identities and cryptographic material, and an optional gossip data dissemination protocol optimized for efficient state transfer among peers.

Through its modular design, pioneering execute-order-validate paradigm, and hybrid replication model, Hyperledger Fabric redefines blockchain architectures. It establishes itself as the preeminent distributed operating system for enterprise networks, providing the resilience, flexibility, and confidentiality demanded by real-world applications while propelling blockchain technology into a new frontier of performance and scalability.

3 Distributed Consensus Methods

A distributed consensus method is a way to securely select a command for execution on a distributed state-machine. Looking at Bitcoin, we've seen an example of such a method used for securely updating account balances. Since Bitcoin's inception in 2008, many other consensus methods have been proposed for updating distributed state in a blockchain.

We define a blockchain-based consensus method as having:

i)Consensus-Determination Function (D), which determines the ultimate block in a chain of blocks from any given blockchain structure (D: C $\rightarrow$ B).

ii)Consensus Strategy (S), which decides which block a client will apply next to a given blockchain structure (S: C $\rightarrow$ B).

iii) State-Transition Function (T), which governs how applying a block to a blockchain transforms the blockchain (T: C × B → C).

3.1 Computational Power: Proof of Work

In cryptocurrency blockchains like Bitcoin and Ethereum, nodes utilize Proof-of-Work (PoW) as the mechanism for validating new blocks and achieving consensus on adding them to the ledger. PoW entails finding a random number (nonce) that, when hashed (e.g., using SHA-256) along with the hash of the previous block, produces a hash with a specific number of leading zero bits. The computational effort required for PoW increases exponentially with the number of zero bits required, potentially demanding significant CPU resources. A block is deemed successfully "mined" by a node if all transactions included in it are valid and the PoW criteria are met. Nodes engaging in PoW are referred to as "miners," and they receive rewards for mining blocks that are added to the blockchain.

Proof-of-work systems require participants who want to publish a state transition to solve a computational problem that demands a certain amount of work on average. Once a participant finds a solution, they share it along with their proposed state transition. The key insight into how consensus is reached in these systems are:

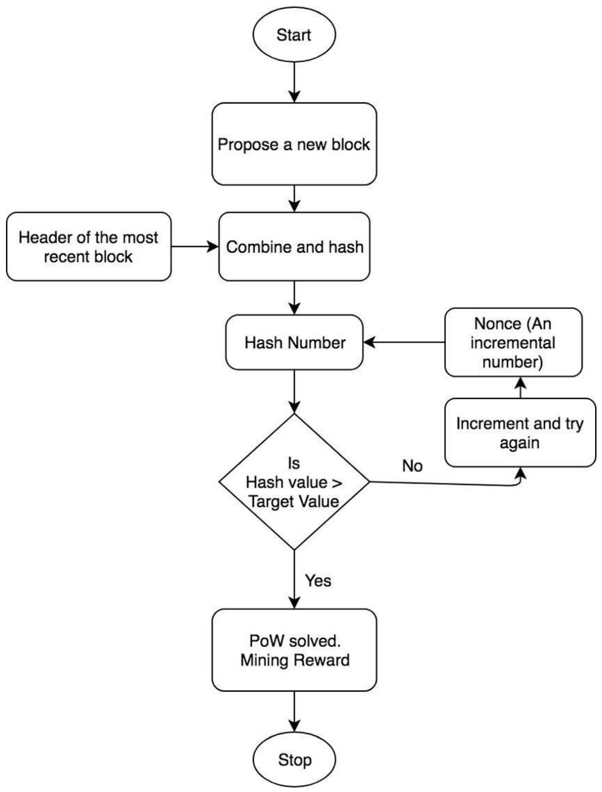

Start
Propose a new block
Header of the most recent block
Combine and hash
Hash Number
Nonce (An incremental number)
Increment and try again
Is Hash value > Target Value
No
Yes
PoW solved. Mining Reward
Stop

Figure 2: Proof of Work

- Multiple participants can propose state transitions, and others continuously verify the validity of the solutions shared with these proposals.
- Participants accept a state transition once its solution is validated, as it demonstrates that someone invested effort in solving the problem. This solution serves as proof of the computational work done by the publisher.
- All participants can reach a consensus on the work invested in publishing a state transition. Since this work usually incurs an economic cost (such as electricity), the publisher bears this cost.
- If most of the computational power in the system is controlled by honest participants, who aim to publish valid state transitions, the system's state is expected to remain valid in the long

run. Honest participants are likely to "win" state transitions more frequently, forming the longest chain. Fraudulent state transitions on other chains, despite incurring a cost, would be invalidated by the longest chain. Therefore, participants can confidently agree on applying a state transition once a small set of proofs of work has been shared.

A proof-of-work puzzle must meet two primary criteria: Its level of difficulty should be highly scalable to prevent it from becoming too easy with technological advancements, and a solution should be readily verifiable to keep network verification costs low. There is ongoing research into formally specifying acceptable puzzles.

3.2 System Stake: Proof of Stake

Proof of Stake (PoS) is a consensus algorithm used in blockchain networks to achieve agreement on the validity of transactions and the state of the distributed ledger. Unlike Proof of Work (PoW), where participants compete to solve complex mathematical puzzles to validate transactions and create new blocks, in PoS, the creator of a new block is chosen in a deterministic (non-random) way based on the amount of cryptocurrency they hold and are willing to "stake" as collateral.

Delegated Proof of Stake (DPoS) builds upon the proof-of-stake concept by assigning state transition rights to a small group of participants known as delegates. These delegates are elected for a specific period through various methods, which could involve automated selection based on stake control or require participants to vote for trustworthy delegates.

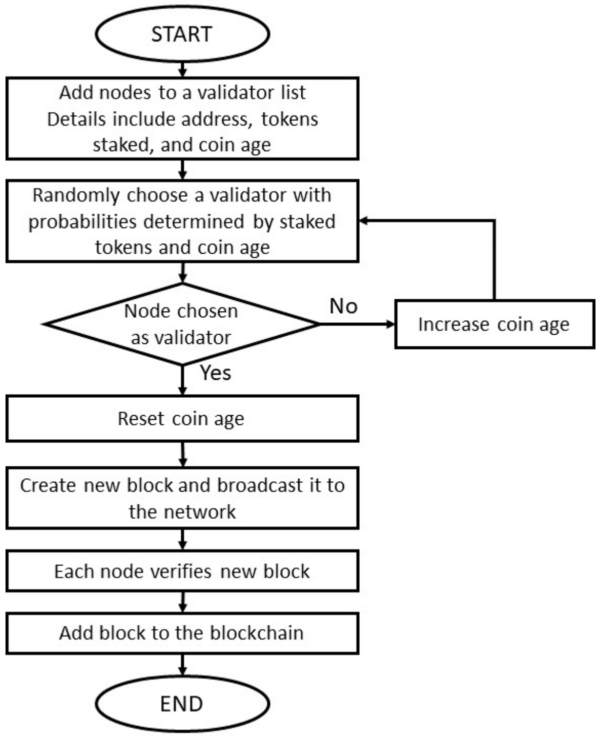

START
Add nodes to a validator list Details include address, tokens staked, and coin age
Randomly choose a validator with probabilities determined by staked tokens and coin age
Node chosen as validator
No
Increase coin age
Yes
Reset coin age
Create new block and broadcast it to the network
Each node verifies new block
Add block to the blockchain
END

Figure 3: Proof of Stake flowchart

The way by which the DPoS works:
1. Token holders' vote: Token holders participate in the consensus process by voting for a certain

number of delegates or witnesses from a pool of candidates.

2. Delegate selection: The delegates with the highest number of votes become active block producers. These delegates are responsible for validating transactions, proposing new blocks, and maintaining the network.

3. Block production: Active delegates take turns producing blocks in a predetermined order. The frequency of block production and the number of active delegates can vary depending on the specific implementation.

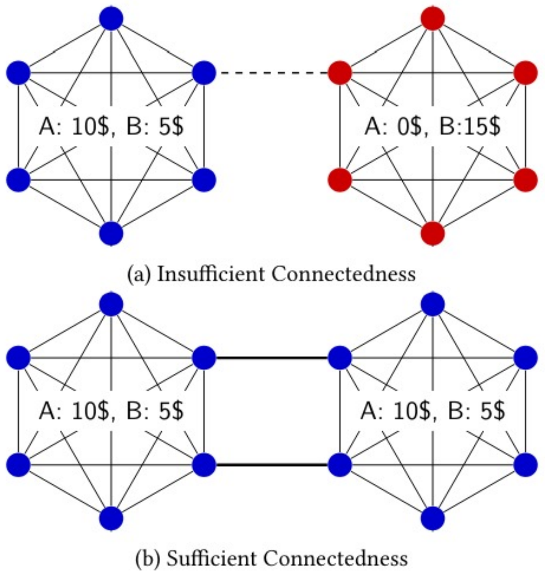
4. Block verification: A fter a b lock i s p roduced, i t i s v erified an d ad ded to th e bl ockchain by the other delegates and nodes in the network.

3.3 Byzantine Agreement

Byzantine Agreement, a concept stemming from the Byzantine Generals Problem, addresses consensus in distributed systems where participants may fail arbitrarily, including due to malicious actions or technical issues. Practical Byzantine Fault Tolerance (PBFT) and Paxos are two prominent methods for achieving Byzantine Agreement. This approach enables quick and economical consensus compared to proof-of-work and doesn't necessitate ownership of assets, as seen in proof-of-stake systems. How-ever, Byzantine Agreement systems typically require a fixed s et o f p articipants, p osing c hallenges if this set isn't precisely defined a nd s hared a mong participants.

3.3.1 Probabilistic Voting (Ripple)

An application of Byzantine Agreement, exemplified b y t he R ipple p rotocol c onsensus algorithm (RPCA), advances distributed blockchain ledgers by maintaining balance and currency records. In RPCA rounds, nodes collect and validate transactions, discarding those not supported by a threshold of nodes in their trust list. With RPCA's threshold set at 80 percent, the system can tolerate up to 20 percent Byzantine nodes while still functioning correctly. Beyond 20 percent but below 80 per-cent Byzantine nodes, consensus may be compromised, potentially resulting in no transactions being applied. If 80 percent or more nodes are compromised, the system fails to ensure ledger correct-ness. Furthermore, RPCA necessitates a minimum level of interconnectedness among nodes to achieve consistent consensus across the network.

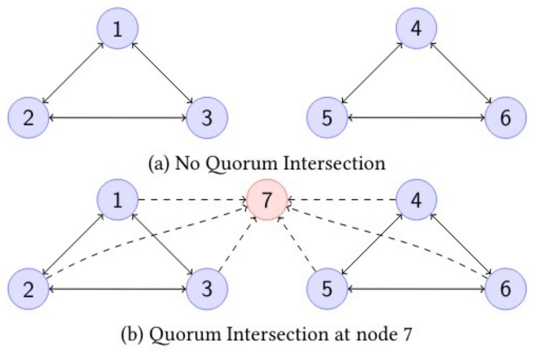


(a) Insufficient Connectedness

(b) Sufficient Connectedness

3.3.2 Federated Byzantine Agreement (Stellar Consensus Protocol)

Motivated by the aforementioned ledger fork in an implementation of RPCA, Mazières proposed Fed-erated Byzantine Agreement (FBA) in November 2015 and furthermore made explicit the conditions which must hold in order for a system to deterministically reach consensus. FBA relaxes the require-ment for a centrally supervised list of UNLs and instead gives participants freedom in choosing their "trusted" nodes in what the author calls quorum slices. Each node in a Federated Byzantine Agree-ment System (FBAS) can specify various quorum slices, sets of nodes whose consensus on some fact is sufficient to also convince the node of said fact. The composite of all quorum slices forms a network of trust which gives rise to quorums. Quorums are sets of nodes which are sufficient to convince a subset of nodes in an FBAS of some fact. As such, a quorum is a weak superset of all participating nodes,



(a) No Quorum Intersection

(b) Quorum Intersection at node 7

Figure 5: Quorum intersection

generated by the union of (at least) one quorum slice of each participant. In this setting, an FBAS maintains consensus if two properties are met: Quorum Intersection and Quorum availability.

Quorum Intersection mandates that the intersection of all available quorums in an FBAS contain at least one correctly functioning and honest node. Quorum intersection is required in order to prevent forks in the ledger history.

Quorum Availability mandates that the FBAS contain at least one quorum that does not contain incorrectly functioning or malicious nodes. If Quorum Availability is violated, the FBAS can be in a stuck state where no consensus can be reached because of failing nodes that do not answer to requests [1, 3]

3.4 Consensus Methods based on other Economic Aspects

There are certain consensus methods yet to be implemented but have the potential to be widely adopted, such as Proof of Activity, Proof of Burn, Proof of Capacity, Proof of Stake Velocity, and Proof of Bandwidth. Each of these methods presents unique approaches to achieving consensus in distributed systems, offering different trade-offs and potential applications.

Proof of Activity: This method requires participants to solve a proof-of-work puzzle and then obtain signatures from past block publishers to successfully publish a block. This approach makes attacks more difficult but can lead to delays if past publishers are unavailable.

Proof of Burn: In this system, participants burn coins by sending them to randomly selected addresses, earning them a chance to publish state transitions. The difficulty of the puzzle participants must solve is inversely related to the amount of burned coins. Slim coin is an example of a system using proof of burn.

Proof of Capacity: State-transition probabilities are distributed based on participants' memory and hard drive capacity. Burst is an example of this system, requiring participants to store and manipulate large amounts of random data.

Proof of Stake Velocity: This is a subset of general proof-of-stake systems where state tran-sition probabilities are distributed according to coin-age weighted balances. The design encourages participants to turn over their balances quickly, increasing the "velocity" of their average balance.

Proof of Bandwidth: This method allocates state transition probabilities based on participants' available bandwidth in a network. Participants are connected in "paths" within the network, and the weight of these paths is measured and signed by decentralized servers according to throughput. Tor coin is an example of this method, incentivizing increased bandwidth in the Tor network.

Each of these methods presents unique approaches to achieving consensus in distributed systems, offering different trade-offs and potential applications.

References and citation

[1]"Hyperledger fabric: a distributed operating system for permissioned blockchains", Authors: Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis,Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich.

[2] N. Papadis, S. Borst, A. Walid, M. Grissa and L. Tassiulas, "Stochastic Models and Wide-Area Network Measurements for Blockchain Design and Analysis," IEEE INFOCOM 2018- IEEE Conference on Computer Communications, Honolulu, HI, USA, 2018, pp. 2546-2554, doi: 10.1109/ INFOCOM.2018.8485982. keywords: {Delays;Peer-to-peer computing;Analytical models;Stochastic processes;Bitcoin}

[3] Nayak, Sambit & Narendra, Nanjangud & Shukla, Anshu & Kempf, James. (2018). Saranyu: Using Smart Contracts and Blockchain for Cloud Tenant Management. 857-861.10.1109/ CLOUD.2018.00121.

[4] https://miro.medium.com/max/1400/1*A8MLhHoq4rTYjh9jw6tNtQ.png

[5]Image source: (https://www.researchgate.net/figure/Proof-of-Work-Flowchart_fig6_331040157 ,cited at May 2024)

[6] Image source: (https://www.researchgate.net/figure/Proof-of-stake-flowchart_fig2_357842909 ,cited at May, 2024)