# Connectivity Standards Alliance Distributed Compliance Ledger and the IoT

September 2022

## Notice and Disclaimer



### Revision History

| Revision | Date | Comments |
|---|---|---|
| 1.0 | 20 September 2022 | First publication date |
| | | |
| | | |

# Table of Contents



# Document Authors


Asad Haque - Comcast
Alexander Shcherbakov - DSR Corporation


# I. Introduction

A new phenomenon is sweeping our lives; smart devices - large and small - are seamlessly

communicating with each other and providing relevant information to make the services and other devices we use more useful than ever. For example, we can tell from miles away if our home is saving energy with the thermostat set at the right temperature, if the porch light is on, or if the front door was left unlocked. This is possible because thermostats, bulbs, and door locks can communicate their status, receive commands, and act on them.

One contributing factor is that these devices employ a range of network protocols to communicate over the internet, requiring “cloud-based” services. What was once hailed as a convenience has become a burden. Using servers on the internet makes it easy for manufacturers to support their devices in geographically dispersed locations while still providing near real-time status or environmental readings, but at the cost of a fragile infrastructure dependent on the manufacturer's continued backend support. These devices are often inexpensive, easy to use, and quick to set up. Another exciting technology garnering much attention involves decentralized databases or ledgers that allow parties to independently validate information without involving a trusted central authority. Known as blockchain, this technology enables cryptocurrencies such as Bitcoin and Ethereum. Blockchain provides data anonymity, integrity, and immutability through decentralization. These properties can be used to improve the overall security of IoT devices.

# II. Distributed Compliance Ledger (DCL)

The Distributed Compliance Ledger (DCL), which was created by Connectivity Standards Alliance, is a cryptographically secure, distributed network that allows device manufacturers (vendors), official test houses, and Certification Centers to publish public information about a given device or class of devices. Based on blockchain technology, it allows participants to update relevant device information that is cryptographically signed.

The Connectivity Standards Alliance (Alliance) DCL is an industry-wide initiative to provide a cryptographically secure distributed ledger of certified IoT devices and their roots of trust, without one company or an entity in charge of the ledger. By using an underlying permissioned blockchain framework, the DCL benefits from the following properties:

- Multi-node network that is run by Alliance member companies.
- All transactions are Individually signed using pre-approved keys by the DCL trustees.
- As a blockchained ledger, the system fundamentally preserves an immutable record of transactions, which is authoritative, transparent, and publicly auditable.
- The participation of globally present Alliance members and the ease of adding new DCL nodes leads to accessibility and distribution of DCL data across geographical locations.
- Consensus protocol to ensure majority approval.

## Immutable and Tamper-Resistance (Integrity)

A block added to the blockchain ledger is attached to the preceding block by including the hash of the old block in the header of the new one.

- This structure ensures two key properties: **Immutability.** Once a block is added it cannot

be removed without orphaning the blocks after it.
- **Tamper-resistance.** Tampering with a block will change its hash, meaning a tampered block will point to a hash that does not exist, making it easy to recognize.

A new block is added after successful consensus protocol approval by a supermajority of authorized nodes participating in the DCL network.

### Distributed and Decentralized

Blockchain ledgers are decentralized and exist in multiple nodes that can perform operations against the blockchain ledger. Records are added and verified using a consensus protocol by a supermajority of the participating nodes in the ledger. The IoT software commissioners or controllers (and other device types) have the ability to query the aggregate ledger anytime and retrieve blocks they are interested in.

## III. DCL Industry Applications

The DCL provides a unique opportunity to address a myriad of issues to benefit consumers, developers, and manufacturers of devices and associated firmware.

### Counterfeit Device Protection

The IoT industry, in general, suffers from devices that purport to be from a certain manufacturer but are actually counterfeit. By utilizing the DCL, manufacturers can upload their public key or certificate to validate a signature of a random challenge sent to a device during commissioning. A counterfeit device will not be able to generate a signature that can be validated by the manufacturer's public key maintained in the DCL. This helps end-users feel secure about the authenticity of the devices they are installing and allows brands to maintain their brand promise.

### Counterfeit Firmware Protection

The DCL provides a trusted platform for manufacturers and developers to maintain firmware links and signatures of firmware hashes. Upgrade code can check the latest firmware released by manufacturers or developers, verifying its integrity before proceeding to installation.

### Publications of Trusted Roots

The DCL provides a platform where a list of trusted roots, also known as a PAA (Product Attestation Authority), can be maintained. Changes to the PAA list are committed if the request receives a supermajority approval from the trustees of the DCL. This provides a mechanism to distribute revocation information on a PAA should the supermajority of the DCL's trustees elect to revoke a PAA for either non-compliance or a security breach.

### Device Provenance

Consumers often rely on the belief that a device they are purchasing is directly from the manufacturer whose name is on the box. However, the supply chain and the multiple entities involved in getting a manufacturer-designed device made and into the end-user's hands depend on trust. This trust can be better assured by using features of the DCL that focuses on recording Device Provenance when purchasing a device. This information is securely posted on the DCL by the manufacturers and publicly available on the ledger.

### Protocol Conformance

The DCL provides a trusted source of device-type protocol conformance issued by a trusted third party. These conformances, in the form of certifications, are granted after successful tests conducted by independent labs and granted by the certification body overseeing the standard protocol. This information can be monitored real-time during device commissioning to ensure interoperability among devices that have been certified against a common protocol.

Furthermore, DCL supports the Certificate Transfer Program that allows a vendor with a certified device to transfer it to another company that then inherits the device and its certifications compliant with the CSA Certificate Transfer Program. This program supports OEM use cases where a vendor's customer is re-branding the vendor's product under its own company name while maintaining the Certified status of the product.

## IV. DCL Goals and Benefits

The DCL's benefits align with the Connectivity Standards Alliance's (Alliance) goals of providing a secure, independently verifiable source of information around IoT devices. The main objectives that are achieved are to have a canonical, secure, tamper-resistant, immutable framework for commissioning purposes.

The benefits of the DCL include:

- **Open-source and vendor-agnostic technologies:** the DCL is vendor-agnostic and leverages open-source frameworks that are stable and mature, which supports proper security practices, audits, maintenance, and community support.
- **Production-ready and proven technology framework:** the DCL is based on a mature permissioned ledger technology that is ready for production deployments.
- **Ability to deploy multiple new networks**: the DCL is designed to be extensible to allow for the flexibility in supporting new networks with ease, as may be needed as the Alliance adds new standards.
- **Proof-of-Authority BFT consensus protocol:** only approved Alliance member nodes can participate in the underlying consensus protocol. DCL requires consensus among

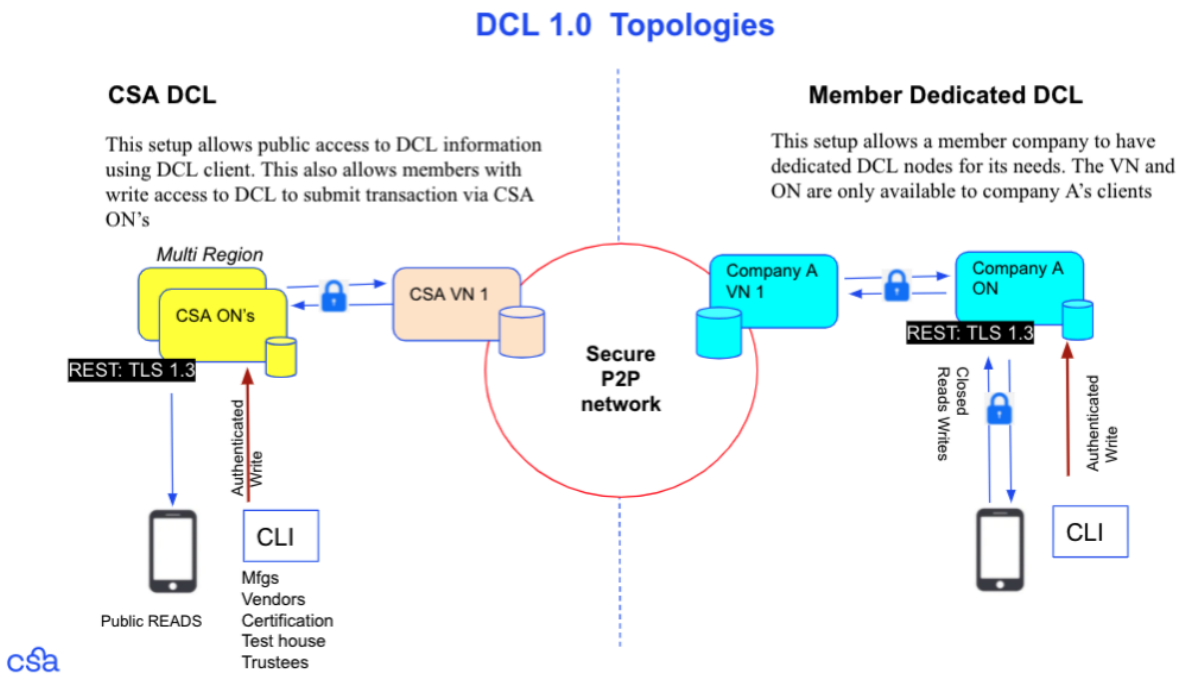
authorized nodes to be BFT (Byzantine Fault Tolerant). This measure reduces the risk of hackers attacking the network via one target/node with the intention to threaten the integrity of the ledger.

- **Permissioned write access:** the DCL is not secured by using a proof of work or typical proof of stake algorithms like the blockchains used by tokens or cryptocurrency. Rather, write access is permissioned and authorized by the Alliance which acts as the DCL's authority; this is a feature of the proof of authority algorithm. Transactions are authenticated with digital signatures. Every transaction must be signed by a key known to the ledger and associated to the sender.
- **Data replication and observer nodes**: the DCL features observer nodes, which contain a full replication of data but do not participate in consensus. These can be thought of as a caching mechanism that can improve availability and response times serving a geographic area. Organizations can utilize these nodes to efficiently and verifiably replicate data for DCL clients and applications.
- **Presence of a light client:** the DCL has multiple ways to perform a read operation. When a DCL user interacts with a trusted node from its organization, they can trust the query responses returned. A light client is used to query results from an untrusted node, employing cryptographic proofs without replication of data.
- **Scalability:** the DCL allows for the creation of a sufficient number of observer nodes to read replicated data in a short timeframe.
- **Performance:** the DCL can load hundreds of transactions per second. A single write request takes less than five to ten seconds to complete, and read request results typically return in under a second.
- **Maintainable solution:** the DCL is built using an open-source blockchain framework that is actively maintained by an extensive community. This reduces the costs required to address any potential unforeseen issues while availing the DCL to use new features/developments made to the underlying framework.
- **Low barrier to entry for new community members:** the DCL is abstracted away from blockchain and consensus-specific logic with an easy-to-use interface to increase adoption and use.

# V. How the DCL Works

## DCL Architecture

The DCL is a distributed secure network that consists of validator nodes (VN) and observer nodes (ON) operated by member companies and the Connectivity Standards Alliance. The figure below is a logical representation of the DCL. More node types can be defined to deploy a secure network with protections against DDoS attacks.



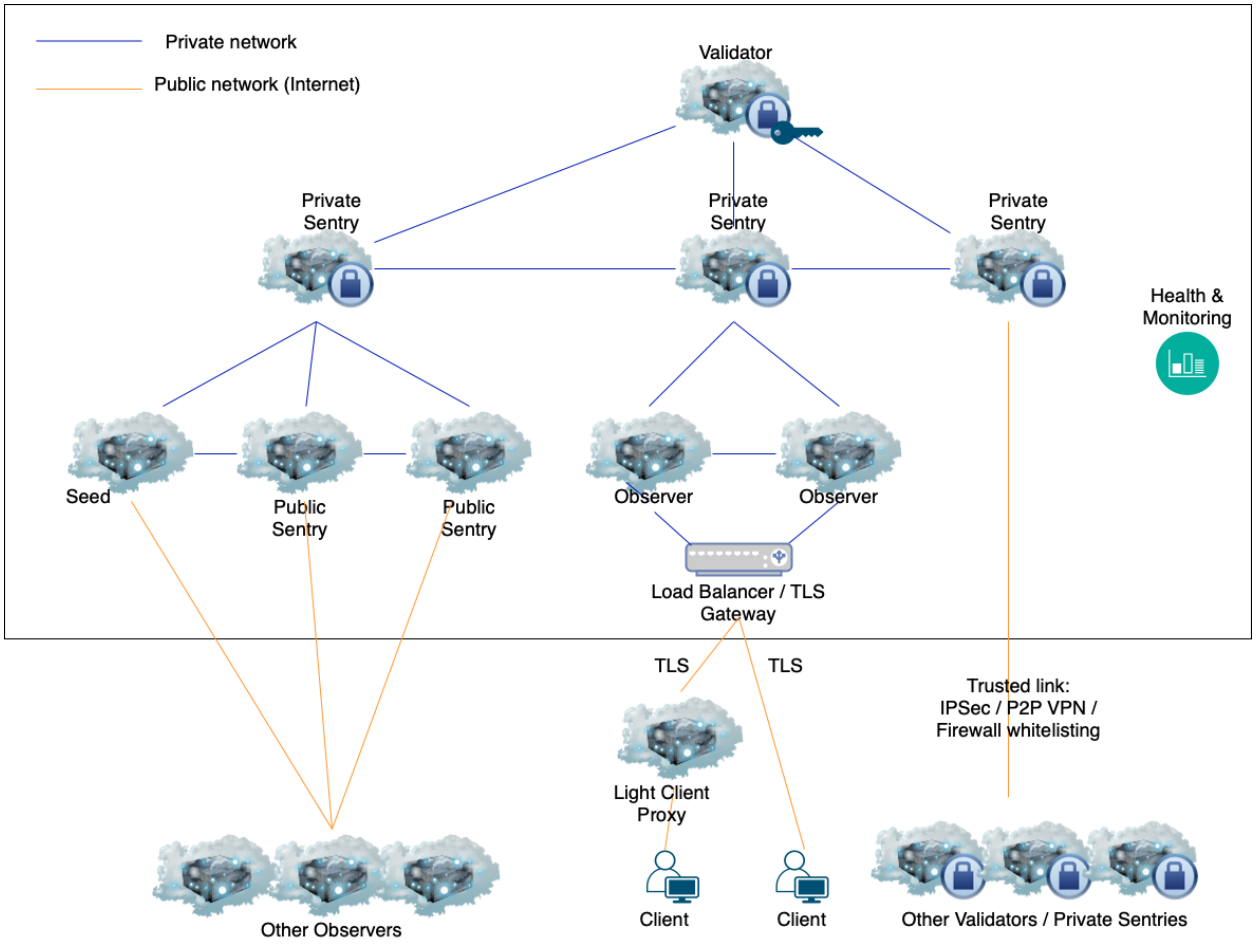

All full nodes maintain replication of all transactions as an immutable ledger that cannot be modified without authorization if more than two-thirds of the nodes are trustworthy.

To ensure the blockchain ledger is forming a valid blockchain, incoming write requests must be written in the same order on all nodes by means of a consensus protocol run by VNs. This consensus protocol guarantees that the network can process new write requests even if up to one-third of all validator nodes are down or malicious.

There is no need to set up and run a node to interact with the DCL. It is sufficient to just connect to one of the nodes via a DCL client.

## DCL Node Types

The DCL has three distinct node types: full node, seed node, and light client proxy.




**Full Nodes**

Most nodes in the DCL are Full Nodes and contain a full replication of the data. There are four (4) types of full nodes:

- **Validator Node (VN)** is a full node that participates in the consensus protocol utilizing an authorized cryptographic key. A VN is responsible for validating new records. In order to run a VN, an organization must have an approved account with a Node Admin role. It is recommended to restrict public access to VNs because they are used to writing transactions to the ledger. A better way to accept requests from the DCL client is via Observer Nodes.
- **Observer Node (ON)** is a full node that does not participate in the consensus protocol. ONs do not require an account with a Node Admin role to be created and member companies can govern access to their ONs.
- **Private Sentry Node** is a full node that wraps a Validator Node, representing it to other Validator or Public Sentry Nodes to protect against a Distributed Denial of Service (DDoS) attack.

- **Public Sentry Node** is a full node that wraps a Validator Node, representing it to other external Observer Nodes protecting against a Distributed Denial of Service (DDoS) attack.

### Seed Nodes

A seed node is an auxiliary node that provides a list of peer IP addresses that can be used for a connection because any new node joining the network requires the IP addresses of other nodes to connect to.

### Light Client Proxy

If DCL clients do not wish to run a VN or ON and do not trust any existing public full node, they can utilize a light client proxy node. A light client proxy serves as a proxy to untrusted full nodes. It does not contain a full replication of the ledger data, but verifies cryptographic proofs for read requests automatically, meaning returned data is valid and trusted.

## DCL Operations

The three primary actions on the DCL are adding nodes and submitting write and read requests to the ledger. Each node added catches up the ledger data in minutes.

- **Adding new nodes:** anyone can deploy an ON as long as there are nodes allowing connection to this ON, while only approved DCL clients can deploy a VN. A DCL client needs to have an approved DCL account with a Node Admin role to add a VN. One Node Admin can have only one VN added to the network.When adding a new node to the network, it is necessary for the node to catch up on all the data. This is usually time-consuming for longer-running networks and must be done from the genesis state. Fortunately, a node can utilize a state-sync feature, where data catch-up is performed at some trusted checkpoint, cutting the time to add a new node from hours to seconds.

- **Writing to the DCL:** the DCL is permissioned for write requests, so only approved DCL clients can write to the ledger. Any DCL client wishing to write to the ledger must generate both a private and public key and have an approved account. This account contains the DCL client's public key and role. Write requests are authorized through digital signatures. DCL clients sign transactions using their private key and the request is sent to a node that verifies the signature getting the corresponding public key from the ledger. If the signature is valid, the transaction is broadcasted to other peer nodes which also do the signature verification

- **Read from the DCL:** the DCL is permissionless for read operations. Any DCL client can read from the ledger if they have access to a Full Node. Neither a ledger account nor a key pair is needed to send a read request, which is anonymous and not signed. Read requests can be processed by any full node without a need to forward them to other nodes.

## DCL APIs

To interact with the DCL, it is sufficient to connect to one of the nodes via a REST API, gRPC, or Tendermint API.

- The **REST API** can be used for read requests and broadcasting signed transactions to the ledger.
- **gRPC** is used for generating and signing read and write requests.
- **Tendermint RPC** is used for generating and signing read and write requests. Tendermint RPC supports the creation of Light Clients meaning the application can work with untrusted nodes.

## DCL Client Applications

The DCL includes an array of client applications that connect to one of the full nodes described above via the DCL API:

- **Command-line Interface (CLI)** manages keys and sends read and write requests, utilizing Tendermint RPC. The CLI in DCL is not a Light Client and should be connected to trusted nodes or a light client proxy node.
- **Web App** allows for a Web App to browse the ledger and write requests. All write requests are signed via a built-in in-browser wallet.

## Information Currently Available in the DCL

The DCL contains the following standard information:

- **Vendor Info:** for example, company name, legal and preferred name, URL, etc.
- **Models and Model Software Versions:** models are identified by a unique combination of `VID` (vendor ID) and `PID` (product ID).
- **Model Certification:** the DCL can be used as a source of information to ascertain if a certain model's software versions (e.g., firmware) are certified by a certification center. If a security breach or violation has been found or reported the model's associated certificates may be revoked.
- **X.509 Certificates:** the DCL can distribute authorized X.509 Certificates (root and non-root).
- **Auxiliary Information:** the DCL has an array of required auxiliary transactions:
  - Validator Node transactions define the current set of VNs participating in consensus.
  - Account transactions contain roles and public keys for DCL clients wanting to write to the ledger. One DCL client can have multiple Account transactions. One account transaction contains just one public key but may contain multiple roles associated with the public key.
  - Upgrade transactions contain information about scheduled and completed updates of the DCL software.

## DCL Client Roles

The DCL restricts members to the roles they are authorized to perform. Individual Connectivity Standards Alliance (Alliance) members cannot act unilaterally or forge a transaction for a role they do not possess. This works to prevent rogue actors from bypassing the required approval steps that the Alliance mandates to protect its members from counterfeiters and otherwise non-compliant products, which may harm the reputation of Alliance brands.

- **Trustee:** these roles are assigned by the Board of Directors for the governance of the DCL. They can be used to approve new DCL clients, certificates, and software upgrades. Further, they can be used to revoke permissions.
- **Node Admin:** is a role assigned to a member that is selected to run a VN. Every VN is created by an approved Node Admin. A Node Admin can add only one VN. Node Admins can disable their own VNs.
- **Vendor:** is a manufacturer who is an Alliance member. All Alliance members can request and receive multiple vendor accounts. A single vendor can have multiple keys for different product models and versions.
- **Certification Center:** The Connectivity Standards Alliance Director of Certification is responsible for this role, which can submit the certification status of a product based on test results submitted by an authorized test lab to the Alliance. A Certification Center can provision, certify, and revoke model version certifications.

## DCL Carbon Footprint

The DCL network is not based on a proof of work consensus algorithm. Therefore validator nodes do not compete to win a hash rate-based competition, which leads to excessive power consumption. This allows the network to maintain a carbon footprint equal to a comparable computer network that involves digital signature validation. Furthermore, the number of validator nodes is capped at twenty-five thus allowing the DCL to maintain a predictable power consumption curve.

## DCL Technologies

Because of the DCL's stated goals and benefits and the availability of mature, open-source frameworks for DLT and blockchain, most DCL core logic is implemented from the following tools utilizing an active community and live production deployments.

**Tendermint Core**

Tendermint Core is a Byzantine Fault Tolerant (BFT) consensus engine with instant finality.

**Cosmos SDK**

Cosmos SDK is a modular open-source framework for building application-specific blockchains: public proof-of-stake (PoS) or permissioned proof-of-authority (PoA) systems. To address DCL use cases, the following customizations have been applied to the Cosmos SDK base module set:

- The validator module replaces staking and tokenomics-related modules to use role-restricted Connectivity Standards Alliance authorized certificates in order to provide a PoA instead of PoS consensus protocol.
- The dclauth module replaces Cosmos's auth and authz modules in order to provide a permissioned network.
- The DCL uses majority approval by trustees before a software upgrade can be achieved. This is based on a trustee proposing an upgrade using a software hash. It only takes effect if the supermajority of the trustees approves the proposed upgrade.

## VI. Future Enhancements

In the future, this decentralized and distributed ledger can enable further sets of exciting use cases. By participating in a blockchain distributed ledger, IoT devices can trust transactions that are cryptographically signed with their private keys. For example, a group of home devices can accept a homeowner's Access Control List (ACL) that is maintained on the DCL. The devices participate in the creation of this ACL by signing transactions with their secure keys. Later, they can accept an access request, as long as an ACL transaction exists that has been previously created by the same (target) device. This will greatly simplify maintenance, distribution, and security of ACL that govern access to a group of devices. Similarly, IoT devices can make decentralized decisions for granting access by relying on secure keying material stored locally on the device. For example, the requesting device must be part of a transaction that targets a device created and signed with its own private key. The IoT devices can have signed firmware that denies all access requests until it has been associated with a blockchain ledger. Once it is associated with a blockchain ledger, it will inherit security properties associated with the class of devices that are maintained in the ledger by the manufacturer. The Connectivity Standard Alliance's DCL is only the first step in the potentially global adoption of decentralized compliance ledger technology for the IoT as a whole.

Additionally, DCL SDK is a library for multiple popular programming languages and platforms that will be used to simplify the usage of the DCL API by various client apps (web, mobile, etc.).

## VII. Conclusion

The IoT is no longer an early adopters' proposition. It is crossing the chasm between early adopters and the pragmatist mainstream. These devices are having a profound impact by providing conveniences that enrich and enhance our lives. Similarly, blockchain ledgers show promise as a way to provide large technology consortiums and initiatives a decentralized, tamper-resistant way to distribute information on a platform. These systems provide features that would otherwise be harder to achieve, less scalable, or unavailable using traditional data models and centralized relational databases. By leveraging the promises of the blockchain and IoT, we put customers in charge of their own security by providing a transparent, canonical source of information about device provenance, certification status, and important setup/operation parameters.

# VIII. Acknowledgments

The authors of this whitepaper acknowledge the technical contributions of the following security subject-matter experts: Steve Hanna (Infineon), Darren Learmonth (Silicon Labs), and Aurash Kamalipour (Silicon Labs) who provided important feedback and review of this paper. Additionally, the authors acknowledge the administrative support of Genie Peshkova (DSR), Will Koerner (DSR), Lance Looper (Silicon Labs), and Ann Olivo (Silicon Labs).

# IX. References


- GitHub Repo: https://github.com/zigbee-alliance/distributed-compliance-ledger
  - Open source README: https://github.com/zigbee-alliance/distributed-compliance-ledger/blob/master/README.md
- Tendermint: https://tendermint.com
  - https://docs.tendermint.com/
  - https://github.com/tendermint/tendermint
  - https://github.com/tendermint/tendermint/blob/master/spec/consensus/consensus.md
  - https://arxiv.org/pdf/1807.04938v3.pdf
- Cosmos SDK: https://v1.cosmos.network
  - https://docs.cosmos.network/
  - https://github.com/cosmos/cosmos-sdk