# Nervos CKB: A Common Knowledge Base for Crypto-Economy


## Abstract

Nervos is a layered crypto-economic network. Nervos separates the infrastructure of a crypto-economy into two layers: a verification layer (layer 1) that serves as a trust root and a decentralized data layer (layer 2) that generates layer-2 applications. This paper focuses on the design of the verification layer.

This document provides an overview of the Nervos Common Knowledge Base (CKB), a public permissionless blockchain and layer-1 of Nervos. The Nervos CKB generalizes and improves the ideas behind the bitcoin UTXO model to extend the proof-of-work based bitcoin network with a flexible network, providing public, secure and censorship-resistant custody services for assets, identities and other common knowledge created in the Nervos network.


## Contents


0. Motivation
1. Overview
2. Cryptonomics
3. Consensus (Nakamoto)
   - Base Generation and Verification
   - Cell
   - Unit
   - Transaction
4. Consensus (PoW)
   - Data Cost and Cell Capacity
   - Computation Cost and Transaction Fees
5. Network
6. Summary
7. References
8. Appendix


## 0. Motivation

We want a peer-to-peer crypto-economy network.

In such a network, people can not only publish/purchase but also have incentives to do so. We need the ability to define, issue, transfer, and use assets in a peer-to-peer network to create such incentives. Bitcoin pioneered the first peer-to-peer crypto-economy.

Bitcoin[1] was the first public permissionless blockchain, designed to be used solely as peer-to-peer cash. Ethereum[2] extends the peer-to-peer cash system to a peer-to-peer transaction execution environment, which allows developers to write decentralized applications. However, the scaling, bootstrapping, and decentralization of Ethereum pose significant challenges. Bitcoin-like blockchains, including Ethereum, have network bandwidth and storage requirements that increase crypto-economic activity. Bitcoin-like blockchains also suffer from the notorious scalability problem. Proof-of-work based blockchains, including Ethereum, also suffer from the notorious scalability problem. Ethereum transaction processing capability cannot scale with the number of participants in the network, which severely limits their growth potential.

The blockchain community has proposed many scalability solutions in recent years, in general, we can divide these solutions into two categories: on-chain and off-chain. On-chain solutions, such as block size adjustments, sharding, and UTXO aggregation, do not change the underlying system. The consensus protocol is the core of a blockchain protocol, in which nodes exchange network messages and reach agreement eventually. A consensus is when almost by definition, because message exchange on a public and open network is slow and unstable. Nodes in a public and open network can also be offline, lying, or even malicious. Bitcoin-like blockchains, such as Bitcoin and Ethereum, have to probabilistically and reduce bandwidth requirements. Bitcoin decentralization due to high bandwidth and infrastructure costs, or "scale out" by sharding. The idea of sharding is to divide nodes into many small "shards", and ask each shard to process only a fraction of network transactions. Sharding is widely adopted by many projects. Many fears that sharding cannot work because sharding breaks atomicity. Sharding is well known for the complexity of shard coordination and cross-shard transactions, which result in a trusted environment, leads to performance degradation, and leads to security risks.

In contrast, off-chain scaling solutions acknowledge the inherent complexity of the consensus process. They recognize that consensus within a network should be as simple as possible, and propose to move most transactions and computations off-chain. Off-chain solutions, such as payment channels and hubs, payments by routing, and off-chain smart contracts, have to assume that participants have enough time to be known by every nodes in the network, except when they actually settled, in other words, when users want to turn their transactions into confirmations, they have to wait for a long period of time. Off-chain solutions, such as payment channels and hubs, with no smart contracts, support transactions with simple and predefined logic, such as fixed delivery and digital cash.

While we agree with the general ideas of off-chain scaling, we have found that there is no existing blockchain designed for it. For example, most off-chain solutions assume that participants have sufficient computation power to monitor the blockchain in real-time, but mobile devices have much less bandwidth than the limitations of the underlying Bitcoin protocol. Ethereum provides powerful programming ability, but its computation power is very limited, and its storage is very expensive. Most off-chain solutions require complex but time-consuming coordinations, which are easily attacked by a distributed flat peer-to-peer network. Decentralized applications might prefer off-chain consensus scripts and hosting all application to use CKB's consensus would be inefficient. Applications can choose the appropriate generation methods based on their particular needs. Nervos CKB generalizes and improves the ideas behind the bitcoin UTXO model to extend the proof-of-work based bitcoin network with a flexible network, providing secure and censorship-resistant custody services for assets, identities and other common knowledge created in the network.

Provide state generation methods (but are not limited to) the following:

- Local generation on the client: Generations run directly on the client's devices. Decentralized generations can implement the generation in any decentralized ways.
- Web services: Users may outsource their generation tasks to generation web services. Users can work with services may work with the SPV to keep gain more trust and liquidity for the generated assets. For example, game companies may define in-game items as assets in CKB. The game itself functions as a web service that generates new in-game data, which is then stored and stored in CKB.
- State channels: Two or more users may use peer-to-peer communication to generate new states.
- Decentralized exchanges: Users may use CKB as a decentralized exchange to exchange assets stored in CKB. Generation chains may be permissionless blockchains or permissioned blockchains. In each generation chain, nodes reach consensus in smaller scopes, providing better privacy and performance.

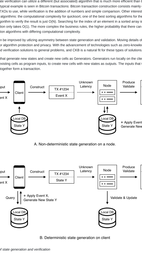



*Figure 1. Layered Architecture*

CKB exists as a Proof-of-Work based consensus, a NOSC if instruction set based virtual machine, a state model based on cells, a state-oriented economic model, and a peer-to-peer network. The Proof-of-Work based consensus makes the CKB a public and censorship-resistant service. The consensus of CKB is based on Nakamoto consensus a stateful Turing-complete programming model for blockchains, today state generation (or layer 2) on CKB practical. The CKB economic model is designed for common knowledge custody and long-term sustainability The CKB peer-to-peer network provides a simple and secure propagation network for blockchain data.

## 3. Consensus

CKB consensus is an improved Nakamoto consensus based on Proof-of-Work, that aims to achieve openness, correctness and high performance in distributed environments with network delay and Byzantine node faults.

Permissionless Blockchains are in competition with each other, anyone can join and leave freely, with no fortunes assumptions. There are several problems for traditional BFT consensus algorithms to solve. Sybil/ Sybil/ Sybil Attack introduced economic incentives and probabilistic consensus is since these problems. Nakamoto consensus uses Bitcoin's ledger as a clock, which allows the system to go 10 minutes in an hour to confirm transactions and leads to an unique canonical history.

Nervos CKB consensus is a Nakamoto consensus variant, which means it allows nodes to join and exit the network freely. Every node can participate in the consensus process rather by mining (to compete with other miners) or by validating (that the Proof-of-Work) to produce new blocks, or by verifying new blocks are valid. CKB uses an NOSC (non-of-six-hash) Proof-of-Work function, with the goals of distributing tokens as evenly as possible and making the network as secure as possible.

Correctness includes eventual consistency, availability, and fairness. Eventual consistency guarantees every node sees an identical copy of data. Availability makes sure the network keeps growing. Miner reward schemes aim to reasonable more. Fairness ensures mining nodes get fair returns for their efforts to keep the network functioning soundly.

High performance means the balance between the consensus delay between the submission of a request and the confirmation of its execution results, and transaction throughput. The number of transactions the system is capable of processing per second. Both of these measures depend on block size, block interval, and the underlying network communication protocol.

Please check the CKB Cryptonomics RFC for more details.

## 4. Programming Model

CKB provides a stateful Turing-complete programming model based on CKB VM and cell model.

| | Bitcoin | Ethereum | CKB |
|---|---|---|---|
| Instruction Set | Script | EVM | eBPF |
| Cryptographic Primitive | Opcodes | Precompile | Assembly |
| Statefulness | No | Yes | Yes |
| State Type | Ledger | General | General |
| State Model | UTXO | Account | Cell |
| State Verification | On-chain | On-chain | On-chain |
| State Generation | Off-chain | On-chain | Off-chain |

*Table 2. Comparison of Bitcoin, Ethereum and CKB Programming Model*

The CKB programming model consists of three parts:

- cell(s) generation (off-chain)
- state verification (CKB VM)
- state storage (CKB model)

In this model, decentralized application logic is split into two parts (generation and verification), running in different places. State generation logic runs off-chain on the client side to generate transactions, and state verification logic runs on-chain to verify that entire network. Developers can write their own state transition functions, or use the Nervos SDK. Transaction inputs are references to previous outputs, along with proofs to unlock them. The transaction outputs are the results of the state transition function. The state transition function is a function that takes in transaction inputs and outputs a set of new cells. The Nervos SDK provides a transaction type specified by a type ID, CKB VM executes these scripts and checks that inputs included in inputs to make sure the type is permitted to use referenced cells and the state transition is valid under specified application logic. In this way, all nodes in the network can agree on the validity of a state transition without knowing the details of the underlying application.

State in CKB is a first-class citizen, states are included in transactions and blocks and synchronized between nodes. Nervos CKB uses a UTXO based data model, which is different from Bitcoin and Ethereum. CKB VM can process functions with no internal state, which makes CKB scripts deterministic, simple/pure to operate and easy to parallelize transactions verification.

### 4.1 State Generation and Verification

Decentralized applications on Nervos separate the generation and verification of state. While these processes occur in different places, CKB provides the architecture that can guarantee that the generated state is indeed a valid state transition.

Utilizing the same algorithm on both generation and verification makes a straightforward choice that works for general problems. In this model, a user runs a program to generate a new state, and then sends the state and the program to a network of nodes to verify. The nodes run the same program to verify that the state is the result of running the program with the given inputs. This approach works for general purposes and can majorly jackpot for a transaction, and then broadcasted to the network. CKB nodes turn this into a state transition function, provided it has valid inputs and outputs, and then verify the state transition by running the corresponding scripts.

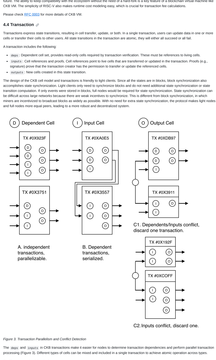
There are several advantages to this separation of state generation and validation:

- Generation becomes a competitive market of independent actors to find the optimal cost for pursuits of decentralized applications. If transactions include only user input and new states and the result of computation on nodes (see on Ethereum), the transaction creator cannot be certain about how much computation resources it takes to verify that a computation result. In CKB, users generate new states on the client side. They can confirm the new states before broadcasting their state transition to the network. The transaction outcome is certain: either the state transition function runs to a valid state, or it fails. In contrast, on the Ethereum blockchain, transaction fees are based on how much change it takes in CKB (Ergon 1).
- Parallelism: If transactions only include user inputs and new states are generated by nodes, then nodes will know what state is going to be accessed by the validation function before it runs. Transactions in Bitcoin have independent transaction contexts, meaning it's easy for developers to explicitly include primitive states and new states, nodes can see dependencies between transactions prior to verification, and can process transactions in parallel.
- Higher resource utilization: An application logic is split and run in different places, the verification can distribute computational workload more evenly across the network, and only require a small amount of computation to verify a state transition.
- Flexible state models: Given the same algorithm can work for different state transitions (an implementation generation and validation is different across apps, On the client side there is the flexibility to use any programming languages and libraries for better performance and libraries development.

In some scenarios, state verification can utilize a different (but equivalent) algorithm that is much more efficient than the one used for state generation. The most popular example is zero-knowledge proofs. Bitcoin-like transaction validation consists mainly of searching processing to identify aggregate UTXOs to use, while verification is the addition of numbers and simple comparisons. Other interesting examples include optimizations for machine learning models, or other computationally intensive algorithms. Nervos CKB supports these types of "off-chain" (CKBuild) but the algorithm to verify the result is on CKB). Searching for the order of an unknown set is a hard problem, but verifying the search result is easy, back for verification only takes O(1). The more complex the business rules, the higher probability that there can be asymmetry generation and validation algorithm offering computational complexity.

System throughput can be improved by utilizing asymmetry between state generation and validation. Mining details of computation is the client side, and the network only needs to verify the proof-of-work to see if a block is valid. Nervos CKB supports off-chain computation for complex proofs, so may find efficient generation and verification algorithms for these scenarios, and CKB is a natural fit for these types of solutions.

We refer to programs that generate new states and create new cells as Generators. Generators run locally on the client side (off-chain). They utilize user input and existing cells as program inputs, to create new cells with new states as outputs. The inputs that Generators use and the outputs they produce together form a transaction.




*A. Non-deterministic state generation on a node.*

*B. Deterministic state generation on client*

*Figure 2. Separation of state generation and verification*

### 4.2 Cell

Cells are the primary state units in CKB, within them users can include arbitrary states. A cell has the following fields:

- capacity: Since that all the cells occupied space in the state of all kinds nodes maintain, cells must be allocated capacity.
- data: State data stored in the cell. It could be simply, however the state types of a cell (including data), must always be less than or equal to capacity.
- type: State verification script.
- lock: Script that represents the ownership of the cell. Owners of cells can transfer cells to others.

A cell is an immutable object, no one can change it after creation. Every cell can only be used once, it cannot be used as input for two different transactions. If a cell is used as input in a transaction, it can be destroyed or updated with new data, type, and lock scripts. Transactions can be sending transactions, some people have cells with new data and lock scripts to them and transactions create cells that store old states atomically. The set of all current (or live) cells represents the latest version of all common knowledge in CKB, and the set of history (or dead) cells represents all historical versions of common knowledge.

CKB utilizes cores to transfer a cell's capacity all at once, or transfer only a fraction of a cell's capacity, which would in turn lead to more cells being created (e.g., a cell of capacity 100 can be used to create two cells with capacities 50 and 50).

Two kinds of scripts (type and lock) are executed in CKB VM. CKB VM executes the type script when a cell is created in a transaction output, and executes the lock script when a cell is used in a transaction input. The lock script is used to verify the ownership of a cell, and the type script is responsible for ensuring a state. It makes sure the state transition is valid and appropriate (according to logic or transfer for a cell). If the transaction script returns true, the user is allowed to transfer the cell or update its data according to validation rules that are specified by the type script.

The type and lock scripts are similar, both are composed of:

- Upgradable cryptography: Anyone can deploy useful cryptography libraries written in languages such as C or C++ and use them in type and lock scripts. In CKB, the Nervos team will provide a recommended cryptographic suite and have to choose any cryptographic signature scheme they'd like to use in type transactions.
- Multisig: Users can require that a cell is unlocking or more complex. type scripts.
- Lending: Cell owners can lend cells for others to use while still maintaining their ownership of the cells.

The Cell model is a more general model compared to the UTXO or Account model. Both the UTXO and the Account model can express relationships between assets and their owners. The UTXO model defines ownership of assets (both the lock script, while the Account model defines ownership of a state (the owner field in a smart contract). The Cell model can express ownership of both assets and states. Cells can store state storage matters for decentralized applications, and the lock script can be used to verify the ownership of the state. The Cell model can also be used to understand and analyze dependencies and identifiers well, but it presents challenges to processing transactions in parallel. The Cell model with type and lock script takes the best of both models to provide a more general model.

### 4.3 VM

CKB VM is a RISC-V instruction set based VM for executing type and lock scripts. It uses only standard RISC-V instructions, to maintain a standard compact RISC-V software implementation which can embrace the broadest individual support. CKB implements cryptographic primitives as extensions to the standard RISC-V instruction set, to accelerate cryptographic primitives speed. Its also possible to add more instructions such as current transaction and general blockchain primitives. CKB VM uses Berkeley's TinyGo compiler for each instruction, and provides stdio syscalls introduced during transaction verification to help means determine transaction flows.

Existing blockchains hardcode cryptographic primitives in the protocol. For example, Bitcoin has special cryptographic opcodes such as OP_CHECK*, and Ethereum uses special 'precompiled' contracts located at a special address (e.g. 0000000000000000000000000000000000000001) to support cryptographic operations such as secp256k1. To add new cryptographic primitives to these blockchains, we can only soft-fork (or hard-fork to save upgrades to support new primitives) or hard-fork.

CKB VM is a cryptographic agnostic instruction set. Bitcoin would need to add a secp256k1 opcode hardcoded in CKB VM. New cryptographic primitives can always be deployed and used by scripts like an ordinary library. Being a RISC-V standard compliant implementation means compilers can easily add new cryptographic primitives as well-defined assembly extensions to the RISC-V ISA. Nervos CKB VM will also implement the default hash function and public key signature and multi-signature verification algorithm. Being crypto-agnostic allows independent application developers or Nervos to use any new cryptography (such as Schnorr signatures, BLS signatures, and ZKSNARKs/zkSTARKs) that first without affecting the base layer. Bitcoin-like blockchains can't do this and need to wait for a hard-fork or soft-fork.

CKB VM chooses a hardware targeting 64-bit because blockchain is hardware-like software. Thought its creation is easy as software, its performance is limited by its hardware specifications. Most CPUs today are 64-bit, and 64-bit provides a reasonable balance between performance and future. The ability to keep compatibility with the mainstream hardware will help CKB VM to take advantage of future hardware improvements. CKB VM implements 64-bit instruction set. The overhead of RISC-V 64-bit instruction set is minimal.

Please check RFC0003 for more details of CKB VM.

### 4.4 Transaction

Transactions express state transitions, rooting in cell transition, update, or both. In a single transaction, users can update data in or mint cells or transfer their cells to other users. Inputs and dependencies of the transaction are unique, they will either all consumed or all fail.

A transaction includes the following:

- deps: Dependent cell set, provides read-only cells required by transaction verification. These must be references to living cells.
- inputs: Cell references and proofs. Cell references point to live cells that are mentioned or updated in the transaction. Proofs (e.g., signatures) satisfy the lock scripts of the referenced cells, and authorize the transaction to update the cells.
- outputs: New cells created in the state transition.

The design of the CKB cell model and transactions is friendly to light clients. Since all the states are in blocks, block synchronization also accomplishes state synchronization. Light clients can easily and securely verify the existence of a specific cell using a Merkle proof state synchronization or state transition computation. If only owners care about a specific cell, light clients can easily request for state synchronization. State synchronization can be done by downloading block headers and block bodies, and then verifying the Merkle proofs. Light clients can also download block headers and block bodies, and then verifying the state transition. With no need for extra state synchronization, the protocol makes light nodes and full nodes move equal peers, leading to a more robust and decentralized system.




*Figure 3. Transaction Parallelism and Conflict Detection*

The deps and inputs in CKB transactions make it easier for nodes to determine transaction dependencies and perform parallel transaction processing (Figure 3). Different types of cells can be mined and included in a single transaction to achieve atomic operation across types.

## 5. Economic Model

A well-designed economic model should incentivize all participants to contribute to the success of the crypto-economy and maximize the utility of the blockchain.

The CKB economic model is designed to motivate users, developers and node operators to work toward the common goal of common knowledge custody. The aspect of the CKB economic model is state instead of computation, by using cell capacity and transaction fees as incentives for stakeholders.

Please check the Nervos Economics RFC for more details.

### 5.1 State Cost and Cell Capacity

The creation and storage of states on the CKB cost nervos. The creation of new states needs to be verified by full nodes (which must download and process all data), and be stored in the canonical chain (which is currently held by miners). Verification and storage consumption bandwidth, computation and storage resources. Nervos plans to lower the costs of verification, occupying storage space and bandwidth.

In CKB, cells are basic storage units of state. A cell owner can use the cell to store state himself or lend it out to others. Because a cell's capacity can be used only by the owner of the cell, owners can be certain that anything they themself would give up the capacity for, such as lent out or sold, is returned to them. Selling the capacity our jetton in CKB is in other words. With this opportunity cost, users pay for storage with a cost that is proportional to the amount of resources they use. Nervos CKB uses cell capacity to measure the storage space a cell occupies, and requires that the capacity of a cell is equal to the number of bytes occupied by the cell. This ensures that the state size is not free, and that it is proportional to its opportunity cost. Nervos CKB uses a flexible and secure model to price state. Cell capacity measures the storage space a cell occupies, and the capacity of a cell is backed by the same amount of native currency. This ensures that the state size is not free, and that it is proportional to its opportunity cost. Nervos CKB uses a flexible and secure model to price state. The capacity of a cell is backed by the same amount of native currency. This ensures that the state size is not free, and that it is proportional to its opportunity cost. Nervos CKB uses a flexible and secure model to price state.

Nodes can be incentivized to store the state. Nervos CKB uses a flexible and secure model to price state. The capacity of a cell is backed by the same amount of native currency. This ensures that the state size is not free, and that it is proportional to its opportunity cost. Nervos CKB uses a flexible and secure model to price state. The capacity of a cell is backed by the same amount of native currency. This ensures that the state size is not free, and that it is proportional to its opportunity cost. Nervos CKB uses a flexible and secure model to price state.

Nodes can be incentivized to store the state. Nervos CKB uses a flexible and secure model to price state. The capacity of a cell is backed by the same amount of native currency. This ensures that the state size is not free, and that it is proportional to its opportunity cost. Nervos CKB uses a flexible and secure model to price state. The capacity of a cell is backed by the same amount of native currency. This ensures that the state size is not free, and that it is proportional to its opportunity cost. Nervos CKB uses a flexible and secure model to price state.

### 5.2 Computation Cost and Transaction Fees

Updating a cell's state is broadcasting and verifying nervos transaction fees. Miners can sort the transaction fee from that they are willing to accept based on CKB VM cycles and state cell state changes in transaction verification, allowing the market to determine transaction fees. With the programming model, developers can also optimize their scripts to reduce cycles, increasing capacity efficiency.

As cell capacity is the only native asset in CKB, it is the most convenient asset when users want to pay transaction fees. However, users can also pay transaction fees using other assets, such as xCKB, sudt, or other assets issued on top of CKB. Transaction fees using other assets are allowed in CKB because its economic model and native asset are not tied to computation, but rather, decoupled. Cycles are used only as a means of pricing transaction fees, this primary function is secure common knowledge storage, which can lower state and hold long-term. Payment network computation costs are very low, so transaction fees are relatively low.

Restricting the transaction fee payment method to a blockchain's native asset is a significant obstacle preventing blockchains' mass adoption. This requires users to acquire native tokens before they can use the blockchain. Bitcoin-like blockchains do not have this limitation, allowing users to pay fees on behalf of their users and allowing payment with any well-defined assets. CKB can provide a better experience in this regard, because it can support many types of assets.

Please check the Nervos CKB Ergon, Paper RFC RFC00 for details of the economic model.

## 6. Network

We can categorize CKB nodes into three types:

- Mining Node: They participate in the CKB consensus process. Mining nodes collect new transactions, package them into blocks and broadcast these blocks to the network. Mining nodes verify all received blocks to ensure they follow the consensus rules. Mining nodes store the current set of UTXOs.
- Full Node: They verify new blocks and transactions, relay blocks and transactions, and select the chain fork on which they agree. Full nodes are the backbone of the network.
- Light Node: They hold full headers, only subscribe and store a subset of cells that they are concerned with. They use retrieved nodes to send transactions and query the blockchain state. Light nodes do not participate in the CKB consensus process, they do not relay blocks.

Bitcoin-like blockchains, in which, only full nodes can participate in the network, have to be operated by technical users. Bitcoin's simplified payment verification (SPV) protocol allows for lightweight clients, but they have to trust a special full node. Nervos CKB uses a layer-2 network to support light clients and improve the security of light clients. Light clients trade collateral for the ability to use the network. Light clients can be incentivized to run a full node, leading to a much better user experience. In a mature crypto-economy network, the largest group of nodes would be light nodes, followed by full nodes and mining nodes. Because light clients can be incentivized to run full nodes, the network can be secured by a large number of full nodes. Because full nodes can be incentivized to run mining nodes, the network can be secured by a large number of mining nodes. Light clients can be incentivized to run full nodes, full nodes can be incentivized to run mining nodes, and the network can be secured by a large number of mining nodes. Light clients trade collateral for the ability to use the network. Light clients can be incentivized to run full nodes, leading to a much better user experience. In a mature crypto-economy network, the largest group of nodes would be light nodes, followed by full nodes and mining nodes. Because light clients can be incentivized to run full nodes, the network can be secured by a large number of full nodes. Because full nodes can be incentivized to run mining nodes, the network can be secured by a large number of mining nodes. Light clients can be incentivized to run full nodes, full nodes can be incentivized to run mining nodes, and the network can be secured by a large number of mining nodes. Light clients trade collateral for the ability to use the network. Light clients can be incentivized to run full nodes, leading to a much better user experience. In a mature crypto-economy network, the largest group of nodes would be light nodes, followed by full nodes and mining nodes. Because light clients can be incentivized to run full nodes, the network can be secured by a large number of full nodes. Because full nodes can be incentivized to run mining nodes, the network can be secured by a large number of mining nodes. Light clients can be incentivized to run full nodes, full nodes can be incentivized to run mining nodes, and the network can be secured by a large number of mining nodes.

## 7. Summary

We envision a world crypto-economy and CKB is its base layer. CKB is the decentralized trust root of this crypto-economy. It ensures the security of the trustless activities of the upper layers. It's a common knowledge custody network, in which states are verified by global consensus and stored in a decentralized way. CKB is a public, open, and permissionless blockchain. It's secured by Nakamoto consensus and Proof-of-Work and its design focuses on states rather than computation. CKB, users and developers can define, issue, transfer and store crypto-assets, they can also create digital identities and utilize these identities in the crypto-economy. Only our imagination is the boundary of its use.

## 8. References


1. Satoshi Nakamoto, "Bitcoin A Peer-to-Peer Electronic Cash System", 2008.
2. Vitalik Buterin, "Ethereum Next Generation Smart Contract and Decentralized Application Platform", 2014.


## 9. Appendix

*Common Knowledge is the knowledge that's accepted by everyone in a community. Participants in the community not only accept the knowledge themselves but know that others in the community also accept the knowledge.*

In the past, common knowledge was scattered across individuals minds, and its formation was a time-consuming and required communication and confirmation. Today, with the advancement of cryptography and distributed ledger technology, algorithms and machines are replacing humans as the common knowledge custodians. Blockchains are a kind of common knowledge. Every peer, or node, in the blockchain network maintains a local copy of the common knowledge, or a piece of common knowledge.

Blockchains are common knowledge bases. Participating in a blockchain network implies accepting and helping validate the common knowledge contained in it. Blockchain data structures are tamper-evident, and the consensus algorithm ensures that it is hard for a malicious actor to alter the latest data in the blockchain.

*The various ways in which the knowledge on which people base their plans is constructed is them is the crucial problem for any theory explaining the economic process, and the problem of what is the best way to utilizing knowledge initially dispersed among all the people is at least one of the central questions of economic policy – or of devising an efficient economic system.*

*The Use of Knowledge in Society, Friedrich A. Hayek, 1945*