CIPC

WHITE PAPER

CIPC-Cosmic Interstellar Public Chain

Large-scale commercial high-performance blockchain platform

V 1.0 SEPT,2022

Sequence

Now, the blockchain has developed to a market value of trillions of dollars, but limited by performance and scalability, the next-generation breakthrough platform is urgently needed. The goal of Cosmic Interstellar Public Chain is to use modern cryptography, communication technology, and distributed computing as It can carry decentralized financial, social, e-commerce, search, and storage business applications, and provide a large-scale commercial high-performance blockchain ecological platform.

This introductory white paper was originally published in 2018 by jeffrey, founder of Cosmic Interstellar Public Chain. It is worth noting that the Cosmic Interstellar Public Chain, like many community-driven open source software projects, has grown since its initial inception. Although several years old, we continue to maintain this document because it continues to serve as a useful reference and an accurate representation of the Cosmic Interstellar Public Chain and its vision. To learn about the latest developments in the Cosmic Interstellar Public Chain and how changes to the protocol are being made, we recommend you use this whitepaper.

1. NEXT-GENERATION LARGE-SCALE COMMERCIAL HIGH-PERFORMANCE BLOCKCHAIN ECOLOGICAL PLATFORM

Bitcoin, developed by Satoshi Nakamoto in 2009 and often hailed as a radical development of money and money, was the first example of a digital asset that simultaneously had no backing or intrinsic value, and no centralized issuer or control By. But another (possibly more important) part of the Bitcoin experiment is the underlying blockchain technology as a distributed consensus tool, and attention is rapidly shifting to another aspect of Bitcoin. Commonly mentioned applications include the use of blockchain's on-chain digital assets to represent custom currencies and financial instruments ("color coins"), ownership of related physical devices ("smart assets"), non-fungible assets such as domain names ("Namecoin"), as well as more complex applications involving passing arbitrary rules.

(Smart Contracts) Even a piece of code in a blockchain-based Decentralized Autonomous Organization (DAO) directly controls a digital asset. What Cosmic Interstellar Public Chain intends to offer is a blockchain with a built-in full-fledged Turing-complete programming language that can be used to create "contracts" that can be used to encode arbitrary state transition functions, allowing users to create any of the above systems, and Many other features we haven't thought of yet, just write logic in a few lines of code. The goal of supporting the next generation of large-scale commercial use of the blockchain platform is to support tens of billions of devices with concurrent connections, second-level confirmation, and high-performance blockchain ecological capabilities with hundreds of thousands of transactions per second.

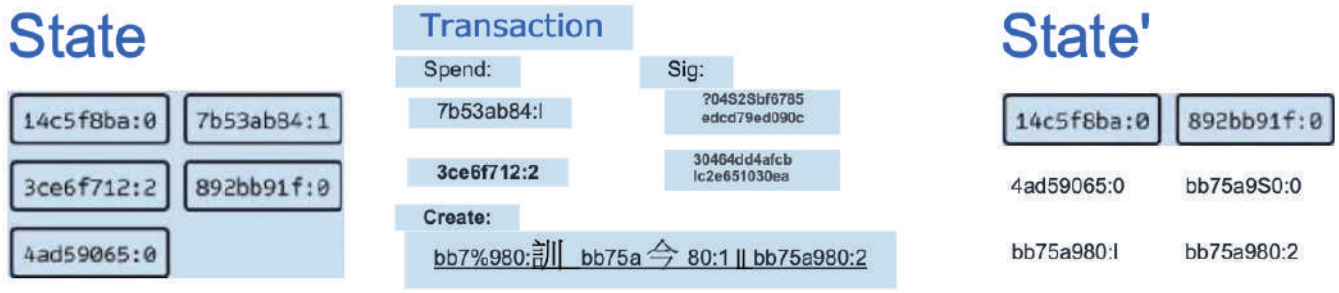
2. INTRODUCTION TO BITCOIN AND EXISTING CONCEPTS

2.1 History

The concept of decentralized digital currencies and alternative applications such as property registration have been around for decades. Anonymous electronic cash protocols of the 1980s and 1990s largely relied on a cryptographic primitive called Chaumian blinding to provide a high degree of privacy for money, but these protocols were largely unreliable due to their reliance on a central intermediary. can get attention. In 1998, Dai Wei's b-money became the first proposal to propose the idea of creating money by solving computational puzzles along with decentralized consensus, but the proposal lacked any details on how to actually implement decentralized consensus. In 2005, Hal Finney introduced the concept of reusable proof-of-work, a system that used the idea of b-money with Adam Back's computationally difficult Hashcash puzzle to create the concept of a cryptocurrency, but because it relies on Trusted computing as backend. In 2009, Satoshi Nakamoto first implemented a decentralized currency in practice, combining established primitives for managing ownership via public key cryptography with a consensus algorithm for tracking who owns coins, known as "Proof of Work". combine.

The mechanism behind Proof of Work is a breakthrough in the field because it solves two problems at the same time. First, it provides a simple and moderately efficient consensus algorithm that allows nodes in the network to jointly agree on a set of canonical updates about the state of the Bitcoin ledger. Second, it provides a mechanism that allows free entry into the consensus process, solving the political problem of deciding who can influence the consensus, while preventing sybil attacks. It does this by replacing formal barriers to participation, such as requiring it to be registered as the only entity on a specific list, with an economic barrier - the weight of an individual node in the consensus voting process is proportional to the computing power the node brings. Since then, an alternative approach, called proof-of-stake, has been proposed that computes a node's weight proportional to its currency holdings rather than computing resources; a discussion of the relative merits of the two approaches goes beyond out of the scope of this article, but it should be noted that both methods can be used as the backbone of cryptocurrencies.

2.2 Bitcoin as a state transition system



From a technical point of view, the ledger of a cryptocurrency such as Bitcoin can be thought of as a state transition system in which there is a "state".

Consists of the ownership state of all existing bitcoins and a "state transition function" that accepts states and transactions and outputs a new state as a result. For example, in a standard banking system, the state is the balance sheet, a transaction is a request to move $X from A to B, and the state transition function decrements the value in A's account by $X, and moves the value in B's account by $X. Value increases by $X. If the balance of the first account in A's account is less than $X positions, the state transition function returns an error, so

it can be formally defined:
APPLY(S,TX) -> S' or ERROR
APPLY({ Alice: $50, Bob: $50 },"send $20 from Alice to Bob") = { Alice: $30, Bob: $70 }

In the banking system defined above: but:
APPLY({ Alice: $50, Bob: $50 },"send $70 from Alice to Bob") = ERROR

"**State**" in Bitcoin refers to the collection of all mined but not yet spent coins (technically, "unspent transaction outputs" or UTXOs), each UTXO has a denomination and an owner (by A 20-byte address definition, essentially an encrypted public key. 1). A transaction contains one or more inputs, each containing a reference to an existing UTXO and a cryptographic signature produced by the private key associated with the owner's address, and one or more outputs, each containing a new UTXO to be added to. UTXOs.

The state transition function APPLY(S, TX) -> S' can be roughly defined as follows:

1. For each input in TX:

Returns an error if the referenced UTXO is not in S.

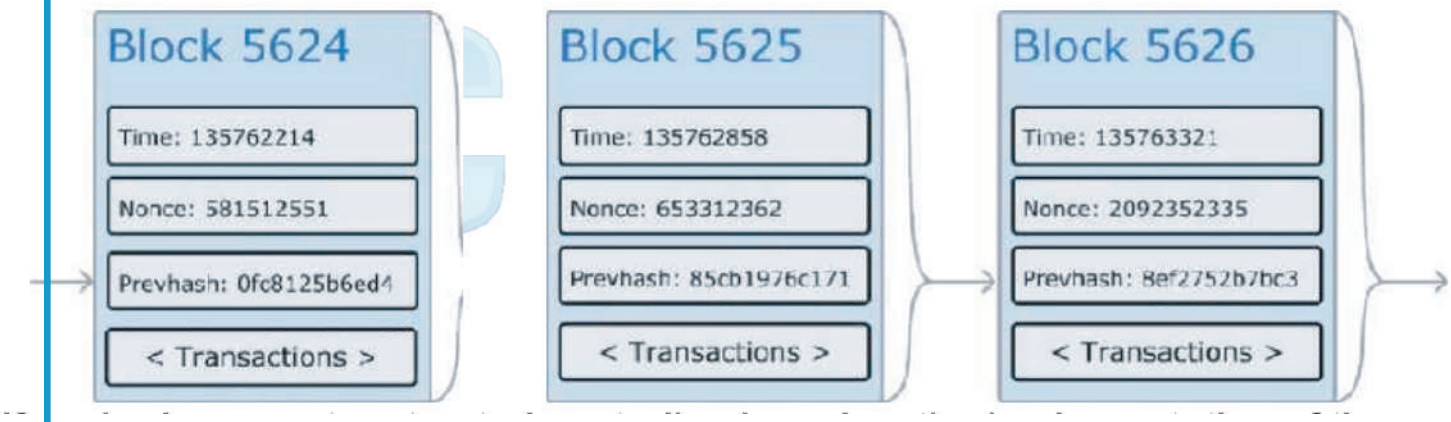
Returns an error if the provided signature does not match the owner of the UTXO.

2. Ifthesumofthedenominationsofallinput UTXOsislessthanthesumofthe denominations of all output UTXOs, return an error.

3. Return to S', delete all input UTXOs, and add all output UTXOs.

The first half of the first step prevents transaction senders from spending coins that don't exist, the second half of the first step prevents transaction senders from spending other people's coins, and the second step enforces protection of value. In order to use it for payment, the protocol is as follows. Suppose Alice wants to send 11.7 BTC to Bob. First, Alice will look for a set of available UTXOs she owns that add up to at least 11.7 BTC. In fact, Alice will not be able to get the exact 11.7 BTC. Say the smallest number she can get is 6 + 4 + 2 =12. Then, she creates a transaction with these three inputs and two outputs. The first output will be 11.7 BTC, with Bob's address as its owner, and the second output will be the remaining 0.3 BTC "change", owned by Alice herself.

2.3 Mining



If we had access to a trusted centralized service, the implementation of the system would be very simple; just use the centralized server's hard drive to keep track of the state, which could be coded exactly as described. However, with Bitcoin, we are trying to build a decentralized monetary system, so we need to combine a state transition system with a consensus system to ensure that everyone agrees on the order of transactions. Bitcoin's decentralized consensus process requires nodes in the network to continually try to produce packets of transactions called "blocks." The network is designed to produce approximately one block every ten minutes, each block containing a timestamp, a nonce, a reference to the previous block (i.e. its hash), and a list of all transactions that have occurred since the previous one. blocked. Over time, this creates a continuously growing "blockchain" that is constantly updated to represent the latest state of the Bitcoin ledger. The algorithm for checking whether a block is valid, represented in this example, is as follows:

If we had access to a trusted centralized service, the implementation of the system would be very simple; just use the centralized server's hard drive to keep track of the state, which could be coded exactly as described. However, with Bitcoin, we are trying to build a decentralized monetary system, so we need to combine a state transition system with a consensus system to ensure that everyone agrees on the order of transactions. Bitcoin's decentralized consensus process requires nodes in the network to continually try to produce packets of transactions called "blocks." The network is designed to produce approximately one block every ten minutes, each block containing a timestamp, a nonce, a reference to the previous block (i.e. its hash), and a list of all transactions that have occurred since the previous one. blocked. Over time, this creates a continuously growing "blockchain" that is constantly updated to represent the latest state of the Bitcoin ledger. The algorithm for checking whether a block is valid, represented in this example, is as follows:

Checks that the previous block referenced by this block exists and is valid. Check if the timestamp of this block is greater than the timestamp of the previous block. Check that the proof of work on the block is valid less than 2 hours into the next 2 hours. Let S[0] be the state at the end of the previous block. Suppose TX is a list of block transactions with n transactions. For all i in 0...n-1, set S[i + 1] = APPLY(S[i], TX[i]) If any application returns an error, exit and return false. Returns trueandregistersS[n]asstateattheendoftheblock.

Essentially, each transaction in the block must provide a valid state transition from the canonical state prior to executing the transaction to some new state. Note that state is not encoded in the block in any way. It is purely an abstraction for validating nodes to remember, which can only be calculated (safely) for any block by starting from the genesis state and applying each transaction in each block in order. Also, note that the order in which miners include transactions in CIPC-Cosmic Interstellar Public Chain blocks matters; if there are two transactions A and B in a block such that B spends the UTXO created by A, then if A precedes B, the block block is valid.

A list of valid conditions that exist above that cannot be found in other systems is the requirement of "Proof of Work". The exact condition-isthatthedoubleSHA256hashofeachblock(tobetreated as a 256-bit number) must be less than the dynamically adjusted target, which is around 2187 at the time of writing. The purpose of this is to make it computationally "difficult" to prevent creation, thereby preventing sybil attackers from rebuilding the entire blockchain to their liking. Because SHA256 is designed to be a completely unpredictable pseudorandom function, the only way to create a valid block is trial and error, repeatedly incrementing the nonce and seeing if the new hash matches.

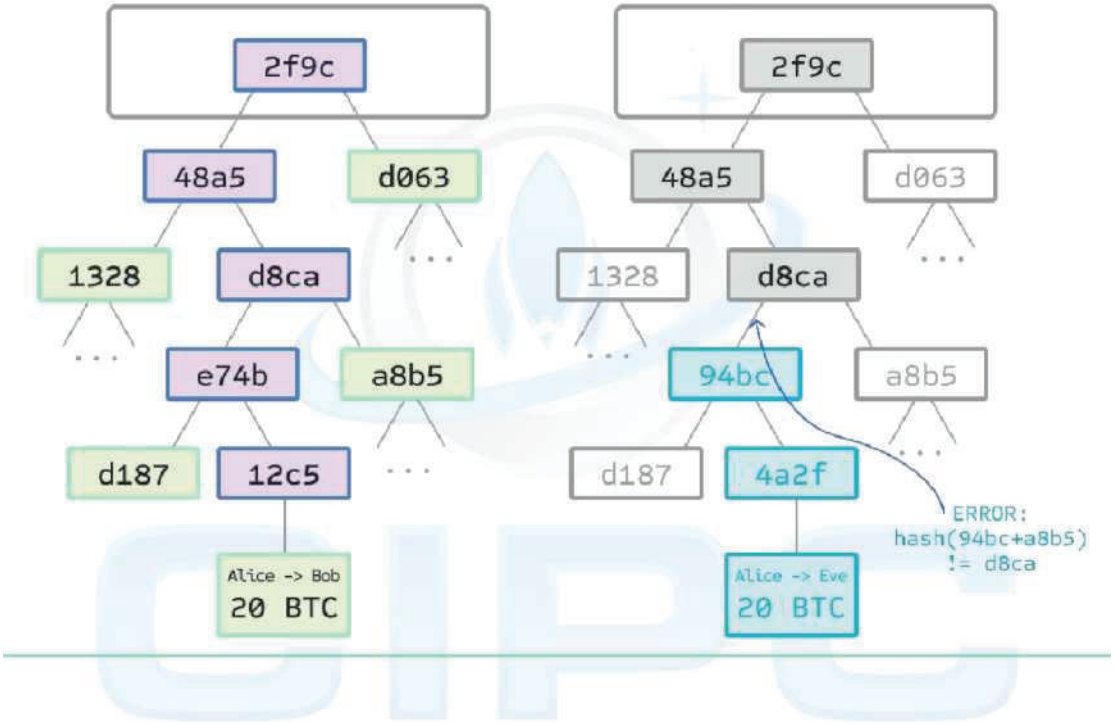
Before the current target of ~2187, the network has to try an average of 269 times to find a valid block. Typically, the network recalibrates the target every 2016 blocks so that, on average, a node in the network produces a new block every ten minutes. To compensate miners for their computational workload, each block miner is entitled to include a transaction that provides 12.5 BTC for free. Also, if the total denomination of the inputs for any transaction is greater than the total denomination of the outputs, the difference is also distributed to the miners as a "**transaction fee**". By the way, this is also the only mechanism for issuing BTC. The state of origin does not contain any coins at all.

To better understand the purpose of mining, let's examine what happened to a malicious attacker. Since Bitcoin's underlying encryption is known to be secure, attackers will target a part of the Bitcoin system that is not directly protected by encryption: the order of transactions. The attacker's strategy is simple: send 100 BTC to the merchant in exchange for some product (preferably a fast-delivered digital item) Wait for the product to be delivered to generate another transaction, send the same 100 BTC to himself to try to convince the network that he is right Your own transaction is the first transaction. Once step (1) is performed, a few minutes later, some miners include the transaction in a block, such as block number 270. After about an hour, five more blocks will be added to the chain after this block, each of which indirectly points to and thus "**confirms**" the transaction.

At this point, the merchant will accept the finalized payment and deliver the product; since we assume this is a digital good, the delivery is instant. Now, the attacker creates another transaction that sends 100 BTC to himself. If the attacker simply released it into the wild, the transaction would not be processed; the miner would try to run APPLY(S, TX) and notice that TX consumes UTXOs that are no longer in that state. So instead, the attacker creates a "**fork**" of the blockchain, first mining another version of block 270 that points to the same block 269 as the parent, but uses the new transaction in place of the old one.

Since the block data is different, the proof-of-work needs to be redone. Also, the attacker's new version of block 270 has a different hash, so the original blocks 271 to 275 do not "**point**" to it; therefore, the original blocks 271 to 275 do not point to that block. Therefore, the original chain and the attacker's new chain are completely separate. The rule is that the blockchain with the longest fork is the truth, so legitimate miners will work on the 275 chain, while only attackers will work on the 270 chain. In order for an attacker to be able to make his own blockchain the longest, he needs to have more computing power than the rest of the network combined to catch up (hence the "51% attack").

2.4 Merkle Tree



Left: Showing only a small number of nodes in the Merkle tree demonstrates the validity of branching.

Right: Attempting to change any part of the Merkle tree will eventually lead to inconsistencies somewhere along the chain.

An important scalability feature of Bitcoin is that the block is stored in a multi- 13 layered data structure. The "**hash**" of a block is really just the hash of the block header, about 200 bytes of data that contains a timestamp, a nonce, the hash of the previous block, and the root hash of a data structure called a Merkle tree that stores all transactions Hope is in the block. A Merkle tree is a binary tree that consists of a set of nodes, at the bottom of the tree a large number of low-level nodes that contain the underlying data, and a set of intermediate nodes, where each node is two of its children The hash of the node, and finally a root node, which also consists of the hashes of its two children, representing the "**top**" of the tree.

The purpose of Merkle trees is to allow data in blocks to be passed around piecemeal: nodes can only download headers for blocks from one source, and a small part of the tree associated with them from another source, and still be assured that all data is correct. This works because the hash propagates upwards: if a malicious user tries to swap fake transactions into the bottom of the Merkle tree, this change will cause a change to the node above, which then changes the node above, which eventually changes the root of the tree and therefore the block's hash, thus causing the protocol to register it as a completely different module (almost certainly with an invalid proof-of-work).

The Merkle tree agreement is arguably critical to long-term sustainability. A "**full node**" in the Bitcoin network that stores and processes the entirety of each block, occupying about 95 GB of disk space in the Bitcoin network as of April 2021, and growing by more than 10 GB per month. Currently this is possible for some desktops rather than phones, and in the future only businesses and hobbyists will be able to participate. The protocol called Simplified Payment Verification (SPV) allows for the existence of another class of nodes, called " light nodes, " which download block headers, verify the proof-of-work on the block headers, and then download only the "**fork**" associated with it transaction associated. This enables light nodes to determine the status of any Bitcoin transaction and its current balance with strong security, while downloading only a small portion of the entire blockchain.

2.5 Alternative Blockchain Applications

The idea of taking basic blockchain concepts and applying them to other concepts also has a long history. In 1998, Nick Szabo coined the concept of secure property rights with owner authorization, and the document described how "**new advances in replicated database technology**" would allow blockchain- based systems to store registries of who owns land, carefully designed Framework, including concepts such as home maintenance, adverse possession and Georgian land tax. Unfortunately, however, no efficient replicated database systems were available at the time, so this protocol was never implemented in practice. However, after 2009, once Bitcoin's decentralized consensus was enacted, many alternative applications quickly emerged. Namecoin - Created in 2010, Namecoin is best described as a decentralized name registration database. In decentralized protocols like Tor, Bitcoin, and BitMessage, there needs to be some way to identify accounts so others can interact with them, but in all existing solutions, the only identifiers available are pseudorandom hashes, For example 1LW79wp5ZBqaHW1jL5TCiBCrhQYtHagUWy. Ideally, people would like to be able to have an account with the name "george". The problem, however, is that if one person can create an account called "george", others can use the same process to register themselves with "george" and impersonate them. The only solution is a paradigm from the first to the first file, where the first registrar succeeds and the second registrar fails - a problem well suited to the Bitcoin consensus protocol. Namecoin is the oldest and most successful implementation of a name registration system using this idea.

Colored Coins - Colored coins are intended to be used as protocols that allow people to create their own digital currencies - or, in important trivial cases, digital tokens in units of numbers on the Bitcoin blockchain. In the Colored Coins protocol, new coins are "issued" by publicly assigning a color to a specific Bitcoin UTXO, and the protocol recursively defines the color of other UTXOs to be the same color as the input used for the transaction that created them. (Some special rules apply to mixed color input).

This allows users to maintain wallets that only contain UTXOs of a specific color, and send them around like regular Bitcoin, backtracking in the blockchain to determine the color of any UTXO they receive.

Metacoins - The idea behind Metacoin is to have a protocol that lives on top of Bitcoin that uses Bitcoin transactions to store Metacoin transactions, but with a different state transition function, APPLY'. Since the metacoin protocol cannot prevent invalid metacoin transactions from appearing in the Bitcoin blockchain, a rule has been added that if APPLY'(S,TX)returns an error, the protocol defaults to APPLY'(S,TX)=S . This provides a simple mechanism for creating arbitrary cryptocurrency protocols, possibly with advanced features that cannot be implemented within Bitcoin itself, but with very low development costs, as the complexities of mining and networking are already handled by the Bitcoin protocol. Metacoins have been used to implement certain classes of financial contracts, name registration, and decentralized exchanges.

So, in general, there are two ways to build a consensus protocol: build an independent network, and build a protocol on top of Bitcoin. The former approach, while quite successful in applications like Namecoin, is difficult to implement; each individual implementation requires bootstrapping a separate blockchain, as well as building and testing all necessary state transitions and networking code. In addition, we predict that the set of applications for decentralized consensus technology will follow a power-law distribution, with the vast majority of applications being too small to warrant their own blockchain, and we note that there is a large class of decentralized applications , especially decentralized autonomous system organizations, need to interact with each other.

On the other hand, the Bitcoin-based approach has the disadvantage that it does not inherit Bitcoin's simplified payment verification features. SPV is suitable for Bitcoin because it can use blockchain depth as a proxy for validity. At some point, once the ancestors of the trade had gone far enough, it was safe to say they were a legitimate part of the state. On the other hand, a blockchain-based meta- protocol cannot enforce that the blockchain does not include transactions that are invalid within the scope of its own protocol. Therefore, a fully secure SPV meta-protocol implementation would need to scan backwards all the way to the beginning of the Bitcoin blockchain to determine whether certain transactions are valid. Currently, all "light" implementations of Bitcoin-based meta-protocols rely on trusted servers to serve data, which is arguably a suboptimal outcome, especially when one of the main purposes of cryptocurrencies is to remove the need for trust Time.

2.6 Scripting

Even without any extensions, the Bitcoin protocol actually does promote a weaker version of the concept of "smart contracts". UTXOs in Bitcoin can be owned not only by public keys, but also by more complex scripts expressed in simple stack-based programming languages. In this paradigm, a transaction costs UTXO and must provide data that satisfies the script. In fact, even the basic public key ownership mechanism is implemented through scripts: the script takes an elliptic curve signature as input, verifies it against the transaction and the address that owns the UTXO, and returns 1 if the verification is successful, 0 otherwise. Other, more complex scripts exist for various other use cases. For example, a script could be constructed that requires two of a given three private keys for verification ("multisig"), a setup suitable for corporate accounts, secure savings accounts, and certain merchant escrow situations. Scripts can also be used to pay bounties for solving computational problems, or even build a script that says "If you can provide SPV proof that a Dogecoin transaction of this denomination was sent to you, this Bitcoin UTXO is yours", Essentially allows for decentralized cross-cryptocurrency exchanges.

However, the scripting language implemented in Bitcoin has several important limitations:

Lack of Turing completeness - that is, although the Bitcoin scripting language supports a large portion of computation, it supports almost all of it. The main missing category is loops. This is done to avoid infinite loops during transaction validation. In theory, this is an insurmountable hurdle for script programmers, as any loop can be emulated by simply repeating the underlying code multiple times with an if statement, but this does make the script very spatially inefficient. For example, implementing an alternative elliptic curve signature algorithm might require 256 repeated multiplication rounds, all of which are individually contained in the code.

Value Blind - UTXO scripts cannot provide fine-grained control over the amount that can be withdrawn. For example, a strong use case for an oracle contract is a hedging contract, where A and B stake $1000 worth of BTC, and the script sends $1000 worth of BTC to A after 30 days, and the rest to B. Oracles determine the dollar value of 1 BTC, but even then this is a huge improvement in terms of trust and infrastructure requirements compared to the fully centralized solutions available today. However, because UTXOs are all-or-nothing, the only way to achieve this is a very inefficient hack, which is to have many UTXOs of different denominations (eg, a 2k UTXO for every k of the 30k) and run it by O chooses which UTXO to send to A, sending to B lacks state - UTXOs may or may not be spent; multi-phase contracts or scripts have no chance to retain any other internal state beyond this. This makes it difficult to formulate multi-phase options contracts, decentralized exchange quotes, or two-phase cryptographic commitment agreements (necessary for secure computation of bounties). This also means that UTXOs can only be used to build simple one-time contracts, not more complex "**stateful**" contracts such as decentralized organizations, and make meta-protocols difficult to implement. The combination of binary state and value blindness also means that another important application, withdrawal limits, is not possible.

Blockchain Blindness - UTXOs are blind to blockchain data such as nonces, timestamps, and previous block hashes. This severely limits the application of several other classes by taking away the potentiallyvaluablerandomnessofscriptinglanguages.

So we see three ways to build advanced applications on top of cryptocurrencies: building new blockchains, using scripts on top of Bitcoin, and building meta- protocols on top of Bitcoin. Building new blockchains gives unlimited freedom to build feature sets at the cost of development time, bootstrapping effort, and security. Using scripts is easy to implement and standardize, but has very limited functionality, and the meta-protocol, while simple, suffers from scalability drawbacks. With Cosmic Interstellar Public Chain, we intend to build an alternative framework that offers greater benefits in terms of simplified development and more powerful light client features, while allowing applications to share the economic environment and blockchain security.

3. COSMIC INTERSTELLAR PUBLIC CHAIN

The purpose of the Cosmic Interstellar Public Chain is to create an alternative protocol for building decentralized applications, offering a different set of trade- offs, which we think will be very useful for a large class of decentralized applications, especially during rapid development times, Small and small security cases. Rarely used applications and the ability for different applications to interact very efficiently are very important. The Cosmic Interstellar Public Chain does this by building what is essentially the ultimate abstract base layer: a blockchain with a built-in Turing-complete programming language that allows anyone to write smart contracts and decentralized applications in which they can create Own ownership, transaction format and state transition capabilities. A basic version of Namecoin can be written in two lines of code, while other protocols (such as currency and reputation systems) can be built in less than 20. It is also possible to build smart contracts on top of the platform,

i. e. cryptographic "**boxes**" that contain value and can only be unlocked when certain conditions are met, which are far more powerful than those offered by Bitcoin scripts, due to the added functionality of Turing completeness , Value Consciousness, Blockchain Consciousness and Status.

PHILOSOPHY

3.1 The Design Behind The Cosmic Interstellar Public Chain Aims To Follow The Following Principles:

1. Simplicity: The Cosmic Interstellar Public Chain protocol should be as simple as possible, even at the cost of some data storage or time inefficiency. Ideally, the average programmer should be able to follow and implement the entire specification. To fully realize the unprecedented democratization potential brought by cryptocurrencies and further advance the vision of the Cosmic Interstellar Public Chain as a protocol open to all. Unless the optimization provides a very substantial benefit, it should not include any optimizations that increase complexity.

2. Universality: A fundamental part of the design philosophy of the Cosmic Interstellar Public Chain is that the Cosmic Interstellar Public Chain does not have a "function". Instead, the Cosmic Interstellar Public Chain provides an internal Turing-complete scripting language that programmers can use to build any mathematically definable smart contract or transaction type. Want to invent your own financial derivatives? With Cosmic Interstellar Public Chain, you can. Want to make your own currency? Set it to the Cosmic Interstellar Public Chain contract. Want to build a complete Daemon or Skynet? You may need thousands of interlocking contracts, and be sure to provide them generously, however, there is nothing stopping you from using the Cosmic Interstellar Public Chain anytime, anywhere.

3. Modularity: The various parts of the Cosmic Interstellar Public Chain protocol should be designed to be as modular and separable as possible. During development, our goal was to create a program where if a small protocol modification were to be made in one place, the application stack would continue to function without any further modification. Innovations such as Ethash (see Yellow Book appendix), modified Patricia trees (Yellow Paper, Wiki), and RLP (YP, Wiki) should be implemented as separate fully functional libraries. In this way, even if the Cosmic Interstellar Public Chain does not require certain functions, even if the Cosmic Interstellar Public Chain does not require certain functions, these functions can still be used in other protocols. Cosmic Interstellar Public Chain development should be done to the greatest extent possible to benefit the entire cryptocurrency ecosystem, not just itself.

4. Agility: The details of the Cosmic Interstellar Public Chain protocol are not set in stone. Although we would make sensible modifications to the high-level structure, such as using a sharding roadmap, abstracting execution, and only incorporating data availability into consensus. Computational testing later in the development process may lead us to discover that certain modifications, such as the protocol architecture or the Cosmic Interstellar Public Chain Virtual Machine(EVM), will greatly improve scalability or security.

5. Non-discrimination and non-censorship: The agreement should not attempt to actively limit or prevent certain categories of use. All moderation mechanisms in the protocol should be designed to mediate harm directly, rather than attempt to oppose specific undesirable applications. Programmers can even run infinite loop scripts on top of the Cosmic Interstellar Public Chain, as long as they are willing to keep paying transaction fees for each computational step.

3.2 Cosmic Interstellar Public Chain Account

In the Cosmic Interstellar Public Chain, state consists of objects called "accounts", each account has a 20-byte address, and state transitions are direct transfers of value and information between accounts. The Cosmic Interstellar Public Chain account contains four fields:
A nonce, a counter used to ensure that each transaction can only be processed once the account's current coin balance The account's contract code (if any) The account's storage space (empty by default) "**CIPC**" is the main internal of the Cosmic Interstellar Public Chain Crypto gas, used to pay transaction fees. Generally, there are two types of accounts: externally owned accounts controlled by private keys and contract accounts controlled by their contract code. Externally owned accounts have no code, and can send messages from externally owned accounts by creating and signing transactions; in contract accounts, each time the contract account receives a message, its code is activated, allowing it to read and write to internal storage and send other messages or create contracts in turn.

Note that "contracts" in the Cosmic Interstellar Public Chain should not be viewed as things that should be "**performed**" or "**complied with**"; instead, they are more like "autonomous agents" in the execution environment of the Cosmic Interstellar Public Chain, when executed by a message or A specific piece of code is always executed when a transaction is "**stamped**" and has direct control over its own coin balance and its own key/value store to keep track of persistent variables.

3.3 Transactions

The term "transaction" as used in the Cosmic Interstellar Public Chain refers to a signed packet of data that stores a message to be sent from an externally owned account. The transaction includes:
The recipient of the message; the signature identifying the sender
The optional data field STARTGAS value for the amount of coins transferred from the sender to the receiver, indicating the maximum number of calculation steps that the transaction execution is allowed to perform GASPRICE value, indicating the fee to be paid by the sender for each calculation step The first three are standard fields expected in any cryptocurrency. The data field does not work by default, but the virtual machine has an opcode that the contract can use to access the data. As an example use case, if a contract acts as a domain registration service on the blockchain, it may want to interpret the data passed to it as containing two "fields", the first is the domain to register and the second is Domain to register. field is the IP address to register with.
The contract will read these values from the message data and store them appropriately.

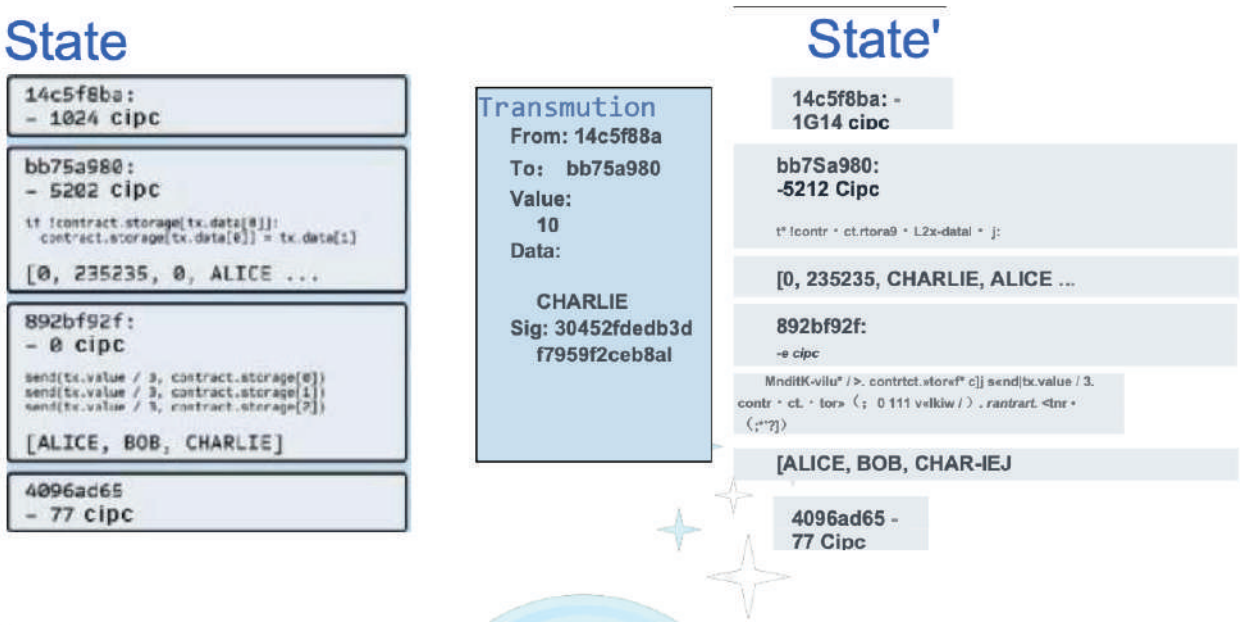
The STARTGAS and GASPRICE fields are critical to Cosmic Interstellar Public Chain's anti-denial of service model. To prevent accidental or hostile infinite loops or other computational waste in the code, each transaction is required to place a limit on how many code execution steps it can use. The basic unit of computation is "gas". Typically, a computational step costs 1 gas, but some operations consume more gas because they are more computationally expensive or increase the amount of data that must be stored as part of the state. There is also a 5 gas fee for each byte of transaction data. The purpose of the charging system is to require attackers to pay proportionally for each resource they consume (including compute, bandwidth, and storage); therefore, any transaction that causes the network to consume a large amount of any of these resources must have a gas fee of roughly proportional to the increment.

3.4 Messages

Contracts have the ability to send "**messages**" to other contracts. Messages are virtual objects that are never serialized and only exist in the Cosmic Interstellar Public Chain execution environment. A message contains:
The sender of the message (implicitly) the recipient of the message The amount of ether transferred with the message Optional data field STARTGAS value.
Essentially, a message is similar to a transaction, except that the message is produced by a contract rather than an external participant. A message is generated when the contract currently executing the code executes the CALL opcode, which generates and executes a message. Like a transaction, a message causes the recipient account to run its code. Therefore, contracts can have relationships with other contracts in exactly the same way as external actors.

Note that the gas allowance allocated by a transaction or contract applies to the total gas consumed by that transaction and all sub-executions. For example, if external actor A sends a transaction to B with 1000 gas, and B consumes 600 gas before sending a message to C, and C's internal execution consumes 300 gas before returning, then B can consume another 100 gas to consume Out of gas.

3.5 Cosmic Interstellar Public Chain State Transition Function



The Cosmic Interstellar Public Chain state transition function APPLY(S, TX)->S' can be defined as follows:

1. Check that the transaction is well-formed (i.e. has the correct number of values), that the signature is valid and that the nonce matchesthe-nonceinthesender'saccount.Ifnot,returnanerror.
2. Calculate the transaction fee as STARTGAS * GASPRICE, then determine the sending address from the signature. The fee is subtracted from the sender's account balance and the sender's currentisadded.Ifthebalanceisinsufficient,returnanerror.
3. Initialize GAS = STARTGAS, then release a certain amount of gas per byte to pay for the bytes in the transaction.
4. Transfer the transaction value from the sender's account to the receiver's account. Create a payout account if it doesn't already exist. If the receiving account is a contract, run the contract's code to complete or until execution ends.
5. If the value transfer fails because the sender doesn't have enough money, or the code execution runs out of money, revert all state changes except paying the fee, then add the fee to the miner's account.
6. Otherwise, refund all remaining gas fees to the sender and send the paid gas usage fees to the miners.

For example, suppose the code for the contract is:

```
if   !self.storage[calldataload(0)]:
 self.storage[calldataload(0)] = calldataload(32)
```

Note that the contract code is actually written in low-level EVM code; for clarity, this example is written in Serpent, a high-level language, and can be compiled to EVM code. Suppose the storage space of the contract starts empty and a transaction is sent with 10 coins, 2000 gas, 0.001 coin gasprice and 64 bytes of data, bytes 0-31 represent the number 2 and bytes 32-63 represent the string CHARLIE .fn. 6 In this case, the state transition function proceeds as follows:

1. Check that the transaction is valid and formatted correctly.
2. Check if the sender of the transaction has at least 2000 * 0.001 = 2 coins Ifso, subtract 2 coins from the sender's account.
3. Initialization gas = 2000; assuming a transaction length of 170 bytes and a byte fee of 5, subtract 850, leaving 1150 gas.
4. Subtract another 10 coins from the sender's account and add it to the contract's account.
5. Run the code. In this case, it's simple: it checks if the contract store at index 2 is used, notices that it's not used, so it sets the store at index 2 to a CHARLIE value. Assuming that 187 gas is required, the remaining gas volume is 1150-187 = 963.
6. Add 963 * 0.001 = 0.963 coins back to the sender's account and return the result status.

If there is no contract on the receiving end of the transaction, the total transaction fee will simply be equal to the provided GASPRICE multiplied by the length of the transaction in bytes, and the data sent with the transaction will be irrelevant.

Note that messages work equivalently to transactions in terms of restore: if a message execution is exhausted, the execution of that message and all other executions triggered by that execution will be restored, but the parent execution does not need to be restored. This means that it is "safe" for a contract to call another contract, as if A called B with G gas, then A's execution is guaranteed to lose at most G gas. Finally, note that there is an opcode CREATE that creates a contract. Its execution mechanism is generally similar to CALL, except that the output of the execution determines the code of the newly created contract.

3.6 Code Execution

The code in the Cosmic Interstellar Public Chain contract is written in a stack- based underlying bytecode language called "Cosmic Interstellar Public Chain Virtual Machine Code" or "EVM Code". The code consists of a sequence of bytes, where each byte represents an operation. Typically, code execution is an infinite loop consisting of repeating the current program counter (starting at zero) and then incrementing the program counter by 1 until the end of code is reached or an error or STOP occurs. or a RETURN instruction detected. These operations have access to three types of space to store data: the stack, a last-in-first-out container into which values can be pushed and popped into memory, and the long-term storage of infinitely expanding byte array contracts, known as key/value storage. Unlike stack and memory, which are reset after computation is over, storage can be retained long-term.

4. KEY TECHNOLOGIES

4.1 SPos Consensus Protocol

There are also quite a few blockchain networks, such as the most famous Bitcoin and Ethereum (ETH 2.0 is moving towards PoS 2.0), which all use a proof-of- work system based on asymmetric algorithms. These proofs are extremely difficult to generate, but third-party verification is straightforward. The security of these networks is achieved through network-wide consensus as subsequent blocks are added to the blockchain, the number of chains becomes very difficult to create. However, these computationally expensive proofs have no other use for the blockchain network. We define usefulness as work that is valuable to the blockchain network in addition to securing the distributed ledger. While other networks try to convert mining power into something useful, such as Ethereum executing what are called smart contracts, most of the work is useless or reusable. The mining process is also very wasteful, as the determining factor for the work is often computing power, which consumes a lot of electricity and requires a lot of hardware to perform. The consensus mechanism of SPos is that for each time period, the system selects a group of entities called validators and plays a key and important role in the next election cycle, such as block production and final validation. The quality and quantity of work is high, requiring costly operations to run to ensure high communication responsiveness and build a long-term, reliable reputation. Therefore, validators also need staking as a guarantee of good behavior. When deviating from the normal track, staking will be punished according to its severity. Instead of being rewarded when they diligently follow the rules. Any node that completes the node task or can meet the requirements can publicly become a candidate validator. Of course, due to various reasons, the number of validators can only be limited. In the early stage, there will be 360 validators. In the later stage, it may reach hundreds or even thousands of development conditions according to the whole network.

We also encourage CIPC holders to join the web ecosystem as much as possible and vote for validators as electors. If the candidate validator supported by the elector officially enters the validator node pool, the elector will receive the corresponding reward according to the proportion. The amount of staking and the ratio of commission are different from validators, and there is no number of electors. As long as the elector selects and supports only candidate validators with good security practices, the risk is low and there is a constant revenue stream. This elector-validator design has very strong security guarantees. It allows the system to select validators with a large total pledge, and eliminate candidates with a relatively small total pledge.

In fact, at any given moment, we want a significant portion of our CIPC to be staked. This makes it difficult for adversary groups to become validating nodes (as they need to build a strong reputation to get the support they need) and it is expensive to attack the system (as any attack would result in massive CIPC slashes). Our consensus mechanism scheme is more efficient than Proof of Stake The mechanism(Pow) is much more efficient and also much faster than the proof-of-stake mechanism (PoS): it allows the continuous participation of almost all participants holding CIPC, thus maintaining a high level of security, while limiting the number of validating nodes, so all Basic network operations are valid.

To prevent collusion between validators and collectors, validators are randomly assigned at regular intervals. The application of blockchain technology has always been the most concerned topic, and as a public chain that carries applications, its performance determines the upper limit of the application, including the upgrade of Ethereum, which is committed to improving system performance.

The birth of the public chain and smart contracts provide a carrier for blockchain applications, but the application scenarios are also limited by the public chain. One of the obstacles to the mass adoption of blockchain is scalability, i.e. the performance of the blockchain. Scalability limits the transaction throughput of the blockchain, leading to congestion and higher transaction fees, a problem that Ethereum often faces. Capacity expansion is the fundamental solution to network congestion, and multi-layer structure is one of the main ways to achieve capacity expansion. Through the multi-layer structure, data can be parallelized in different network partitions to improve processing efficiency. For example, the Layer 2 expansion scheme of Ethereum puts the calculation process off the chain, and sends the final result back to the chain to enhance information processing.

4.2 Layer2 Scheme

The Layer 2 extension solution on Optimistic Rollups (ORU), ORU has many desirable properties. Among all blockchain scalability technologies, it has incomparable characteristics with the following characteristics:

Trustless

Unlike traditional sidechain technologies, ORU is trust-free (or more technically, trust-minimized). You can withdraw funds from a Rollup at any time without trusting that the vast majority of block producers on-theORUarehonest.

License

Unlike Plasma, ORU is license-free. Anyone can become a block producer on ORU because all block data on rollup is published on and available from Ethereum. How to choose the next leader is a specific implementation issue, not a fundamental limitation.

Unmanaged

Since ORU is both trusted and license-free, you can withdraw money anytime and no one can stop you. Therefore, ORU is hosted for free.

Expressive

UnlikeZKrollup,ORU(bothintheoryandpractice)ishighlyexpressive. Whether it's a Bitcoin-like UTXO payment or a full-fledged compliant EVM execution, ORU can handle it.

Open To Participation

Unlike payment channels, ORU supports smart contracts and is open to everyone, just like Uniswap.

High Capital Efficiency

Unlike payment channels, ORU does not require users to lock up funds in advance.

Anti-Chain Congestion

Unlike payment channels and Plasma, ORU can defend against on-chain congestion because ORU fraud is proven at the block level, not a shutdown mechanism like payment channels, or an exit mechanism like Plasma.

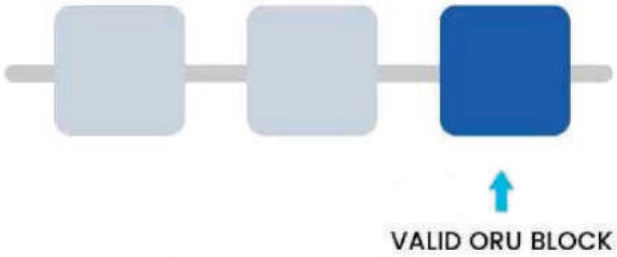
No New Cryptographyrequired

Unlike ZKrollup, ORU does not require anynew cryptography.

Fast (Non-instant) Determinism

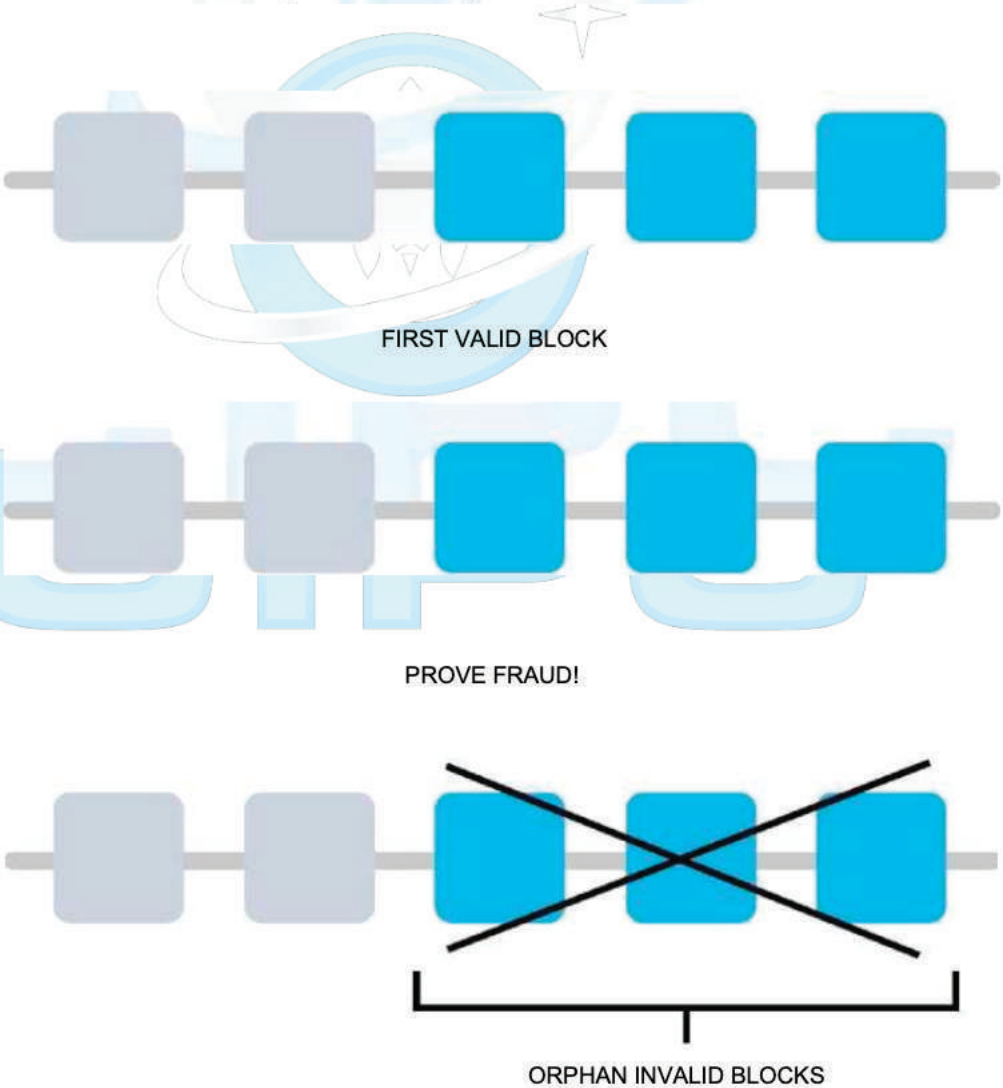
Unlike ZK rollup, ORU does not need to generate proofs, ORU blocks can be published to Ethereum immediately. Since valid ORU blocks cannot be rolled back, like Ethereum, once finalized, they can be published to Ethereum for use.

CASE 1 : **VALID BLOCK**



CASE 2 : **INVALID BLOCK**

1..One or more invalid rollup blocks are posted to Ethereum



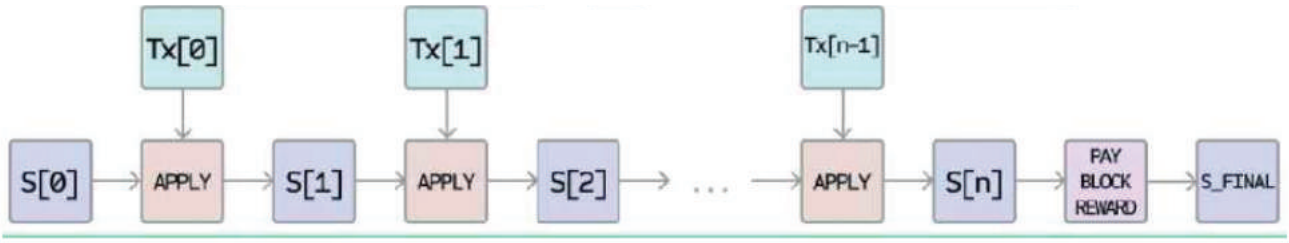
1. The aggregator collects the transactions on the rollup, packs them into a rollup block, and sends the rollup block along with the deposit (we'll explain why a deposit is required shortly) to a smart contract on Ethereum (or another Ethereum-like The blockchain runs a large number of smart contracts). This aggregated block is not translated or executed - the smart contract just records the block hash and keeps track of the hashes of all aggregated blocks. Rollup blocks themselves are not stored in smart contracts, but everyone can find them in Ethereum's historical transactions.

2. The Rollup block contains a state root, which is the root of the state tree of the rollup block. If the state root is invalid, anyone can use a fraud proof during the challenge to prove that it is invalid. This could be because the transaction in the summary block is invalid, or because the state root is invalid. If a rollup block turns out to be invalid, the contract rolls back the rollup chain, and all rollup blocks following the invalid block become orphaned. Once the fraud is proven successful, part of the deposit will be paid to the validator and the rest will be destroyed.

3. If the fraud proof is not submitted by the end of the challenge period, the contract will finalize the rollup block and allow the aggregator to withdraw the deposit. When a user withdraws money from the Rollup chain to the main chain, a withdrawal request needs to be initiated on the Rollup chain. Funds can only be withdrawn after the aggregated block is finalized in the contract.

4.3 BFT Blockchain Algorithm

BFT (Byzantine Fault Tolerance) solves the problem of low efficiency of the original Byzantine Fault Tolerance algorithm, and reduces the algorithm complexity from the exponential level of the number of nodes to the square level of the number of nodes, making the Byzantine Fault Tolerance algorithm feasible in practical system applications. When each witness generates a block, it is still broadcast on the whole network. After other witnesses receive the new block, they will immediately verify the block, and return the block whose signature is completed to the block witness immediately, without waiting for other witnesses Confirm when you generate the block yourself. From the perspective of the current block-producing witness, he produced a block and broadcast it to the entire network, and then successively received confirmations of this block from other witnesses. At the moment when 2/3 of the witnesses confirmed, so the transaction It onlytakes 3 seconds.



The Cosmic Interstellar Public Chain blockchain is similar in many ways to the Bitcoin blockchain, although it does differ. The main difference between the Cosmic Interstellar Public Chain and Bitcoin in terms of blockchain architecture is that, unlike Bitcoin, which only contains a copy of the transaction list, Cosmic Interstellar Public Chain blocks contain a copy of the transaction list and the latest state. In addition to this, two other values, the block number and difficulty, are also stored in the block. The basic block verification algorithm in the Cosmic Interstellar Public Chain is as follows:

1. Check that the referenced previous block exists and is valid.
2. Check that the timestamp of the block is greater than the timestamp of the previous block referenced and not more than 15 minutes into the future.
3. Check that block number, difficulty, transaction root, uncle root and gas limit (various Cosmic Interstellar Public Chain specific low-level concepts) are valid.
4. Check that the proof of work on the block is valid.
5. LetS[0]bethestateattheendofthepreviousblock.
6. Let TX be the transaction list of the block, containing n transactions. For all i in 0...n-1, set S[i+1] = APPLY(S[i], TX[i]). An error is returned if any application returns an error, or if the total gas consumed in the block until that point exceeds GASLIMIT.
7. Let S_FINAL be S[n], but add the block reward paid to miners.
8. Check that the Merkle tree root of state S_FINAL is equal to the final state root provided in the block header. If so, the block is valid; otherwise, the block is invalid. Otherwise invalid.

A common problem is the "**execution location**" of the contract code on the physical hardware. This has a simple answer: the process of executing the contract code is part of the definition of the state transition function, which is part of the block validation algorithm, so if a transaction is added to block B, the code execution resulting from that transaction will be executed. Executed by all present and future nodes that download and validate block B.

5. APPLICATION AREAS

Generally, there are three types of applications on top of the Cosmic Interstellar Public Chain. The first category is financial apps, which provide users with more powerful ways to use their funds to manage and enter into contracts. These include sub-currency, financial derivatives, hedging contracts, savings wallets, wills, and eventually even full-scale employment contracts in certain categories. The second category is semi-financial applications, involving money, but there are also substantial non-monetary aspects of the work being done. A perfect example is self-enforcing bounties for solving computational problems. Finally, some applications, such as online voting and decentralized governance, are simply not financially capable.

5.1 Token System

Token systems on blockchains have many applications, from sub-currency representing assets like USDor gold to company stocks, individual tokens representing smart property, secure unforgeable coupons, or even having no connection to traditional values at all The token system can have a role. The token system is very easy to implement in the Cosmic Interstellar Public Chain. The key point to understand is that a currency or token system m is basically a database with one operation: subtract X units from A and give B to X units, provided that (1) A has at least X units before the transaction unit, and (2) the transaction has been approved by A. All that is required to implement a token system is to implement this logic into a contract.

The basic code to implement the token system in Serpent is as follows:

```
def send(to, value):
    if self.storage[msg.sender] >= value:
self.storage[msg.sender] = self.storage[msg.sender] - value
self.storage[to] = self.storage[to] + value
```

This is essentially a literal implementation of the "banking system" state transition function described further above in this article. First, a few extra lines of code would need to be added to provide the initial steps for distributing currency units, as well as some other corner cases, and ideally, a function would be added to allow other contracts to query the address's balance. But that's all. In theory, a token system based on the Cosmic Interstellar Public Chain could act as a subcurrency, possibly with another important feature that an on-chain Bitcoin-based meta-currency lacks: the ability to pay transaction fees directly in that currency. The way this is implemented is that the contract will maintain a coin balance with which it can refund the coin used to pay the sender, and it will replenish that balance by taking the internal currency unit collected and reselling it. Ongoing auctions. Therefore, users will need to "activate" their accounts with coins, but once they have coins they can be reused because the contract returns them each time.

5.2 Financial Derivatives And Stable Value Currencies

Financial derivatives are the most common application of "smart contracts" and one of the easiest to implement in code. The main challenge in implementing financial contracts is that most financial contracts need to reference external price quotes; for example, a highly desirable application would be a smart contract that hedges the Cosmic Interstellar Public Chain (or another cryptocurrency) against the US dollar Volatility, but doing so requires the contract to know the value of CIPC/USD. The easiest way to do this is to design through a "data feed" contract maintained by a specific party (such as Nasdaq), so that party can update the contract as needed, and provide an interface that allows other contracts to send the contract. A message is sent to this contract, and a response is returned providing the price.

Taking this key factor into consideration, the hedging contract would look like this:

1. Wait for Party A to input 1000 coins.

2. Wait for Party B to input 1000 coins.

3. Record the USD value of 1000 coins calculated by querying the data feed contract in storage, say this is $x.

4. After 30 days, either A or B is allowed to "reactivate" the contract to send $x worth of coins (calculated by querying the data feed contract again for the new price) to A and the rest to B.

Such contracts would have enormous potential in crypto commerce. One of the main problems with cryptocurrencies is the fact that they are volatile. While many users and merchants may want the security and convenience of dealing with crypto assets, they may not want to face the prospect of losing 23% of the value of their funds in one day. By far the most common solution is issuer-backed assets. The idea is that issuers create a sub-currency that they have the power to issue and withdraw units of, and provide a unit of currency to anyone who provides (offline) a unit of a specified underlying asset (e.g. gold) to them. The issuer then promises to provide one unit of the underlying asset to everyone who backs one unit of the cryptoasset. This mechanism allows all non-crypto assets to be "promoted" into crypto assets, provided that the issuer can be trusted.

In practice, however, issuers are not always trustworthy, and in some cases banking infrastructure is too weak or too hostile to provide such services. Financial derivatives offer another option. Here, a decentralized market of speculators, rather than funded by a single issuer to back an asset, plays the role by betting that the price of a crypto-reference asset, such as CIPC, will rise. Unlike issuers, speculators have no reason to choose to default because the hedging contract places their funds in the hands of custodians. Note that this approach is not fully decentralized, as a trusted source is still needed to provide price quotes, although arguably this is still a huge step forward even in terms of reducing infrastructure requirements (unlike being an issuer, issuing Price feeds do not require a license and may be classified as free speech) and reduce the potential for fraud.

5.3 Identity and Reputation Systems

The earliest alternative cryptocurrency, Namecoin, attempted to use B-itcoin-like blockchains to provide a name registration system where users could register their names in a public database along with other data. The main use case cited is for DNS systems, mapping domain names (such as "**bitcoin.org**" (or "bitcoin.bit" in the case of Namecoin)) to IP addresses. Other use cases include email authentication and possibly more advanced reputation systems. Here is the basic contract that provides a Namecoin-like name registration system on the Cosmic Interstellar Public Chain:

```
def register(name, value):
    if !self.storage[name]:
      self.storage[name] = value
```

The contract is simple,it's just a database inside the Cosmic Interstellar Public Chain network to which it can be added, but not modified or removed from it. Anyone can register a name with some value, and that registration will remain forever. More complex name registration contracts will also have a "**feature clause**" that allows other contracts to query it, as well as a mechanism for the name "owner" (ie the first registrant) to change data or transfer ownership. You can even add reputation and trust web features on top of it.

5.4 Decentralized File Storage

The goal of IPFS is to surpass or even replace HTTP, connect all devices in the network through a file system, and build a faster, more secure, and open sharing Internet.

The CIPC team believes that the low-cost and high-speed IPFS storage solution must be the necessary infrastructure for the future decentralized application network. The IPFS network combines decentralization and cost. In digital life, privacy is becoming more and more important, and decentralized storage will surely become a necessity for people in the future. iCloud was once exploited by hackers to leak nude photos of more than 50 Hollywood actresses. This is not an isolated case, as long as the storage of the Internet remains centralized, such things will continue to happen.

CIPC has built-in LibP2P protocol module. LibP2P is used as the network layer for IPFS. A node using LibP2P to communicate with another node can use a variety of different transports, including connection delays, and communicate over different protocols, negotiating as needed.

The LibP2P module is mainly responsible for the data transfer function in IPFS, that is, routing, network, exchange, etc. DCE takes protecting the privacy and security of user data as its mission, and will provide sustainable, efficient and low-cost storage solutions to DCE users.

An important feature of the protocol is that while it appears that many random nodes can be trusted rather than decide to forget a file, the file can be split into many parts through secret sharing, reducing this risk to near zero. Watch the contract to see that each piece is still in a node's property. If the contract is still being paid, a cryptographic proof can be provided that someone is still storing the file there.

5.5 Decentralized Autonomous Organizations

he general concept of a "**decentralized autonomous organization**" refers to a virtual entity with a certain set of members or shareholders, who may own 67% of the majority of the members, and can use the entity's funds and modify its code. Members will collectively decide how the organization should allocate its funds, which may range from bounties, salaries, to even fancier mechanisms (such as an internal currency to reward work) and more.This essentially replicates the legal pitfalls of traditional corporations or nonprofits, but only uses cryptographic blockchain technology to enforce them. So far, discussions about DAOs have revolved around the " **capitalist** " model of a " **decentralized autonomous company** " (DAO), with shareholders receiving dividends and tradable shares. Another option, possibly called a "**decentralized autonomous community,**" would give all members an equal share in decision-making and would require 67% of existing members to agree to add or remove members. As such, the requirement that a person can only have one membership would need to be enforced collectively by the group. A general overview of how to code a DAO is as follows. The simplest design is just a self-modifying piece of code that changes if two-thirds of the members agree to the change. Although in theory the code is immutable, one can easily solve this problem and in fact improve the Operability. In this simple implementation of a DAO contract, there would be three transaction types, differentiated by the data provided in the transaction:

[0, i, K, V] register proposal with index i to change address on storage index K to value V [1, i] vote on proposal i [2, i] If enough votes have been made, then finalize proposal i

The contract will then provide terms for each term. It will keep a record of all Open Storage changes and a list of who voted for those changes. It will also have a list of all members. When any storage change receives a two-thirds vote of the members, the transaction is completed to execute the change. A more sophisticated framework would also have built-in voting capabilities for things like sending transactions, adding and removing members, and could even provide "**liquid democracy**"-style voting delegation (i.e. anyone can assign someone to vote for it) and distribution). is transitive, so if A assigns B and B assigns C, then C decides A's vote). This design will allow the DAO to grow organically as a decentralized community where people can eventually delegate the task of filtering out who is a member to experts, although unlike the "**current system**" experts can easily over time Appear and disappear. As individual community members change their alignment.

Another model is a decentralized company where any account can have zero or more shares and requires two-thirds of the shares to make a decision. A complete framework would involve asset management functions, the ability to provide offers to buy and sell stocks, and the ability to accept offers (preferably with an in-contract order matching mechanism). Delegations will also exist in the form of liquid democracy, thus encapsulating the concept of a "**board of directors**".

6. FURTHER APPLICATIONS

6.1 Savings Wallet

Suppose Alice wants to keep her funds safe, but fears that she will lose or someone will steal her private keys. She contracts the coins with Bank Bob as follows:
Alice alone can only withdraw up to 1% of her funds per day. Bob alone can withdraw up to 1% of his funds per day, but Alice has the ability to trade her keys to turn off the feature. Alice and Bob together can withdraw everything. Usually, 1% of income per day is enough for Alice's needs, if Alice wants to withdraw more, she can contact Bob for help. If Alice's keys are stolen by a hacker, she will run to Bob to transfer the funds into the new contract. If she loses her keys, Bob will eventually get the money out. If Bob turns out to be malicious, then she can turn off his ability to quit.

6.2 Crop Insurance

Derivatives contracts can be easily entered into using weather data instead of any price index. If Iowa farmers buy derivatives that are paid in reverse based on Iowa precipitation, farmers will automatically receive money if there is a drought, and if there is enough rain, farmers will be happy because they have a good harvest it is good. In general, this can be extended to natural disaster insurance.

6.3 Decentralized Data Feeds

For differentiated financial contracts, it may actually be possible to decentralize the data feed through a protocol called SchellingCoin. The basic working principle of SchellingCoin is as follows: N participants all put values of given data (such as CIPC/USD prices) into the system, these values are sorted, and everyone between 25% and 75% will Get a token as a reward. Everyone has an incentive to provide answers that others will provide, and the only value that many participants can actually agree on is the obvious default: facts. This creates a decentralized protocol that can theoretically provide any number of values, including the CIPC/USD price, the temperature inBerlin,oreventheresultofaspecifichardcalculation.

6.4 Smart Multi-Signature Escrow

Bitcoin allows for multi-signature transaction contracts, for example, given that three of the five keys can spend funds. The Cosmic Interstellar Public Chain allows for more granularity; for example, four out of five can spend everything, three out of five can spend 10% per day, and two out of five can spend 0.5% per day. Additionally, Cosmic Interstellar Public Chain multi-signature is asynchronous - two parties can register their signatures on the blockchain at different times, and the last signature will automatically send the transaction.

6.5 Cloud Computing

EVM technology can also be used to create verifiable computing environments that allow users to ask others to perform computations, and then optionally require proofs to ensure that computations were done correctly at some randomly chosen checkpoints. This allows for the creation of a cloud computing marketplace where any user can participate using their desktop, laptop or professional services. ver, spot checks and deposits can be used to ensure that the system is trustworthy (i.e. nodes cannot profit from it). Although such a
system may not be suitable for all tasks; for example, tasks that require a high level of interprocess communication are difficult to accomplish on a huge cloud of nodes. However, other tasks are easier to parallelize. Projects like SETI@home, folding@home, and genetic algorithms can all be easily implemented on such a platform.

6.6 Peer-to-Peer Quiz

Many peer-to-peer betting protocols such as Frank Stajano and Richard Clayton's Cyberdice can be implemented on the Cosmic Interstellar Public Chain blockchain. The simplest betting protocol is really just a difference contract on the next block hash, and more advanced protocols can be built from there to create betting services with almost zero fees and no cheating. prediction market. Prediction markets are also easily implemented if Oracle or SchellingCoin are provided, and together with SchellingCoin, prediction markets may prove to be the first mainstream application of futarchy as a governance protocol for decentralized organizations.

7. MISCELLANEOUS AND CONCERNS

7.1 Modified GHOST Implementation

The "greedy most heavily observed subtree" (GHOST) protocol is an innovation first proposed in December 2013 by Yonatan Sompolinsky and Aviv Zohar. The motivation behind GHOST is that blockchains with short confirmation times currently suffer from reduced security due to the high rate of obsolescence - because if miner A mines a block, then miner B happens to be propagating miner A's block to B Another block was mined before, then Miner B's block would end up being wasted and would not contribute to network security. Also, there is a centralization issue: if miner A is a mining pool with 30% hash power and B has 10% hash power, A will risk producing stale blocks 70% of the time (because the other 30 % of the time A generates the last block and thus will get mining data immediately), while B runs the risk of generating stale blocks 90% of the time. So if the block interval is short enough that the stale rate is high, A will be substantially more efficient simply because of its size. Combining these two effects, a blockchain that produces blocks quickly is likely to result in a mining pool with a sufficiently large percentage of the network's hashing power to have actual control over the mining process.

As described by Sompolinsky and Zohar, GHOST addresses the first issue of network security loss by including stale blocks when calculating which chain is the "longest" chain. That is, count not only the parent and other ancestors of the block, but also the stale descendants of the block ancestors ("uncles" in Cosmic Interstellar Public Chain jargon) to calculate which block has the largest total work quantity to prove it. To address the second issue of centralization bias, we go beyond the protocol described by Sompolinsky and Zohar and also provide block rewards for stale blocks: stale blocks get 87.5% of their base reward, while nephews including stale blocks get Get the remaining 12.5%. However, transaction fees are not awarded to uncle blocks.

The Cosmic Interstellar Public Chain implements a simplified version of GHOST, dropping only seven levels. Specifically, the definitions are as follows:

A block must specify a parent and must specify 0 or more uncles contained in an uncle must have the following properties:

It must be a direct descendant of the k-th ancestor of B, where 2 <= k <= 7. It cannot be an ancestor of B An uncle block must be a valid block header, but does not have to be a previously verified or even a valid block

An uncle must be distinct from all uncles contained in previous blocks and all other uncles contained in the same block (non-double inclusion) For each uncle U in block B, the coinbase obtained by B's miners The reward increases by 3.125%, and U's miners receive 93.75% of the standard coinbase reward.

This restricted version using GHOST has up to 7 generations of uncles for two reasons. First, an infinite GHOST would involve too much complexity in calculating which uncles of a given block are valid. Second, the infinite GHOST with compensation used in the Cosmic Interstellar Public Chain removes the incentive for miners to mine on the main chain instead of the public attacker chain.

7.2 Fees

Since every transaction posted to the blockchain adds to the cost of downloading and validating the transaction on the network, some kind of regulatory mechanism (often involving transaction fees) is needed to prevent abuse. The default approach used in Bitcoin is to charge purely voluntary fees, relying on miners to act as gatekeepers and set dynamic minimum fees. This approach has been well received in the Bitcoin community, especially in becau. Because it is "**market-based**," it allows supply and demand between miners and transaction senders to determine the price. The problem with this way of reasoning, however, is that transaction processing is not a marketplace. While it is intuitively appealing to understand the services that miners provide to senders as transaction processing, in practice every transaction a miner includes needs to be processed by every node in the network, so the vast majority of the cost of the transaction is processed The work is decided by a third party, not by the miners who decide whether to include it or not. As a result, the usual tragic problems are highly likely.

However, it turns out that in the presence of this flaw in a market-based mechanism, given certain inaccurate simplifying assumptions, it magically cancels itself out. The parameters are as follows, assuming:

1. A transaction results in k actions that provide a reward kR to any miner that includes that reward, where R is set by the sender and kandRare(roughly)visibletotheminerinadvance.
2. An operation has a processing cost of C for any node (ie, all nodes are equally efficient)
3. There are N mining nodes, each with exactly equal processing power (i.e. 1/N of the total)
4. There are no non-mining full nodes.

If the expected return is greater than the cost, then miners will be willing to trade. Therefore, since the miner has a 1/N chance to process the next block, the expected reward is kR/N, while the miner's processing cost is only kC. Therefore, miners will include transactions where kR/N > kC or R > NC. Note that R is the per-operation fee offered by the sender, and therefore a lower bound on what the sender can earn from the transaction, while NC is the cost of the entire network processing the operation together. Therefore, miners have an incentive to include only those transactions whose total utility exceeds cost.
However, there are actually several important deviations from these assumptions:
1. Miners do pay higher transaction fees than other validators because the extra validation time delays block propagation, increasing the chance of blocks becoming stale.
2. There do exist non-mining full nodes.
3. In practice, the distribution of mining rights may end up being fundamentally unequal.
4. Speculators, political enemies, and madmen do exist, and their utility functions include causing damage to the network, and they can subtly build contracts at a much lower cost than what other validating nodes would pay.

(1)makes miners tend to reduce transactions(2)increases NC; thus, these two effects at least partially cancel each other out. To solve these problems, we simply set a floating cap, no block can do more than BLK_LIMIT_FACTOR times the long-term exponential moving average. Specifically:

blk.oplimit = floor((blk.parent.oplimit * (EMAFACTOR - 1) +
floor(parent.opcount * BLK_LIMIT_FACTOR)) / EMA_FACTOR)

BLK_LIMIT_FACTOR and EMA_FACTOR are constants and will be set to
65536and1.5fornow,butmaychangeafterfurtheranalysis.

There is another factor that does not incentivize large blocks in Bitcoin: large blocks will take longer to propagate and thus have a higher probability of becoming old. In the Cosmic Interstellar Public Chain, the propagation of gas- intensive blocks may also take longer, both because they are physically larger, and because they take longer to process transaction state transitions for verification. This latency inhibitor is an important consideration in Bitcoin, but less so in the Cosmic Interstellar Public Chain due to the GHOST protocol. Therefore, relying on regulated block limits provides a more stable baseline.

7.3 Computation and Turing Completeness

An important note is that the Cosmic Interstellar Public Chain virtual machine is Turing complete. This means that EVM code can encode any computation imaginable, including infinite loops. EVM code allows looping in two ways. First, there is a JUMP instruction that allows the program to jump back to a previous location in the code, and a JUMPI instruction that does a conditional jump, allowing statements such as x < 27: x = x *2. Second, contracts can call other contracts, possibly allowing traversal recursion. This naturally raises the question: can a malicious user essentially shut down miners and entirenodesbyforcingthemintoaninfiniteloop?Thisproblemarises due to a problem in computer science (called the "stopping problem"): in the general case, there is no way to tell whether a given program will stop or not.

As described in the state transitions section, our solution works by requiring the transaction to set the maximum number of computational steps allowed to execute, and if execution takes longer, the computation can be resumed, but still incurs a fee. Messages work the same way. To show the motivation behind our solution, consider the following example: An attacker creates a contract that runs an infinite loop, and then sends a transaction that activates the loop to the miner. The miner will process the transaction, run an infinite loop, and wait for it to run out of gas. Even if execution runs out and stops halfway through, the transaction is still valid, and the miner still charges the attacker a fee for each computational step.

The attacker creates a very long infinite loop in order to force the miner to keep the calculation for so long that more blocks will appear when the calculation is complete, and it is impossible for the miner to include the transaction to charge a fee. However, the attacker will need to submit a value for STARTGAS to limit the number of computational steps the execution can perform, so the miner will know ahead of time that the computation will take too many steps.

The attacker sees the contract with some form of code, e.g. send(A, contract.storage[A]); contract.storage[A] = 0, and sends a transaction with enough volume to perform the first step but not The second step of the transaction (i.e. make a withdrawal without letting the balance decrease). Contract authors don't need to worry about such attacks,

because if executionstops in the middle of changes, they will be reverted.

Financial contracts work by taking a median of 9 proprietary data feeds to minimize risk. The attacker took over one of the data feeds, which was designed to be modified via the variable address call mechanism described in the DAO section, and turned it into running an infinite loop in an attempt to force any attempt to claim funds from a financial contract Run out. However, financial contracts can place limits on messages to prevent this problem.

An alternative to Turing Incompleteness is Turing Incompleteness, where JUMP and JUMPI do not exist and only one copy of each contract can exist in the call stack at any given time. Using this system, the described fee system and uncertainty surrounding the effectiveness of our solution may not be needed, as the cost of enforcing a contract will be limited by the size of the contract. Also, Turing incompleteness isn't even that big of a restriction. Of all the contract examples we've conceived internally, only one so far requires a loop, which can even be eliminated by doing 26 repetitions of a line of code. Given the serious implications and limited benefits of Turing-complete, why not simply use Turing- incomplete languages? In practice, however, Turing incompleteness is far from an ingenious solution. To see why, consider the following contracts:

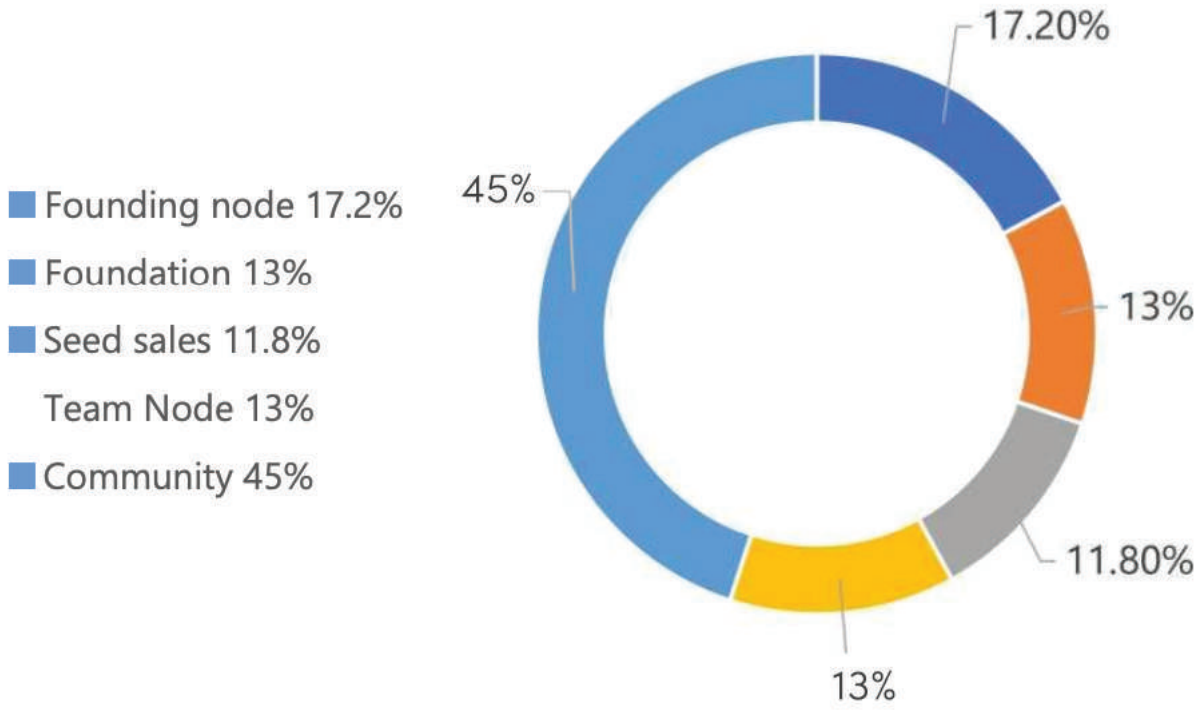
C0: call(C1); call(C1);
C1: call(C2); call(C2);
C2: call(C3); call(C3);
C49: call(C50); call(C50);
C50: (run one step of a program and record the change in storage)

Now, send the transaction to A. So, out of 51 transactions, we have a contract that takes 250 computational steps. A miner can try to spot such logic bombs ahead of time by keeping a value next to each contract specifying the maximum computational step it can take, and calculating that value for contracts that recursively call other contracts, but this would require the miner to forbid the creation of Contracts other contracts (since the creation and execution of all 26 contracts above can easily be combined into one contract). Another problematic point is that the address field of the message is a variable, so often it may not even be known ahead of time which other contracts will be called by a given contract. So, in conclusion, we come to a surprising conclusion: Turing completeness is surprisingly manageable, and lack of Turing completeness is equally astonishingly difficult to manage unless the exact same controls exist, but there In this case why not just make the protocol Turing complete?

8. CIPC ISSUANCE MECHANISM

The Cosmic Interstellar Public Chain network includes its own built-in currency, CIPC, whose dual purpose is to provide a primary liquidity layer to allow efficient exchanges between various types of digital assets and, more importantly, to provide a payment transaction fee mechanism.

The distribution model is as follows:
A total of 9.7 billion CIPC tokens were issued

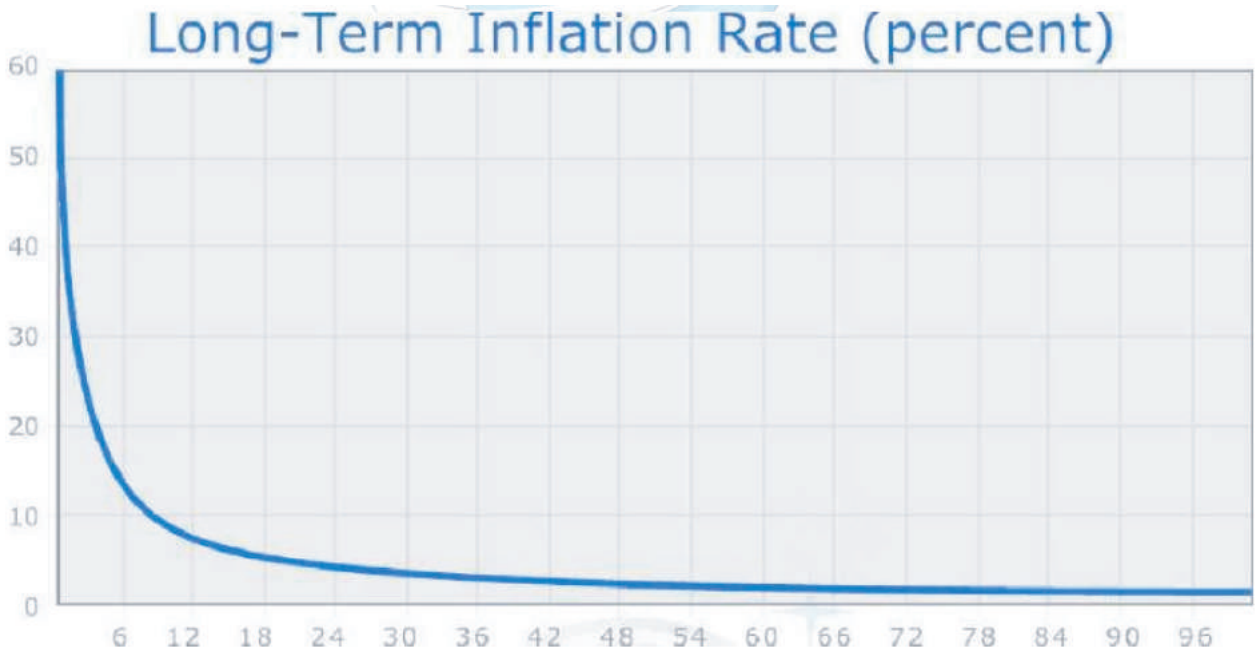


CIPC coins will be sold at a negotiated price, a mechanism designed to fund the Cosmic Interstellar Public Chain organization and pay for development, which has been successfully used by other platforms such as COINLIST. Earlier buyers will benefit from bigger discounts. All proceeds from the sale will be used to pay salaries and bounties to developers and invest them in various for-profit and non-profit projects within the Cosmic Interstellar Public Chain and cryptocurrency ecosystem.

From then on, 0.26 times the total sales are permanently distributed to miners each year.

CIPC	When launching	1 year later	5 years later
Currency Unit	1.198 X	1.458 X	2.498 X
Buyer	83.5%	68.6%	40.0%
Pre-sale reserve expenses	8.26%	6.79%	3.96%
Second-hand reserve after sale	8.26%	6.79%	3.96%
Miner	0%	17.8%	52.0%

Long-term supply growth rate (percentage)



Even though money is issued linearly, like Bitcoin, the supply growth rate tends to zero over time.

The two main choices in the model above are(1)the existence and size of the endowment fund, and (2) a linear supply that grows forever, rather than a capped supply like Bitcoin. The rationale for the donation pool is as follows. If the endowment fund did not exist and the linear issuance was reduced to 0.217x to provide the same inflation rate, the total amount of the Cosmic Interstellar Public Chain would be reduced by 16.5% and thus the value of each unit would increase by 19.8%.

The perpetual linear supply growth model reduces the risk of what some see as an excessive concentration of wealth in Bitcoin and provides a fair opportunity for those living in present and future times to acquire units of currency, while retaining a strong incentive to acquire and hold currency Ether, because the percentage of "supply growth rate" will still tend to zero over time. We also theorize that since coins are always lost over time due to carelessness, death, etc., and coin loss can be modeled as a percentage of the total supply per year, the total money supply in circulation will actually eventually stabilize at on a value. Equals annual issuance divided by loss rate (e.g. 1% loss rate, once supply reaches 26x, 0.26x will be mined and 0.26x will be lost per year to create a balance).

9. DEVELOPMENT PLANNING

On November 21, 2022, CIPC participated in and became one of the management functions of Crypto Industry Development Monitoring and Reporting Association. It was also designated as a public chain for national use by the Türkiye government. It cooperated with Hazine ve maliye bakanlığı, Mali Suçları araştırma kurulu, sermaye piyasası kurulu and Türkiye cumhuriyeti merkez bankası, and participated in the blockchain industry management. To help develop the cryptocurrency industry and carry out international monitoring and management of cryptocurrencies and exchanges.

The tight integration of real-world hardware with blockchain and native tokens is a new and valuable innovation that can be applied to other types of networks and physical layers. The future of blockchain is not about who has the most computing power or access to the cheapest computing power, but is about proof of mining in the blockchain linked to providing a valuable and verifiable service. CIPC has taken or intends to take and initiate a number of development processes, including but not limited to:

• Investigate the applicability of applying these ideas to other hardware physical layers such as smartphones, cars, home appliances

• Explore the potential to provide 5g 60GHz + mmWave connectivity with similar designs

• Research and implement more consensus proofs to ensure the CIPC network remains secure during development

• Game theory analysis of incentive systems • Exploring higher caps for kubernetes cluster management containers

• Scoring algorithm used by miners

• Create and publish CIPC wireless specifications

• Manufacture of relatively dedicated equipment modules to be available at CIPC network startup 36
• Investigate the deployment of smart contract environments beyond basic CIPC primitives
• Powerful architecture and design patterns for scalable smart contracts
• Blockchain Interoperability
• Ongoing work and development of forward error correction techniques

10. CONCLUSION

The Cosmic Interstellar Public Chain protocol was originally conceived as an upgraded version of cryptocurrencies, providing advanced features such as blockchain custody, withdrawal limits, financial contracts, gambling markets, etc. through a highly general programming language. The Cosmic Interstellar Public Chain protocol will not directly "support" any application, but the existence of a Turing-complete programming language means that in theory arbitrary contracts can be created for any transaction type or application. However, what is more interesting about the Cosmic Interstellar Public Chain is that the Cosmic Interstellar Public Chain protocol has gone far beyond currency. Among many other such concepts, protocols around decentralized file storage, decentralized computing, and decentralized prediction markets have the potential to greatly increase the efficiency of the computing industry and greatly facilitate the development of other peer-to-peer protocols by adding: The first layer is the economic layer, and finally, there are tons of applications that have nothing to do with money.

The concept of arbitrary state transition capabilities implemented by the Cosmic Interstellar Public Chain protocol provides a unique potential for the platform. The Cosmic Interstellar Public Chain is not a closed, single-purpose protocol intended for specific applications in data storage, gaming, or finance, but is designed to be open, and we think it is very suitable as an infrastructure, in the coming years, This protocol layer can be used for a large number of financial and non-financial protocols.

The development of blockchain has encountered bottlenecks in performance and scalability, and the next-generation breakthrough platform is urgently needed. Cosmic Interstellar Public Chain is based on modern cryptography, communication technology, and distributed computing, carrying decentralized finance, social networking, e-commerce, Search and store commercial applications, and build a large-scale commercial high-performance blockchain ecosystem.
The Cosmic Interstellar Public Chain, like many community-driven open source software projects, has grown since its initial inception. To learn about the latest developments in the Cosmic Interstellar Public Chain and how to make changes to the protocol, we recommend this guide.