A Self-Forking, Zero-Cost, Sub-Second Blockchain Protocol[1]

0chain.net

Jan 22, 2018

Abstract

As the world transforms to a tokenized economy, there is a need for an efficient scalable protocol for typical web, enterprise, and IoT applications. 0chain provides a *zero-cost*, *fastest finality*, *infinitely scalable* blockchain for web and IoT applications, essentially providing a *zero-cost decentralized cloud.* 0chain enables current DApps to move their off-chain code and data onto our decentralized compute and storage platform. Its *self-forking* feature enables different verticals and applications to fine-tune their needs create separate chains without worrying about the integrity of the blockchain. Unlike a traditional cloud subscription model, DApps need to *hold 0chain tokens* to use the blockchain, more like a *bank* for a free scalable cloud, and as more applications use our network, 0chain will grow in its intrinsic value and integrity.

1 Motivation

1.1 Scaling issue

Conventional blockchain technology does not scale and has a high economic cost of consensus, which makes it difficult to use for IoT devices and micro-transactions. IoT devices and micro-transactions typically send a lot of data and so, cumulative costs for such transactions would be too high for a business to be able to use such data. Take for example the fees of Bitcoin is on average $2.25 per transaction, with Ethereum around $0.41[2]. This may be fine for high value transactions, but for a single IoT device transmitting every minute, it would cost $215k annually on Ethereum, unless we register most of them *off-chain* and record periodically some values on the blockchain. Additionally, the number of transactions per second that can be executed for Bitcoin is about 3, and Ethereum is between 5 to 15, far short of what we need. Consider an IoT application, which has an installation of 6M sensors, each transmitting every minute; we need a blockchain that can accommodate at least 100k transactions per second, something that none of the current blockchains support today.

1.2 Energy waste

Traditional blockchain technology such as Bitcoin[3] and its derivatives use work-oriented schemes (proof-of-work) to build consensus and advance a block. This scheme wastes energy resources, and needs specialized computing power. Indeed the hashing power requirement is so large today that only a few pools mine the bulk of the blocks. This practical economic effect runs against the original purpose of decentralization.

1.3 Resource scaling issue

A more recent blockchain technology, Ethereum[4], has incorporated scripts within transactions and use compute, memory, storage, and bandwidth resources. While the flexibility of a Turing complete smart contract enables new applications, it complicates the mining process and puts a strain on the resources. This led to charging fees (gas) to force contract developers to restrict contract compute and storage capability. Hence, most of the applications today have computations architected to be off-chain because on-chain computations are too slow and expensive. Depending on the implementation of the code, the gas cost varies from $0.05 to $3, and is also tied to the value of Ether[5]. As Ether token value appreciates, the gas cost increases. Indeed, for just one IoT device, the gas cost could easily be $215k annually depending on the number of smart contracts being used to convert raw data to calibrated visual and analytic sets. The only way to speed up Ethereum is to have *sidechains* or off-chain transactions which occasionally pegs back to the main chain. While this may be a band-aid for current transactions, it will be very difficult for any IoT or web application to work in this scenario. In the future, Ethereum is expected to adopt Proof-of-Stake, which should

[1] *Patent pending*

[2] https://bitinfocharts.com/comparison/bitcoin-transactionfees.html

[3] https://bitcoin.org/bitcoin.pdf

[4] https://github.com/ethereum/wiki/wiki/White-Paper

[5] https://hackernoon.com/ether-purchase-power-df40a38c5a2f

alleviate scaling problems, but Casper[6] or Plasma[7] have a complicated design and economic incentives of a hybrid Proof-of-Stake and Proof-of-Work protocol with fraud proofs between the two chains.

1.4 Forking issue

Several prominent blockchains (Bitcoin, Ethereum) have gone through the forking process and this period is destabilizing because of uncertainties[8] over the integrity of the forked chain, miner economic incentives, and user demand. Forks happen because of the need to change the code that cannot be done with a minor upgrade, and is necessary to meet certain application requirements that were not thought of in the initial design. An additional reason for a fork is to reverse a malicious transaction that has taken place because of an implementation flaw (DAO[9]). Bitcoin and Bitcoin Cash [10] went through a volatile period after a hard fork event, where the latter's token value fluctuated between $400 and $1000 within a day.

1.5 Inflation & Volatility

Both Bitcoin and Ethereum have a very high inflation rate of mining, although it does reduce over time. Bitcoin started out with 100% before settling to its current 4% inflation rate. Ethereum's current inflation rate is about 14% but is expected to reduce after a hard fork in future. Even then, there is too much reward going toward the miners — in fact, Bitcoin miners have earned $2 billion since its inception[11]. This miner economy is not efficient and would hamper growth of truly decentralized applications, that desire a protocol with a fair computing and consensus price and use of less energy resources. Both Bitcoin and Ethereum have a very high volatility history. Bitcoin lost 30% of its value within 48 hours of Jamie Dimon's[12] comments, and Ethereum lost 20% after fake news surfaced on Vitalik's[13] car crash. Our protocol grows its intrinsic value as the utility of the applications on our network increases over time.

2 Multi-Dimensional Blockchain

2.1 Multiple Dimensions

We propose a novel blockchain that solves the problem of cost, scalability, fork instabilities, and high inflation. Our blockchain is *n-dimensional* with multiple chains based on different *forkable* parameters detailed later in Section 3, with incentives for consensus, computing, and storage entities, *miners*, *sharders*, *blobbers,* to scale the blockchain with a high level of integrity and security. See Fig. 1. The miners generate and validate a block, sharders store the blocks, and blobbers store unstructured data. With multiple chains under one native token, we enable multiple verticals to be satisfied with forkable parameters, without the need for a new blockchain. The sharders help reduce compute, memory, storage, and bandwidth requirements of the underlying hardware implementation, and allow for fast indexing and access of data and code. Note that the definition of sharding here is different from that described in Ethereum and Zilliqa[14], where sharding implies different consensus sets. In our case, sharding is defined as splitting the chain for manageable storage and access. The blobbers help reduce the cost and complexity of storing content for web and IoT applications. The purpose of having multiple blockchains is for a single protocol to be applicable for various verticals, and to decouple the value of the underlying token with its utility for mining, sharding, and blobbing activities.

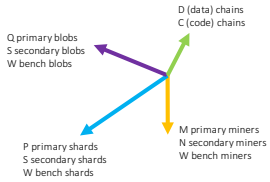

Fig 1. n-Dimensional protocol scalability with chains, miners, shards, and blobs

---

[6] https://github.com/ethereum/research/blob/master/papers/casper-basics/casper_basics.pdf

[7] http://plasma.io/plasma.pdf

[8] https://www.coindesk.com/ether-price-fluctuating-ethereum-fork-concerns/

[9] https://www.coindesk.com/ethereum-executes-blockchain-hard-fork-return-dao-investor-funds/

[10] https://cointelegraph.com/news/bitcoin-fork-cash-volatility-continues-while-bitcoin-price-remains-stable

[11] https://bitcoinist.com/bitcoin-miners-2-billion-since-2008/

[12] https://www.usatoday.com/story/money/markets/2017/09/15/bitcoin-loses-third-its-value-month-after-400-run-up-2017/668259001/

[13] http://mashable.com/2017/06/26/ethereum-price-drop/#zZ8oMHVXHiqa

[14] https://www.zilliqa.com

2.2 Code and Data Chains

There are protocols today that have the concept of sidechains[15] for better speed and scalability. In the case of Blockstream, the sidechains are pegged to the BitCoin network, and so merged-mining can be used for verification of the blocks on the sidechains to prevent hash attacks on a new set of miners. In a similar vein, Plasma.io's concept is expected to enable micro-transactions on its off-chain and periodically use fraud proofs to peg the states back to the Ethereum network. These changes to the legacy Bitcoin and Ethereum may help patch up scalability of transactions but seem too complex, unstructured, and expensive for a web application that needs to scale deterministically at a low cost. As shown in Fig. 2, we present two forked blockchains from the genesis block on the 0chain network. The purpose of having two chains at the very onset of the network is to separate the transactions into *stateful* and *stateless* buckets. The separation is easier for development as it would then conform to the MVC (Model-View-Controller) architecture that is used by most enterprise grade applications, where model represents data or the database, the controller embody methods that work with data and change states, and the view typifies visualization of the data by the client. The data-code separation places different memory requirement for the miner's infrastructure. A stateful chain would need all the states to be in memory to facilitate changes to its states. A stateless chain can be placed in SSD or Disk depending on the frequency of access. There is hardly any memory requirement to process a *data* transaction, as there is no need to know the previous state.

An IoT data set, *Oracles* (real events represented on the blockchain), or published content are examples of 'stateless' data that has no memory or coding requirement for such a transaction. The transactions can be processed much faster and be kept in SSD or disk after the block is mined. And so, a block time for such a chain can be set to be a shorter time compared to a 'stateful' chain. The miner incentives for a data chain is expected to be less than the code chain because the infrastructure cost would be much less.

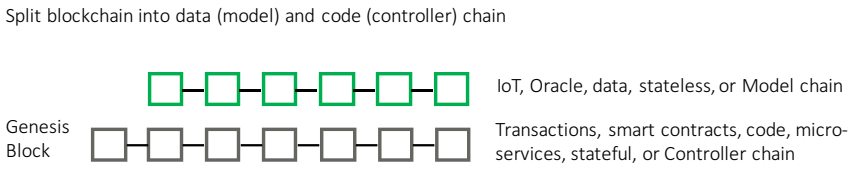

Fig 2 Dual-chain blockchain protocol

A micro-transaction, such as paying for coffee, or a bunch of micro-services, that converts raw IoT data to calibrated data, or to different datasets such as hourly and day data averages, are examples of 'stateful' code, that needs states, and the code needs to be loaded in memory for faster execution. For a large enterprise application, which may have 1000s of micro-service calls, it makes sense to have all the states and byte-code in memory to achieve a result faster than if it were constrained by disk I/O of SSDs. One can conceivably use a larger block time for a code chain for applications that require larger processing time to generate an output.

2.3 Self-forking Multiple Chains

Fig 3 further depicts the self-forked chain dimension of the blockchain. The self-forking process would involve 2/3 majority of stakeholder votes to allow for a fork proposal. After such a fork has been established, an initial set of miners are chosen for the future chain. For a transaction to be sent to this chain, an easy implementation would be to address the transaction to a specific chain address. If there is no address, the transaction would default to the genesis chain (which is a code chain). Miners on each chain ignore transactions from their buffer queue that are not addressed to them. Depending on whether the data or code is sent to a code or data chain, the miner will treat it as such. So, a byte code sent to a data chain will be treated as just a piece of data, while if sent to the code chain, it will be executed as code.

[15] https://blockstream.com/strong-federations.pdf

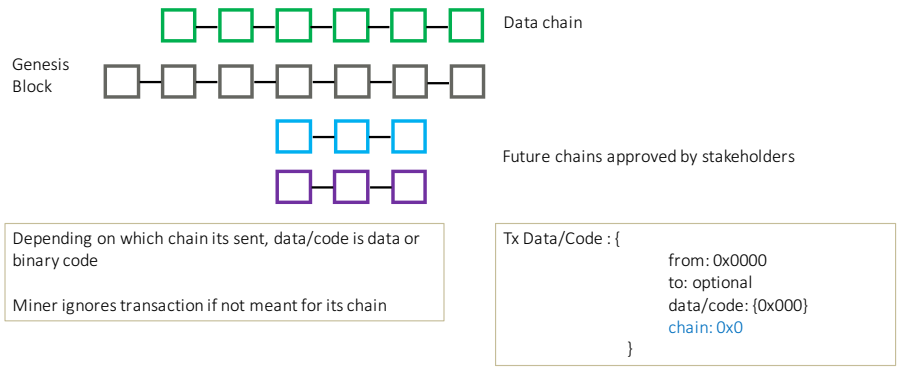

Fig. 3 Self-forking chain

2.4 Miner MxN dimension

Fig. 4 shows the miner dimension of the blockchain in a typical DPoS [16] (delegated proof-of-stake) configuration, where M miners are delegated by stakeholders through a voting process. As is typical in a DPoS scheme, one miner produces a block while others verify the block. In order to make the consensus deterministic, instead of probabilistic (Bitcoin, Ethereum, others) we looked at several algorithms such as PBFT[17] and Paxos[18] to address this Byzantine problem. We have drawn inspiration from Paxos and DPoS schemes, and have innovated around the ability to produce a deterministic consensus within a round of generating a block to be able to confirm it, and have the fastest finality compared to all the blockchains, where a round is defined as the progression of the block through M miners. The miners are then shuffled in a random order by a scheme detailed later in the paper. In our innovative scheme, we add an additional group of miners that back up the M set of primary miners and form a MxN set where M are the designated primary miners and N are the secondary miners. The purpose of the backup miners is to prevent malicious transactions, DDoS attacks, withholding and censorship by primary miners. Additionally, if the primary miner is offline or has an unusually high latency, backup miners would be able to advance a block to the network. Out of the N blocks generated during the block production slot, if n/(2n+1) have the same *Block Hash*[19] then that block is selected to be added to the chain, where n is number of total miners in the active set that need to agree to confirm a block. Typically, the N set would be a small set, otherwise it would be similar to proof-of-work where all the miners are generating blocks. This architecture results in a *dynamic decentralization* contrary to traditional *static* decentralization on Ethereuem, Zilliqa, and others. In this scenario, as our bench miner pool grows we don't need a big miner set, because the probability of an attack dwindles without the need to engage a large miner set. Not sure I understand their argument on scalability.

If the set of miners is kept small, then the clients can conduct a fast and easy validation of the last m blocks produced to determine if a transaction has been processed. SPV (simple payment validation), sometimes referred to as a light client validation can be easily done by syncing with one of the miner nodes, compared to proof-of-work or naïve proof-of-stake. In the latter case all the nodes are producing blocks and uncles and there is no way to verify that the node is malicious and has the finalized blocks, other than conducting a proper Merkle validation at the light client or completely syncing with the node. The way it is done now for Ethereum is to download the block headers and use a distributed hash table for trie nodes to verify a transaction, account balance, validate a block, or monitor an event. In our MxN set, only a few miners exist and so by connecting to the miners, the client can sync up much faster to a node, and once the client establishes the node to be *honest*, then it can query the node for specific transactions, account balance, validate a block, or monitor an event. To prove an *honest* node, one needs to verify the signatures of the mined blocks, or one can compare the Block Hash of the latest blocks from the M miners and determine if they are consistent.

[16] https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md

[17] http://pmg.csail.mit.edu/papers/osdi99.pdf

[18] https://lamport.azurewebsites.net/pubs/paxos-simple.pdf

[19] Block Hash is defined as a collection of all hashed transactions

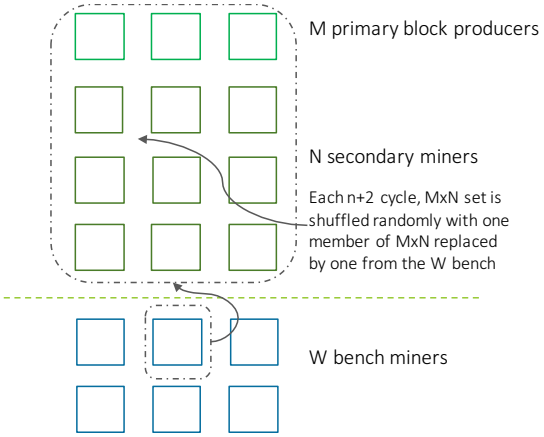

Fig. 4 Miner dimension of the blockchain consisting of primary, secondary, and bench miners

2.5 Shuffling scheme

The shuffling of the MxN set is critical as it determines the proper decentralization process, otherwise attackers can hone in on one miner or be that miner that generates the random seed. The idea of generating a random seed is inspired by the method proposed by Tezos[20]. Each MxN miner generates a hash of a random number in one cycle and post it on the data chain, and in the next cycle the miners reveal their random numbers, and then the resulting seed is generated from the combined random numbers of the miners, and is used to deterministically map to a MxN set, such that at least one member of the MxN set is dropped in favor of another from the W bench set. In the next cycle, the miners use the random seed to determine if they are on the active set, and if they are in the primary or secondary category. In this way, apriori knowledge of miner status is avoided to prevent focused attack on a specific miner. The whole process of generating a shuffled MxN set with a new member from the bench is recorded and can be verified by anyone. Lets consider some malicious attack scenarios.

> *Selfish Mining* — if a miner wants to stay on the MxN set for selfish mining, the miner needs to be either lucky or collude with all the miners since it is impossible to have the same hash if one miner does not collaborate. This prevents the miner from selecting itself in the primary spot or from exiting the MxN set.
>
> *Bad Transaction (e.g. Double Spend)* — for this to happen, the miner needs to be lucky to collude with or be Sybil of n/(2n+1) other miners, where n is the total number of miners in the active set. So for 3 primary, 6 secondaries, and 6 bench miners, a successful attack needs at least 5/9 majority to conduct a successful attack, but the probability of such an attack is dependent on miner selection from the bench set and is about 4.2%. With more bench miners, the probability decreases even further. This is a pretty good scenario relative to Bitcoin and Ethereum, where with a 51% control you can conduct an attack at any time.

2.6 Matching scheme

Each M block producer is focused on producing a block, and include as many transactions as it can within a specific wall time, or until all the transactions in the queue are exhausted. The hashes of all the transactions are then combined separately, to produce a Block Hash, which is then compared with the secondary block producers. If the Block Hashes *match*, then the block is considered an *almost valid* block, and hence only one block time is required for finality in such a case. If the primary and secondary miners have non-matching Block Hashes, then the primary miner's block is chosen to be added to the chain and later validated by the other miners in the MxN set. It is assumed that the algorithm will be such that the *same number of transactions* will be selected by the primary and secondary miners, as long as they both have similar types of computing resources, and have enough guard time to account for network issues. Its important that the stakeholders stipulate to the miners to have a *similar computing* environment to benefit the protocol and miners, and unlike other blockchain protocols, these select groups of miners need to have a high network bandwidth connection to the internet.

When a primary miner is engaged in producing a block, the other miners are busy *validating* the previous blocks. The miners in the MxN set get rewarded every production cycle, and their rewards are set periodically by the stakeholders. The rewards are

[20] https://www.tezos.com/static/papers/white_paper.pdf

based on miner's *bid* price and the miner selection protocol would be based on *the reputation score* of a miner. A miner needs to meet the minimum computing requirement to be a part of the mining pool. A miner needs to reserve a percentage of their total earning potential over the course of a period. A miner's reserve is "locked" before, during, and after the mining cycle for n cycles, before they are released. When the reserve is locked, a miner cannot make a withdrawal from that portion of their fund.

2.7 Sharding dimension PxS

As the chain gets substantially bigger, the miners will need to add memory or storage or both, depending on the chain they are working on. To speed up data access and reduce memory and/or storage requirements, the chain is sharded after it reaches a certain size. See Fig. 5. Thus, there will be multiple shards and with a good indexing scheme, access for data and code would be much faster. The shards are either kept in memory as for a code chain, or in SSDs if it is storing data. The shards will have a replica set so that if the primary shard is offline or fails or is a fraud entity, then the secondary (replicated) ones can provide the relevant data or code. Miners communicate with Sharders to complete transactions. Sharders are expected to have similar processing, bandwidth, and memory to keep processes balanced and maintain a synchronous operation most of the time.

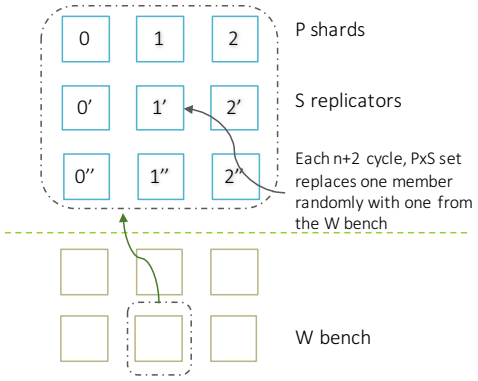

Fig 5. Sharding dimension of the blockchain consisting of primary, secondary, and bench sharders

In an enterprise application, it is conceivable that multiple calls are placed from the code chain to an older sharded code chain for older smart contracts, which gets executed at those shards or a single shard spawning multiple processes, and the results sent back to the current code, and is subsequently published in the current data chain. So, the infrastructure requirement of a miner is a bit different from a sharder. Sharders do not need as much computing power, even in the case of code sharders, since they are not doing any hashing or verification activities. The method calls between the miner and the relevant shard(s) is through a secured API call. The expectation is that sharders will have good connectivity on the network to minimize latency. The sharders, like miners, collect tokens after each n cycles.

Lets consider attack scenarios. A sharder can *cheat* in several ways. It can pretend to be several active sharders (*Sybil* attack) where it pretends to shard even though it does not hold any data or code, and hope that it does not get caught by a miner's call during the period it is serving as a sharder. A sharder can *outsource* its sharding activity in the sense that if a miner calls a particular block, it just relays the call to another shard. To prevent these two attacks, we force the miner to randomly choose a sharder for old contract calls. This randomness can be deduced from a different mapping function, but the input is derived earlier from the same random seed.

Now there is a possibility where a miner is colluding with a sharder, or could be the same entity (Sybil). In this case, the miner would not call that particular sharder, but then the other miners would have to collude as well for this sharder to hide, which is a remote possibility.

One last attack scenario is when all the replica is a Sybil version of the sharder for a particular allocation. In this case, there is no way to verify that the replica is genuine. To solve this, we propose that a sharder is replaced from its set every n+2 cycles with one from the bench. This ensures that any Sybil entries would be short-lived, and the integrity of the network would be restored,

if violated. The sync of the new member takes some time depending on the size of the shard, and after its completed, the new shard set is created and active.

To prevent any long range attacks, we have a *Shard Hash*[21] for every shard, so that if an attacker goes back in time and changes a block, it would be reflected at the Shard header.

2.8 Blobbing dimension QxS

Fig. 6 shows the blob dimension of the blockchain designed to provide storage capacity to data that cannot fit into the transaction data limit size. Blobs are defined as storage entities that can store large, unstructured data (image, audio, video) and is based on IPFS protocol for easy and efficient retrieval by anyone, and propagate during demand spikes through client nodes without burdening our network. Unlike FileCoin[22], we enable fast store and retrieval of the stored data with no induced latency, something that Filecoin does to enable replication integrity. Additionally, we do not cater to Retrieval markets, and let the web applications perform such a function or collaborate with a CDN if necessary. If the IPFS works properly, then during spike events, the content will be served from the clients, instead of from our network. Also, our group of blobbers would typically compose of a select group of enterprise-grade quality storage infrastructure companies with high network bandwidth connectivity as opposed to home storage devices. The blobbers on our network earn a collective token based on amount of data added per n cycles. There are QxS blobbers and they get rewarded equally as data flows in, based on the size of data in n cycles. Data is distributed evenly among Q blobbers. If data is uploaded via the data or code chain and it exceeds the size limit of the data chain, it is sent to the blob assuming the user has enough reserve token for such data to stay persistent, otherwise the transaction would be invalid. A user may indicate to the network via a transaction that it wants to prune some files to reduce its locked token commitment. The blobber holding the files would then mark those files for deletion. For each request for a blob, the requestor needs to have enough unlocked tokens in their account. As new storage providers are accepted by stakeholders, they are first *benched* and later *added* to the list of QxS blobber set as storage demand grows. Blob rewards are based on the *bid* rate of the blobbers. Blobs are impossible to change or fake because it would otherwise result in a new hash which will not match with the content hash on the data chain.

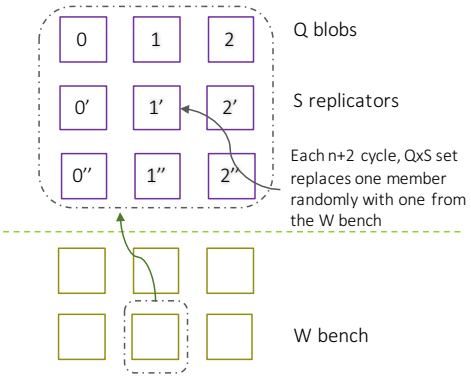

Fig 6. Blobbing dimension of the blockchain consisting of primary, secondary, and bench blobbers

Unlike FileCoin there is no bid and ask process of selecting storage providers. As with Filecoin, which need proofs to prevent Sybil (pretend to store data as several entities), outsourcing (commit to store more data) and generation (claim to store more data) attacks, we need to consider these attacks as well.

The *generation* attack is a net loss for the attacker because the attacker needs to hold tokens to store data and any reward token is divided up among other blobbers. Of course, it is possible for all blobbers to collude and start generating for the well-being of the blobber community. Although they would still need to collectively hold more tokens than they get rewarded, we can further discourage this behavior by making the storage token requirement to be greater exponentially than the reward token for spike volumes, which would not normally happen, since content creation takes time. The *outsourcing* attack is prevented by the same

[21] Shard Hash is defined as a hash of all the block hashes

[22] https://filecoin.io/filecoin.pdf

algorithm as discussed in the Sharding section 2.7, where the miner randomly calls a replica of a blob allocation. The *Sybil* attack is prevented by the same algorithm proposed in section 2.7, where one of the blobbers in the QxS set is replaced by one from the bench. The sync time may take a little longer than for the shard as the storage is expected to be much larger by at least 2 orders of magnitude. However, this creation and deletion process would prevent a fake replica from existing on the network. We could have used FileCoin's PoRep[23] protocol, but that leads to computing and network issues for storage providers. Our protocol is simpler, and if there is a good network connection, and the blob sizes are kept reasonably small, then the sync process will not take that much time. The network rewards the blobbers tokens based on the storage used. However, it is important to note that there is no reward for retrieval, but if a blobber has a high latency or if there is no data, then they will be punished during the retrieval process. Since a blobber gets their token rewards after n cycles of service, they are held accountable to any problems during the retrieval process, whether that is due to bad bandwidth or any malicious activity.

2.9 Application example with multiple chains, shards, and blobs

Fig. 7 shows how a contract (e.g. parse Oracle file) in codeA chain calls a utility contract in ShardA 0 of code A chain, which calls another contract (e.g. get Oracle file) in ShardB 0 of code B chain, which retrieves a content address from ShardD 2 of the data chain, and then retrieves the actual file (Oracle file) from Blob 2. Finally, the codeA contract parses the file and outputs a particular data, and records it on the data chain (mined on the next block). This is similar to an enterprise micro-services architecture for agile web development where different teams or businesses develop different services and updates their codes. An example would be companyB uses codeB chain because of larger block times to perform data analytics. CompanyA can request that service from their contract in codeA chain, which has a smaller block time and get the results posted on the data chain.

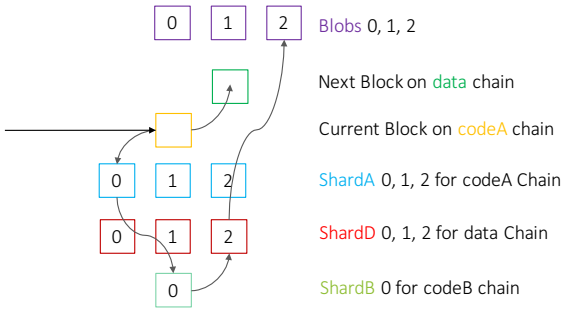

Fig 7. Calls to different chains, shards, and blobs from a single transaction

2.10 Logically separate miners, sharders, blobbers

Fig. 8 shows that a logically different set of miners, sharders, and blobbers exist for different chains. However, the same physical miner can have different mining nodes for different chains. So, if a miner, sharder, and blobber have a good reputation on one chain, then they will have a good chance of being included on a different chain. So, the miners can develop the same trusted reputation as *merged-mining* in Bitcoin, and new chains will not have suffer any negative consequences of "starting over".

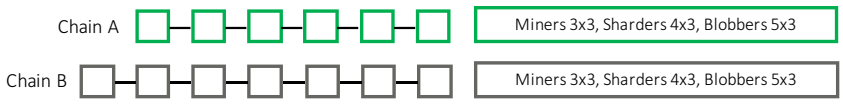

Fig. 8 Logical separation of miners, sharders, and blobbers for different chains

2.11 Transaction Processing

The transaction processing of a block is shown in Fig. 9. The transactions are queued in multiple threads, sorted based on priority of their stakes before they are placed in the block. The multiple threads can be processed in one server or a cluster of servers.

[23] https://filecoin.io/proof-of-replication.pdf

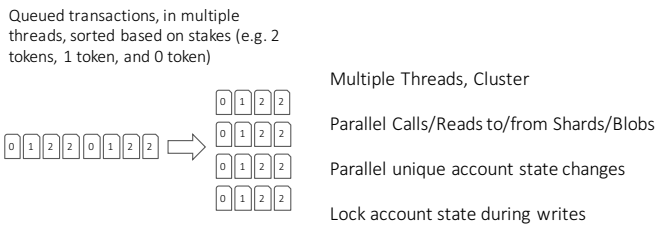

Fig. 9 Transaction processing sequence inside a block

Each transaction is processed serially within a thread, although since there are multiple threads or cluster of servers. However, states can only be changed by the same transaction and not concurrently. And so, within a block, there can be multiple transactions with parallel reads and calls, and if there are no concurrent state changes, they will be successfully included in the block. So, if a state is locked by one transaction during the block processing time, then all other transactions that try to change that state will have to wait their turn or fail if it goes beyond its allocated wall time, and will not be included in the block. This architectural implementation makes sure that all code will be thread-safe, regardless of the number of parallel transactions.

If a malicious transaction calls itself recursively and try to change state, then it will fail. If a transaction changes multiple states, such as paying several entities, then it is acceptable if those entities are not locked by another transaction. A malicious transaction can also try to call several contracts to waste a lot of computing power, memory, and bandwidth; but by doing so, it needs to reserve tokens and its intent would be hampered by the wall time.

The bandwidth, computation, memory, and storage are all based on an individual's stake. For storage, the user needs to lock their tokens. For transactions, they need to have enough tokens for n cycles before and after the transactions to guarantee bandwidth, compute, memory to place a transaction. Having very few tokens in the account does not mean a transaction will not go through, but it will be sent at a much lower priority, and will take time. If there are 100,000 users, and the block has transactions from 100 users, and each of them have similar stakes, then the bandwidth, compute, memory, and storage size per user is 1/100 percentage of the node's capacity. In the case of IoT devices or Oracles, the application is responsible for holding tokens to guarantee availability of the required bandwidth, compute, memory, and storage size per device.

3 Forking parameters

3.1 Stateless and Stateful chains

Fig. 10 shows the different types of parameters that can lead to a forked chain. The code and data chains are examples of a *stateless* and a *stateful* chain. Today multiple chains co-exist with different set of rules, but there is no protocol that allow for multiple chains to co-exist under the same network topology, and multiple tokenized application may choose to use multiple chains as they fit.

3.2 Block Time

Block time is another parameter that has forked many chains in the past, where the need for several transactions is balanced against the time to finalize a transaction. With a shorter block time, transactions within a block can get finalized faster. But with a shorter block, there will less number of transactions in the block and a higher overhead for the block. Additionally, if a code chain has a short block time, and if a code takes a longer time to execute then it can never be included in the block unless it is modified. However, in the case of micro-payments, the transaction finality is important. For example, the merchant should not have to wait for an hour to have their service tokens for a cup of coffee to be finalized.

3.3 Resource Usage and Fees

The fees for compute, bandwidth, and memory usage for a transaction is typically what miners charge to prioritize a transaction over others in BitCoin and Ethereum. While this works for high value payment transactions, other applications such as IoT and web applications, need to have very little fees or none imposed on their computation, memory, bandwidth and storage. There are several decentralized storage services offered by Storj, MaidSafe, Siacoin, and Filecoin. They are based on order matching bidders and sellers, but need recurrent proofs and transactions to be able to create a viable marketplace. However, because of the extra layer of complexity, computing, and bandwidth required to generate this market, it would be difficult to keep the cost down.

Data Type (stateful, stateless, …)

Block Time (100ms, 1s, 3s, 15s, …)

Data Size (1MB, 2MB, 100kB, …)

Transaction time (100us, 100ms, 1s, based on stake, …)

Compute, Bandwidth, Memory, Storage allocation (free based on stake, …)

Finality (sub-second, second(s), minute(s), …)

Number of Block Producers (3, 21, 100, 2048, …)

Number of Secondaries (zero, one set, two sets, …)

Number on Bench (zero, one set, two sets, …)

Signatures (ECDSA, BLS, Lamport, …)

Anonymity (none, zk-SNARKs, zk-STARKs, RingCT, …)

Verification – Merkle, Miner signature, zk-STARK

Shard size (10 GB, …)

Blob size (1 TB, …)

User or Application reserve tokens (1%, 10%,…) for transactions and storage

Lock Deposit for Miners, Sharders, Blobbers ($10k, $100k, …) for n (2, 3, 10, …) cycles after inclusion in the set to prevent malicious activity

Reward p tokens to Miners, Sharders, Blobbers (mean, median, or minimum of ask price)

Fig. 10 Forking parameters for the blockchain protocol

Additionally, individual storage on home computers cannot effectively compete with enterprise data centers with respect to scale, cost, and reliability. So, these services may work for users seeking "decentralization" and willing to pay more. Since IoT and other Enterprise applications need an enterprise level storage but at a low cost, 0chain network would bear that cost for all applications. Users or applications storing the data permanently need to lock some tokens permanently. Users or applications accessing the data need to reserve some tokens for a volume of transaction expected in the future, otherwise their transaction will be de-prioritized.

3.4 Number of Block Producers

The *number of block producers* determine the level of decentralization, and how fast the *finality* can take place. The minimum set of block producers need to be at least 3 for a 2/3 majority to validated a transaction. So, if one block producer mines a block, that block needs to be verified by the other 2 block producers, before a transaction is deemed finalized. The less number of verification enables less number of redundant computations required for verification and would serve to reduce energy and financial cost. As the number of block producers increases, the security and decentralization of the transaction increases.

3.5 Number of Secondaries

Based on the number of block producers, block time, and scheme, the transaction finality can be designed to be long or very short. In a typical Delegated-Proof-of-Stake, there is one block producer and the produced block is verified by 2/3 miners for it to be validated and added to the chain. If the number of block producers are small, say 3, the minimum for a proper decentralized system, then the finality is based on the block time of generating 3 blocks. Another way to reduce the finality is to add *secondary* miners. They serve the purpose of a quick validation and security over network loss or censorship. If you have at least 2 secondary set of miners, then for every primary block producer you have 2 secondaries that are producing the blocks in the same time slot. If 2/3 blocks have the same Block Hash, then that block is finalized for all practical purposes and is advanced on the chain. If the blocks are dissimilar, then the primary one is added to the chain. In most cases, the Block Hash will be same and the finality would then be the size of the block time. The secondary miners also protect transactions that suffer from data withholding, censorship, or if the node is offline. In the latter case, if the primary does not produce a block because it is offline, one of the secondary miner's block will be advanced to the network. The primary miner may also decide to withhold or a censor transaction from a block. If the secondary miners avoid such activity, and produce equivalent Block Hash that contradict the primary, then their block will be advanced instead, and the primary will be put on notice and they would forfeit their reserve. The *number of secondary miners* is an important parameter, as it speeds up *finality* and increases the chance of *data availability*.

3.6 Bench, Signature, Anonymity, Verification

The number of bench players for miners, sharders, and blobbers can be determined based on the level of integrity required for the chain. The Other forking parameters include type of signature, anonymity, verification, reserves, and rewards. Elliptic curve *signatures* are faster to encrypt messages compared to traditional RSA, and there are other signatures in development in the crypto community to consider depending on whether one desires more security such as Lamport or faster execution such as BLS. *Anonymity* is another parameter in Fig. 10 to consider to fork a chain. The anonymity of a user can be done through ring signatures, but to prove the validity of a computation, one needs to provide a verifiable proof of the transaction so that the network can verify it through such proof. Such is the intent of the zk-SNARK algorithm, but it has one issue — it needs a trusted setup. zk-STARK is an evolution of the SNARK algorithm and it does away the need for a master key. Verification is another

parameter that may change in the future. Today, every transaction is verified by *replaying* the transaction or looking at the *Block Hash*. Replaying the transaction provides a decentralized verification process, but this forces redundant operations wasting money and energy. For a 3x3 miner set, there will be 5 redundant operations to verify if the transaction is valid. If this validation is not a replay of the full transaction operation but a shorter verification enabled by zk-STARK like algorithm, then it will cut down on the energy and cost of the network.

3.7 Reserves

The *reserve* tokens necessary to make sure that miners do not engage in malicious activity may differ from one application to another. A simple IoT application may not need for a big reserve, but an exchange or bank operation miners may need to put up a bigger reserve and have a larger number of cycles to hold the reserve for every mined block.

The reserve token parameter for the user or application reserve tokens and the number of cycles its held are decided based on how much is needed to execute a transaction by using the network's compute power, bandwidth, memory, storage, code, and content. This may depend on the user and application in consideration that use the chain(s) and the amount of resources consumed. The reward token for the Miners, Sharders, and Blobbers determine the amount of *inflation* incurred by the network. The reward amount is decided based on the token value, the amount of reward activity, and the asking rate.

3.8 Initial chain parameters

The initial parameters chosen for the chains are as shown in Fig. 11. This would be the genesis chain for 0chain. The idea is to start with two chains (data and smart contract chains), have a sub-second finality, be scalable (infinite transactions) with concurrent threads and clusters, and have flexibility to allow future parallel chains for different verticials. All chains on 0chain have absolutely zero cost to the user for compute, memory, bandwidth, and storage, as long as they hold sufficient number of 0chain tokens.

Data Type – Stateful (code, smart contracts), Stateless (data) chains

Block Time – 300ms

Data Size – 1MB (4 threads)

Avg Transaction time – 25ms

Compute, Bandwidth, Memory, Storage resource usage – based on token stake

Finality – < 300ms-900ms

Number of Block Producers – 3

Number of Secondaries – 3x2

Number on Bench – 2x3

Signatures – ECDSA

Anonymity – none

Verification – Compare block hash, Miner signature

Shard size – 10 GB

Blob size – 1 TB

User or Application reserve x% token for transactions and storage

Lock $y deposit from Miners, Sharders, Blobbers over for m cycles after inclusion in the set

Reward p tokens for Miners, Sharders, Blobbers (median ask price)

Fig. 11 Initial parameters of the blockchain protocol

4 IoT platform + Web/Enterprise application

Fig. 12 shows an IoT platform architecture that can be implemented on the network. The data from an IoT device is sent to the data chain. The application then sends a transaction to call an appropriate smart contract on the code chain that acts on the data received from the IoT device, and generate new data sets such as calibrated data, averaged data, AI related data, and alerts. The data is placed either on the data chain or blob depending on the size of the dataset. With all the data on the blockchain, any client application can visualize the data by using a client browser or a hosted server that copies the data from the blockchain. In a similar way, one can visualize how a generic web or enterprise application can use this model to separately upload data to data chain or blob and upload microservices on to the code chain.

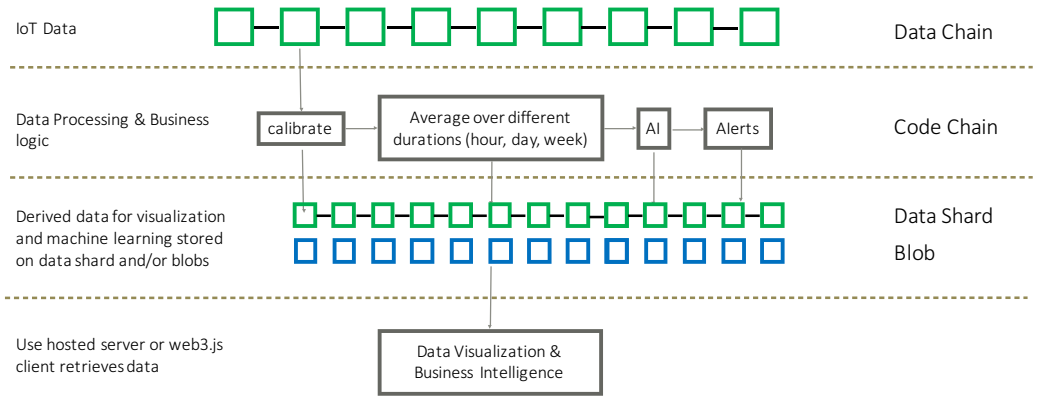

Fig. 12 IoT platform implementation on the blockchain

5 Reward Pool

Fig. 13 shows how the reward pool will be reduced over time, as *inflation protocol* regulate the number of tokens given out over a period. As the number of applications increase, the reward pool will be used more often for the miners, sharders, and blobbers. This may increase the inflation rate of the rewards. The inflation protocol sets the reward size based on the *bid* rate of the incentivized entities, and this rate may change daily as the value of the token changes, for the mutual benefit of the network and the miners.

The reward pool may last 100 years or more, depending on how the protocol control inflation, which is a function of the number of rewards, DApp hold rate, miner bid rate, and the value of the network token. If there is a big demand in the network token, say two times, then there will be as many rewards given out, but the value of token is likely to increase in value because of higher demand, thus causing the token bid price to drop. And so, the inflation rate is expected to remain about the same as it started out in the beginning.

Compared to Ethereum and Bitcoin, we would start the inflation much lower. It would be set at a 1% nominal rate, but will vary depending on number of transactions and storage needed by the applications, but will nominally look like as in Fig. 13.

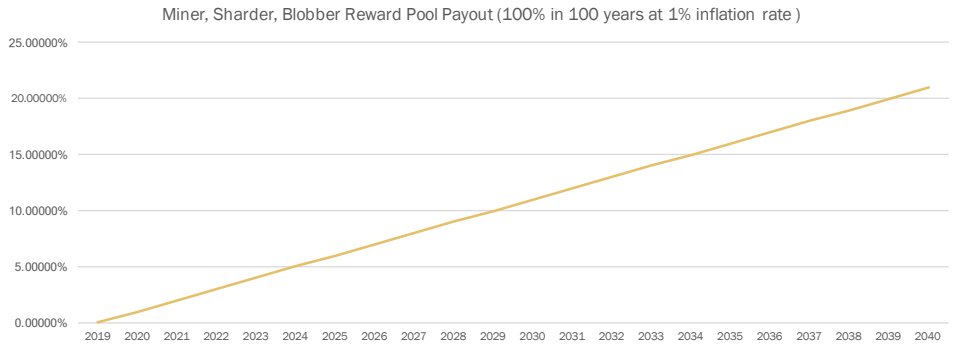

Fig. 13 Reward pool payout over 20 years

6 Attack Scenarios

6.1 Sybil

A Sybil attack happens when a malicious node pretends to be a miner and sends fake blocks to a client. This is hard to do in our network, because there are only a few miners on the MxN set at any given time. Even if the attacker is successful and pretends to be one of the miners, the client can easily verify if the miner's signature and Block Hashes are consistent with other nodes. Once an honest node is determined, then the client can latch on it unless that node is replaced by another from the bench, in which case the network will automatically handle the transition.

6.2 DoS

DoS (Denial of Service) attacks by the user can be easily traced and the user tokens can be frozen. The user needs to have ample tokens to perform this attack. So, there are mitigation steps that the network can take to prevent the user from flooding the network.

6.3 Double Spend

A double spend attack can happen if a miner colludes with 2/3+ miners, which is difficult, and self-defeating because of the stake the 2/3+ miners need to lock up for n cycles, not to mention their accounts frozen and transaction reversed or forked at the end.

6.4 Nothing-at-Stake

A Nothing at Stake attack happens when a miner does not have any stake and mines a block to all available forks to increase their chances of getting rewards. Such economic incentive is not applicable for our network. This was a problem for naïve proof-of-stake algorithms, but since all miners need to reserve stakes for n cycles, this is not an applicable attack scenario, unless miners want to forfeit their reserves and need to collude with at least 2/3 for MxN miners.

6.5 Long-Range

A Long-Range attack happens when a miner secretly creates blocks from genesis and then reveals itself for other nodes to accept its version of truth. This attack is not applicable because there is a designated slot for every assigned miner to create a block, and the hash of the block is signed by the miner. Every block is signed by the assigned miner which is shuffled in the MxN set. Additionally, all shards will have a hash of the entire shard as it fills up and would be difficult to duplicate with a bad block. For example, Shard 0 will not change and have a Shard Hash that can be verified against other shards. For a current shard, the Shard Hash changes as new blocks are added, but this is difficult to duplicate with a bad block created some time ago.

6.6 Initial Distribution or a new Fork

During the initial Distribution or at a new Fork, or in the case of a bribe, the miners could generate bad blocks, but since they have locked stakes, the opportunity cost for such activity is high. Additionally, it will be stipulated to have at least 50% of the existing miners on the new chain to preserve its integrity.

6.7 Censorship or Data availability

Censorship or withholding data are avoided by having one or more secondary miners. If the primary miner withholds data or appears to do so, the secondary miners block is accepted, as long as all the miner's computing platforms are similar.

6.8 Selfish mining

Selfish mining is traditionally meant only for proof-of-work systems where the miner secretly mines blocks and reveals them later to win the longest chain and hence win all the miner tokens away from other block creators. For this network, selfish mining can occur if it selects itself repeatedly in the MxN set, which it cannot do unless it colludes with others on the network.

6.9 Blobber attacks

Blobbers can have several attack scenarios identified by Filecoin. The Sybil attack, where the blobber pretends to store as multiple blob entities, will be prevented with the replacement algorithm, where a blob is periodically replaced with another one

from the bench. Blobbers pretending to commit more or claiming to store more can be avoided by having the miners commit to random calls during the retrieval process. And generation attacks are inherently lossy as the attacker would then need to purchase tokens.

6.10 User attacks

User attacks based on creating code that causes infinite loops, calls, reads, or writes are inherently prevented with a finite wall time for a transaction. The user can only attack until it has exhausted its allocated portion of token resources, which would be self-defeating, as they would need to buy more tokens to continue the attack. Additionally, these attacks would be continually monitored by the network and trigger an account from being frozen for some time until the activity is under control.

A large data upload attacks can be conducted by a malicious client. Since the upload needs to finish within the transaction wall time, the upload transaction would be terminated. If this behavior is persisted, the user will exhaust its token resources and would need to purchase additional tokens to continue the attack.

7 Other Items

7.1 Roadmap

The current development has been ongoing since July 2017 on the design of the protocol, its nuances and security issues. The following roadmap in Fig. 14 gives an immediate view of the development leading up to the ICO, following which we will work on other aspects of the blockchain as detailed in Fig. 15.

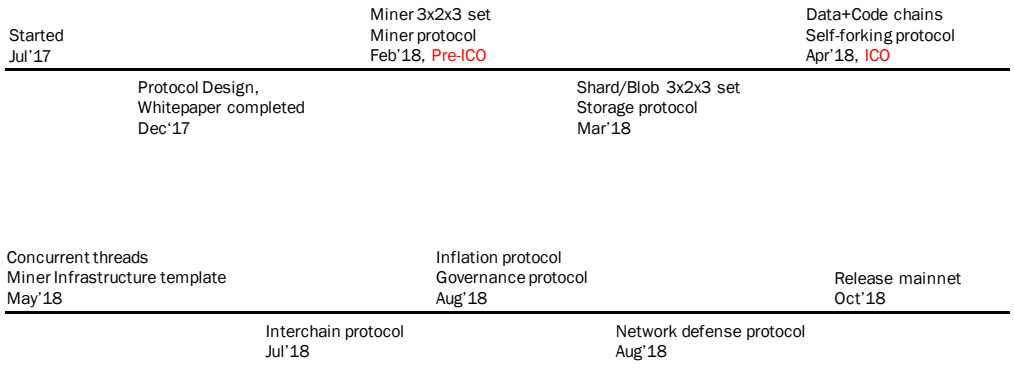

Fig 14. Development timeline leading up to the ICO and launch of mainnet.

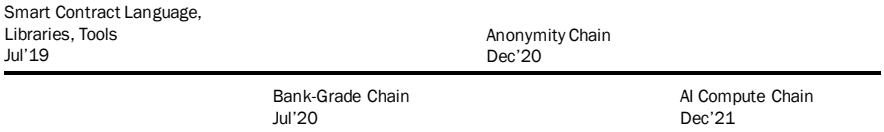

Fig 15. Development timeline after ICO

7.2 Token Distribution

The distribution of token is outlined in the pie chart of Fig. 16. There is a total of 400,000,000 (400 million) tokens minted. The distribution is as follows: 30% for the team, advisors, development partners, and seed token buyers, 10% for the private sale, 10% reserved for future offering, and 50% for miners, sharders, and blobbers. The team will have a linear vesting cycle of 4 years with no cliff, and will have their tokens locked for 180 days after the pre-sale event concludes. The token buyers participating in the pre-sale are subject to linear vesting over 180 days after the pre-sale event is closed, with a certain percentage released on Day 1. The reserve pool has 50% of its tokens locked for 2 years and the rest for 4 years.

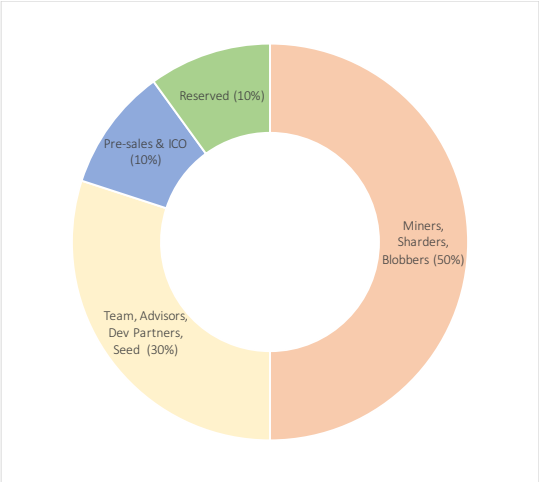

Fig. 16. 0chain token distribution.

7.3 Go To Market Strategy
We're partnering with DApps that need to off-load data for fast computation and storage activity in a decentralized manner. We're positioning ourselves as a decentralized cloud solution, rather than a competitive chain, and will partner with DApps on any existing chain and future ones to optimize their current solution with higher performance, decentralization, security, and cost. DApps need to hold as much as they would pay AWS over a certain period. There would be a suggested margin to ensure automated scalability if there is an intermittent demand when our on-demand protocol would hold more tokens, based on scaling requirement, and release them after a spiked event. If the token price drops dramatically, our protocol will guarantee the same level of performance and storage for a period of time, to make sure that DApps are protected during catastrophic events.

7.4 Utility Token
ZCHN tokens allow you to compute and store data on our blockchain, as well as create and run decentralized applications, and create new chains for different verticals. The network does not derive any fees or revenues from the blockchain.

7.5 Intrinsic Value Token
ZCHN token is considered an intrinsic value token as the value of token is proportional to the computation and storage the token holder is using. Unlike other tokens, which are primarily used as a store of value, our token encompasses both value and data. A DApp would not dispose of their tokens, otherwise their business will be affected. Similarly, an individual would think twice about deleting their images and videos stored on our network, in order to dispose their tokens.

7.6 Governance
The governance and development of the protocol will be determined by the team initially for up to 3-4 years, and later by the stakeholders based on a 2/3 majority voting. The key areas of governance are future development, inflation rate, holding rate, punishment rate, changes to the parameters of a particular chain, creation of a new chain, changes to the computing and storage environment.

7.7 Formal Proofs
Formal proofs of what we have stated in this document will be done in the coming months. While proofs are necessary and will be on one of our tracks of development, it will be done in parallel with code development as outlined in the roadmap.

Team
Saswata Basu, Project Lead
Saswata is a serial entrepreneur with 25 years of experience, and has worked in pioneering technologies such as Blockchain, AI, IoT, Cloud, CleanTech, and 4G at various industries including Intel, Nortel, Harris, and Aviat. A full stack developer, Saswata has written code for multiple applications and have open sourced some in GitHub. Saswata completed his MS and Ph.D. in 3 years at UCLA.
https://www.linkedin.com/in/saswatabasu/
https://github.com/guruhubb/

Thomas Austin, Technical Lead
Thomas is an assistant professor at San Jose State University, where he is an expert in information security and programming language paradigms. He earned his PhD in computer science from UC Santa Cruz. He has previously worked with Mozilla's research group, ESIEA Ouest's Cryptology and Operational Virology lab, and CloudFlare, Inc.
http://www.sjsu.edu/people/thomas.austin/publications/
https://www.linkedin.com/in/tom-austin-49195b1/

Atif Yaqub, Business Development Lead
Atif Yaqub, born in Manchester, UK, raised in London, is a globe trotting serial entrepreneur. During his BA (Hons) Arts study he embarked on his first SME, later selling to a competitor. After enjoying a successful retail business in the UK, he decided to increase his global connections. Continuously on the move, he forged high level business relationships in Europe, Middle East & Asia. Attracting investors with his multifaceted skill set, he then built an international trading company between London & Dubai dealing in luxury cars, heavy plant and tech products. Atif currently owns and partners with several SMEs in UK & UAE, ranging from fast food, general trading & management.
https://www.linkedin.com/in/atif-yaqub-454427a/

S. Kombai, Blockchain Engineer
A full stack developer, Kombai has 10 years of experience in architecture and code development in Solidity, Java, Node, Meteor, Swift, Android, and Angular at InSead, Ford, Cisco, and Merck. Kombai earned his BE in computer science from Anna University.
https://www.linkedin.com/in/kombai-s-2a920a157/

Kenny Wesela, Protocol Engineer
Kenny is a hacker and loves to break things and then figures out how to fix them. Ken has reverse engineered communication protocols, hacked a wifi based quadcopter, and conducted a man-in-the-middle attack. Ken graduated from San Jose State University with a BS in Computer Science, where he focused on Information Security & Artificial Intelligence.
https://www.linkedin.com/in/kenny-wesela-5718a036/

Neha Rajkumar, Protocol Engineer
Neha is a versatile software engineer and graduated from San Jose State University with a Masters in Computer Science. She specializes in information security, programming language paradigms and data analytics. She has previously worked with Symantec Corporation and Robert Bosch.
https://www.linkedin.com/in/neharajkumar

Nikhaar Shah, UI/UX Design
Nikhaar has been working as a graphic and visual designer for over 7 years and has won multiple online design contests. Nikhaar has a Bachelors in Graphics, Multimedia & Animation.
https://www.linkedin.com/in/nikhaarshah/

Advisors

Ken Huang, Blockchain Advisor
Ken is a well known blockchain expert, a former Chief Blockchain Scientist and VP at Huawei. Before joining Huawei, he worked at CGI Federal office in USA for 18 years, and served as Director of Cyber Security and Director of Cloud Security. He has consulted with the US Federal Government, financial institutions, utility companies, and provided expertise in Finance, Blockchain, and Cyber Security. He is a member of CISSP, ACM, and a Blockchain Expert committee member of the Chinese Electric Academy, and a visiting professor of Zhejiang Normal University. Ken has an MS in CS from Xi'an Jiaotong University in China, and an MBA from University of Lausanne in Switzerland.
https://www.linkedin.com/in/kenhuang8/

Vishwas Manral, Network Protocol Advisor
Vishwas is currently the Co-Founder & CEO of NanoSec and a Vice-Chair of the Cloud Security Alliance. He has been a Co-founder, CTO at Ionos Networks, and a Chief Technologist at HP. Vishwas has deep protocol design experience that traces back to his work in the IETF where he has been instrumental in design and progressing protocols like ADVPN, IPsec, MPLS, and 6LoWPAN.
https://www.linkedin.com/in/vishwasmanral/

Doug Park, Legal Advisor
Park is a corporate and securities lawyer and business strategist who is on the Super Lawyers list for Corporate in Northern California and advises various blockchain projects in the Bay Area. Park combines his strengths in business and legal to advise clients on complex financing and commercial transactions, corporate governance, securities law, and corporate policy. Previously he was the General Counsel of Tessellation Capital Management where he advised on investment fund, compliance, corporate governance, and investor relations. Other professional experiences include Director of Legal Policy and Outreach at the Sustainability Accounting Standards Board and securities litigation at Cotchett, Pitre & McCarthy. As an academic, Doug has taught Strategy and Organization, Entrepreneurship, M&A, and Organizational Theory to undergraduates, MBA students, and executives from around the world. He holds a PhD from the Stanford Business School, a J.D. from the University of Michigan Law School, and an A.B. magna cum laude with highest honors in Sociology from Harvard College.
https://www.linkedin.com/in/douglaspark/