# Internet of Services: The Next-generation, Secure, Highly Scalable Ecosystem for Online Services

The Internet of Services Foundation

December 31, 2017

## Abstract

Despite the recent hype in the cryptocurrency market, the underlying blockchain technology is still at an early stage and is far from mass adoption. One of the well-recognized issues with current blockchain technologies is scalability. Without the capability of processing large volumes of transactions swiftly, heavy usage services like Facebook, Amazon, and digital asset exchanges are nearly impossible to deploy onto the blockchain.

In this paper, we propose the Internet of Services ("**IOS**"), an innovative and secure blockchain paradigm designed to provide horizontal scalability and high transaction throughput. By implementing our novel sharding architecture and consensus mechanism, the IOS system is able to process up to 100,000 secure transactions per second.

This work makes the following contributions. It introduces:

1. *Efficient Distributed Sharding* (EDS) - an innovative sharding scheme that makes shards sufficiently large and strongly bias-resistant via a combination of a client-server randomness scavenging mechanism and leader election via cryptographic sortition.
2. *TransEpoch* - a secure validators-to-shards assignment during epoch transitions while maintaining transaction operability.
3. *Atomix* - a novel two-step inter-shard atomic commit protocol that guarantees transaction atomicity in Byzantine setting.
4. *Proof-of-Believability* (PoB) - a groundbreaking Byzantine consensus protocol with a Believable-First approach that guarantees safety and liveness of the system while largely maximizes the transaction throughput by size-one-shard.
5. *Micro State Block* (MSB) - a novel mechanism to minimize the storage and bootstrapping costs for validators.

**Note**: the IOS is a work in progress. Active research is under way, and new versions of this paper will be updated at http://iost.io. For comments and suggestions, contact us at team@iost.io.

## Contents

# 1 Background



Excessive commission fees, privacy violations, frauds and censorship are common issues encountered during daily interactions with centralized online service providers. Given these well-recognized problems with centralization, a wide range of blockchain technologies attempting to resolve these issues have been developed since the launch of Bitcoin [15] in 2008. Specialized projects like SteemIt [29], Bitshares [30], and Syscoin [21], as well as more versatile projects like Ethereum [28] and EOS [11], are some of many examples. However, most of those attempts are either too specialized in certain applications, or burdened by low transaction throughput. Due to these limitations in flexibility and transaction throughput, it is impossible for developers and enterprises to bring heavy services like Facebook or Amazon onto the blockchain - not to mention something more complicated like digital asset exchanges.

The heart of the scalability issue lies in the fundamental design of these existing blockchain technologies -- their consensus protocols and blockchain architectures. Most of the existing blockchain technologies face two major challenges in their way of scaling up: a) every full node must store the entire ledger in order to participate; b) every participating node in the network is obligated to handle every transaction. Since all the participating nodes are essentially conducting the same work, the number of transactions the system can process will not exceed that of a single node. Moreover, the growing size of the blockchain increases the requirements and costs of storage space, bandwidth, and computational resources for a node to fully participate in the network. The increasing mining cost will inevitably make the participation in the network become a privilege for the few, leading us straight back to the problem of centralization.

The IOS is designed to fill the void. In our vision, the IOS is a next-generation blockchain technology that provides the network infrastructure to support a service-oriented ecosystem. The IOS platform not only provides its users a completely decentralized way to exchange online services and digital goods, but also enables developers to deploy large scale dApps with the ability to support massive number of users. With a series of groundbreaking innovations, such as Efficient Distributed Sharding (“EDS”) and Believable-First consensus approach, we are able to increase the system’s throughput enormously while guaranteeing security.

We developed EDS based on the sharding technique. It is widely adopted in distributed systems and databases to enable parallel transaction processing. Inspired by the classic





“Divide and Conquer” principle in computer science, sharding is a technique that partitions the entire IOS network into certain numbers of subspaces called *shards*. We can consider each shard as a miniature network that runs its own consensus protocol in parallel. Instead of having the entire network validating the same set of transactions, subsets of transactions

of having the entire network validating the same set of transactions, subsets of transactions can be handled by various consensus groups simultaneously. Therefore, the throughput of the system can be significantly enhanced even if the size of the network and number of transactions grow rapidly. Moreover, in order to ensure the network is divided homogeneously, we have developed a Bias-Resistant Distributed Randomness protocol in order to introduce unbiased and transparent randomness into the sharding process.

With the EDS, the IOS is also packed in a very powerful arsenal of technologies that empowers deployment of large scale dApps with high-performance and flexibility. It allows developers to build a wide range of products: from counterparts of traditional monopolistic online service providers to brand new business models that would have been considered impossible previously. Admittedly, running such services could be quite expensive when the size of the IOS network is relatively small. However, with increasing number of nodes and resources in the network, the cost of running such large scale dApps might be greatly reduced. Additionally, there are many benefits including: avoiding cyber attacks, high-level of data security, and the immutable property.

During the development of the IOS blockchain, we thoroughly examined all currently available solutions in order to learn from previous attempts.



# 2 Related Work

## 2.1 State Machine Replication

In a nutshell, blockchain technologies are state machine replication protocols. Every state

machine replication protocol has to satisfy two important properties:

1. *Safety*, i.e., all servers in the network have the same record of transactions;
2. *Liveness*, i.e., transactions of clients are submitted and documented into the log quickly.

There are two fundamentally different ways to achieve state machine replication: classical-style consensus and blockchain-style consensus [18]. Classical-style consensus generally applies Paxos-like algorithms and is used in the permissioned setting where there is a priori knowledge of the consensus nodes known by the system. An example of this application would be the servers at software companies like Amazon, where their servers collectively use a classical way to replicate and store information, and the classical algorithm establishes the fundamentals to form a consensus of the ordering of their data.

### 2.2 Bitcoin and Proof-of-Work

Satoshi Nakamoto was the first to introduce Bitcoin as a solution to establish consensus in the permissionless setting , e.g. any node can freely join and leave the network without a priori knowledge of the consensus nodes. The network underlying Bitcoin, *blockchain* improves the scale of distributed systems without human involvement by providing economic incentives to the servers, dubbed miners in Bitcoin's settings. Miners in the Bitcoin network form consensus by calculating partial hash collisions with a certain difficulty level. The chain with the greatest cumulative difficulty would be acknowledged by other nodes as the consensus result. This solution is named Proof-of-Work (PoW), which in essence is to have all the nodes in the network contribute their computing power as a way to earn incentives and thus determine the ordering of transactions for the whole system. A benefit of PoW is its ability to defend against Sybil attack in a permissionless setting [15]. Despite its advancements in scale and security, Bitcoin has a few major drawbacks: (1) unlike other modern cryptocurrencies, it takes more than an hour to confirm a transaction according its configuration; (2) it is difficult to develop various applications upon Bitcoin network; (3) the consensus mechanism wastes too much energy, i.e., it costs more than two million dollars per day in electricity. More importantly, earlier works show that Bitcoin-style blockchain must have a sufficiently large interval to retain security [16] [17].





Therefore, Bitcoin would not be a good replacement for the current centralized system to support day-to-day applications and large transaction volume.

### 2.3 Proof-of-Stake

The concept of Proof-of-Stake was first discussed on an online blockchain forum [31] and was adopted by a few cryptocurrencies like PPcoin [22], PeerCoins[23] and Nxt [5]. The idea of PoS is essentially one vote per unit of stake, such that for each validator, owning more stake will have higher voting power. Therefore, validators have no economic

incentive to harm the whole blockchain network. For attackers, the cost of attack is huge because they have to own the majority of the stakes. In the early development, Proof-of-Stake consensus mechanism is known for being vulnerable to "nothing-at-stake" attacks, where servers are able to vote on multiple blocks at the same time with no incentive to converge, damaging the security of the blockchain. Later work solves the problem using *slasher* [3], which enforces a punishment for violating nodes. Many other work are also described as the ad hoc application of Proof-of-Stake [1–3,26][12][10]. Although PoS fulfills the *liveness* of the replicated state machine protocol, it still faces challenges like centralization and security problems. For instance, validators possessing more tokens will be more likely to generate new blocks and get richer, leading to a potential centralization problem. Furthermore, previous work shows that Proof-of-Stake protocol can only be a provably secure and robustly configured consensus protocol if its token is not being exchanged too frequently [19] , which potentially implies that there is a ceiling throughput for Proof-of-Stake in order to preserve security.





## 3 Blockchain Architecture

The infrastructure of IOSChain is similar to existing well-known blockchains like Bitcoin and Ethereum, where nodes disseminate data through gossip protocol. The system split its data and state into shards. Each node is responsible for one shard of the system. Unspent transactions (UTXOs) are stored in the memory of the nodes in the corresponding shards. This raises several new challenges.

- How to divide the system into shards.
- How to reach consensus in each shard

- How to reach consensus in each shard.
- How to perform inter-shard transactions.

In order to solve the above challenges in a fair and secure manner, we have to perform many random operations, for example, assigning nodes into shards, electing leaders in each shard. As a result, we have to first design an unforgeable and unbiased (uniformly random) distributed random number generation protocol. With the random number generation protocol, the above challenges can be addressed one by one.

In the rest of this paper, we present the techniques and methods used to address these challenges.

- In Section 4, we present Distributed Randomness Protocol (DRP), which is unforgeable and unbiased when the ratio of malicious nodes are below some certain predefined threshold. The random numbers generated by DRP is used to divide the system into shards, assign nodes to different shards, and elect leaders in each shard.
- In Section 5, we present *EDS* - a novel scheme to form shards (subsets of validators to record state and process transactions) that are both sufficiently large and strongly bias-resistant using a combination of DRP and a VRF-based leader election via cryptographic sortition.
- In Section 6, we present *TransEpoch* - a secure validators-to-shards assignment protocol during epoch transitions while maintaining system operability.
- In Section 7, we present *Atomix* - a novel two-step inter-shard atomic commit protocol that guarantees transaction atomicity in Byzantine setting.
- In Section 8, we present *Proof-of-Believability* - a novel Byzantine consensus protocol with a Believable-First approach that guarantees safety and *liveness* of the system while largely maximizing the transaction throughput by size-one-shard.





- In Section 9, we present Micro State Blocks (MSB) - a novel mechanism to minimize the storage and bootstrapping costs for validators.

# 4 Distributed Randomness Protocol

Traditional approach to generate randomness like Proof-of-Work based mechanism [13] or a trusted beacon [6] have computational wastes and centralization concerns. It is desirable to use cryptographic tools to generate distributed random numbers, which not only saves resources but also is provably secure.

There are multiple algorithms to generate distributed random numbers for the purpose of node-shard assignment and leader election in IOSChain. Here we present the one that, to our knowledge, the best fits the requirement in the IOSChain scenario. In IOSChain, the distributed random number generator has the following requirements.

(1) It has to be resistant to dishonest participants (including clients and servers) with in a certain ratio. To be detailed, the system is able to make progress when the ratio of dishonest participants are below the threshold, and nothing bad happens when

of dishonest participants are below the threshold, and nothing bad happens when making progress.

(2) The final random number must be unforgeable and unbiased (uniformly random), except negligible probability.

(3) Dishonest participant is not able to try multiple times to generate the random number that favors the participant, even multiple dishonest participants collude.

(4) Third parties are able to verify the output is generated by faithfully running the protocol (i.e., verify that it satisfies all the above requirements).

In order to achieve these requirements, we propose to use a client-server protocol, called Distributed Randomness Protocol (DRP) [24], where a client communicates with a set of servers to generate an unforgeable, uniformly random value through non-interactive zero-knowledge proof (NIZK) and publicly verifiable secret sharing (PVSS). In a certain protocol run, before the protocol finishes and the final random output is revealed, no entity in the protocol is able to learn any information about the final output, which makes sure a dishonest client is not able to try multiple runs to generate the random number that favors the dishonest client.

The protocol consists of two phases - randomness generation and randomness verification. It works as follows. Initially, the client starts a protocol run by broadcasting to all the servers a message including a randomly generated balanced grouping. In the first phase, each server generates a random input value and creates shares only for other members of the same group using PVSS. Upon receiving encrypted shares with the NIZK [25] proofs





from all the servers (or timeout), the client chooses a subset of server inputs from each group. This allows the client fix each group's secret and the output of the protocol. In the second phase, the servers decrypt and send their shares to the client as soon as the client receives a sign-off on input values in a global run of collective signature (CoSi) [25]. Then the client combines the recovered group secrets to reveal the final random output.

# 5 Efficient Distributed Sharding

With the distributed randomness protocol (DRP) presented above, it is not difficult to implement efficient distributed sharding. However, the protocol only works well without malicious or failure nodes, since it is performed by validators collectively. Therefore, we have to design backup protocols for scenarios with malicious or failure nodes. To conquer this problem, we propose a solution that uses Algorand [9] and Omniledger [8] to elect a leader.

## 5.1 Algorithm - Leader Election with Back-up Protocol

***Algorithm 1:*** **Leader Election With Back-up Protocol**

Inputs:

1) $v$ is a view counter
2) $i$ is a validator
3) $sk_i$ is the private key for $i$
4) $e$ is the current epoch
5) $\Delta$ is the synchrony bound

Output: a validator $i$ which has the minimum-value valid lottery to run DRP

1) For each $e$, each $i$ computes a lottery $lottery_{i,e,v}$ using Verifiable Random Function with its view $v$ and the node's private key $sk_i$.
2) Then for a time bound $\Delta$, the validators gossip these lotteries with each other.Each validator collects the top 3 minimum-value lottery in the gossip process.
3) After the time bound $\Delta$, the validators fix the minimum-value valid lottery they have seen so far.
4) The validator corresponding to the minimum-value valid lottery is elected as the leader, while the other two validators corresponding to the second and third minimum-value valid lottery are used as the pool for back-up leaders.
5) If the elected validator successfully runs the DRP, it broadcasts the output $rnd_e$ to all other validators with its correctness proof.
6) Each $i$ can use $rnd_e$ to compute a permutation and divide the result into $m$ buckets with same size, thus the mapping from nodes to shards is determined.
7) After the time bound $\Delta$, if the elected validator fails to start DRP, validators mark the current run as failed and exclude this leader in the rest of the epoch $e$. In this case,the back-up leader will be used to run DRP. If the two backup leaders fail





continuously, the lottery will roll back to step 1 and the whole protocol will be rerun.

### 5.2 Analysis

The leader election mechanism provides required properties which is the same as those described in Section 4. Each validator can produce only a single valid lottery per view $v$ in a epoch $e$. The DRP design provides scalability. Since the private key $sk_i$ is kept secret, the output of VRF is unpredictable. Given our synchrony time bound $\Delta$, the lottery will be seen by all other validators within $\Delta$. If malicious nodes win the lottery, it cannot perform arbitrary behaviors---either choose to cooperate and run the DRP protocol, or decide to fail the epoch. If any of the malicious/abnormal cases happens, the malicious nodes would be excluded from participating in the rest of the epoch.

## 6 Operability During Epoch Transitions

There are many shard configuration schemes, such as static configuration and some different rolling schemes. IOSChain uses a dynamical rolling scheme - it swaps out and in validators in batches for each epoch $e$. This configuration will give IOSChain an idle period that only after enough validators have bootstrapped appropriately, the network can begin processing transactions. Many designs of the blockchain did not take the issue that how to make sure the system is operational during this period into consideration.

A key factor of the issue during the transition is the batch size, which is is highly relevant to the safety of the system. When the swap batch size grows, the risk increases as the number of remaining honest validators will not be sufficient to reach consensus. Another disadvantage of growing swap batch size is that the downloading and bootstrapping information will cause network stress increases. Given our threat model that there are at most $1/3$ malicious nodes, the maximum size of the swap batch should be less than $1/3$ nodes.

To maintain full operability during transition/idle phases, we use the method of selecting a subset of the validators to be swapped out and replaced with new members [8,24]. This is based on Omniledger's approach [8]. It enables the remaining validators to continue offering services while the newly joined nodes are downloading history data and bootstrapping. We present the node-to-shard transition assignment protocol - TransEpoch as follows.

### 6.1 Node-to-shard Transition Assignment Algorithm - TransEpoch

***Algorithm 4:*** **TransEpoch**

Inputs:

1) $n$ is the total number of nodes
2) $m$ is the size of each shard
3) $k$ is the the swap-out batch size, i.e., the number of validators that will be swapped out at a given time in a given epoch.

Outputs:

1) Set $k = log\frac{n}{m}$
2) For each shard $j$,





a) Generate two seeds $s_{j,rnd_e}$ and $s_{0,rnd_e}$ using the generated random output from DRP.
b) Use $s_{j,rnd_e}$ and $s_{0,rnd_e}$ to get the permutation $\pi_{j,e}$ and $\pi_{0,e}$ and divide nodes into buckets of size $k$.In this way, the node to swap-batch mapping is determined.
c) $\pi_{j,e}$ is used for current nodes and $\pi_{0,e}$ is used for the newly joined nodes.
d) Each batch waits $\Delta$ and then starts the swap.

### 6.2 Analysis

In the algorithm presented above, we ensured safety of Byzantine Fault Tolerance (BFT) consensus in each shard transition. The reasons are in two folds. Firstly, we made sure the size of the group. There are least $2/3$ shard size validators running consensus. Secondly, safety against adversary is also guaranteed as per epoch randomness is used to generate the permutation of validators batches.

## 7 Inter-Shard Transactions

The mechanism that supports inter-shard transactions is critical in the system, since transactions are likely to happen cross shards. We introduce a Byzantine Shard Atomic Commit (Atomix) protocol to ensures the atomicity cross shards. This protocol prevents double spending and keeps the consistency of transactions. Our design is a variant of the Omniledger algorithm. [8]

We first present Atomix in the UTXO state model. Previous work has proved that the Atomix can ensure that the smart contract is also supported by our inter-shard transaction mechanism [27] , if the UTXO model is supported.

In a nutshell, when a cross-shard transaction from node ***a*** at shard ***A*** to node ***b*** at shard ***B*** happens, the algorithm does the following:

1) Create the TX within the shard ***A*** and let all nodes validate the transaction.
2) If the TX is approved by all nodes in shard ***A***, the transaction is logged in ***A***'s blockchain. At the same time, the client will gossip a proof-of-acceptance to endorse the transaction,lock the fund of ***a*** in to a UTXO, and send it to ***B***.
    a) If the TX is rejected by nodes in ***A***, the fund gets returned to ***a***.
3) ***A***'s blockchain commits the TX to the ***B***'s blockchain and have nodes in the receiver's shard validating the TX.
    a) If the TX is rejected by nodes in ***B***, the fund gets returned to ***a***.
4) If the TX gets approved by all nodes in the ***B***'s blockchain,

the fund is released to **b**.

a) If the TX is rejected by all nodes, the fund is returned to **a**.





# 8 Consensus Mechanism

## 8.1 Tokens and Motivations

In the IOS system, *IOS Token*, like tokens in other blockchain systems, serves as the medium of exchange for all transactions and commission fees. More importantly, IOS also plays a important role in calculating a user's believability score. All IOS tokens will be generated in the Genesis Block. In the IOS ecosystem, IOS tokens can be used for:

- **Payment**: Payments for services and goods provided by merchants or other community members.
- **Commission**: Payment to validators as compensation for running smart contracts, processing messages and transactions, and using resources shared by the general ecosystem including but not limited to storage space, computing power, etc. The commission fee incentivizes the validators and prevents malicious users from damaging the interests of the community through excessive deployment of smart contracts.
- **Believability**: IOS tokens will be used to calculate users' believability scores (will be explained in the following section).

In addition, as a member of the IOS ecosystem, each user can acquire IOS tokens through validating transactions and contributing resources (e.g., running smart contracts, providing storage space, etc.).

As mentioned in previous sections, A major challenge faced by traditional Proof-of-Stake consensus mechanism is the tendency towards centralization. In order to mitigate this risk, we introduce Servi as both a measurement of users' contribution to the community and a way to encourage members to contribute to the continued development of the IOSChain. It has the following attributes:

- **Non-tradable**: Since Servi is not designed as a medium of exchange, Servi can not be traded or exchanged in any way.
- **Self-destructive**: After validating a block, the system will automatically clear the

Servi balance owned by the validator. In this way, nodes with high believability scores can take turns in validating blocks, to ensure a fair block generation process.

- **Self-issuance**: Servi will be generated and deposited to user accounts automatically after certain contributions, such as providing community services, evaluating services provided by another entities, and/or making other special contributions.





### 8.2 Proof-of-Believability

Traditional blockchain systems have an inherent trade-off between safety and throughput, depending on shard size. A system with a large number of small shards delivers better performance but provides less resiliency against bad actors, and vice versa. In order to break the trade-off in a way that keeps safety and increases throughput, we propose an innovative Proof-of-Believability (PoB) consensus protocol for IOSChain. PoB guarantees that the nodes are with negligible probability to misbehave, while significantly increasing the transaction throughput by size-one-shard.

The Proof-of-Believability consensus protocol uses an intra-shard Believable-First approach. The protocol divides all validators into two groups, a believable league and a normal league. Believable validators process transactions quickly in the first phase. Afterwards, normal validators sample and verify the transactions in the second phase to provide finality and ensure verifiability. The chance of a node being elected into the believable league is determined by believability score which is calculated by multiple factors (e.g., token balance, contributions to the community, reviews, etc). One with higher believability score is more likely to be elected into the believable league. Believable validators follow the procedures to decide the set of committed transactions and their order, as well as process them in order. Believable validators also form smaller groups - one validator per group. Transactions will be randomly distributed among these believable validators. Consequently, they produce smaller blocks with extremely low latency.

However, it may introduce security problem as only one node is performing the verification. As a result, some corrupted transactions might be committed by misbehaved validators. In order to solve this security problem, we specify a sampling probability $p$ that normal validators will sample transactions and detect inconsistencies. If a validator is detected as misbehaviour, it will lose all the tokens and reputation in the system while the defrauded users will be compensated for any loss. The believable-first approach makes processing transactions extremely fast as only a single (believable) validator is doing the verification and it is unlikely to misbehave.

In the IOS system, the sharding policy file specifies the sizes of the believable and normal league, respectively, and the sampling probability $p$. Upon the inception of an epoch, all validators will be assigned to shards using the distributed randomness generation

protocol. Their states will be bootstrapped from the corresponding shard's last Micro State





Blocks (MSB). Depending on the believability score , validators will be assigned to either believable group (small) or the normal group (large) within a shard.

In the first phase, transactions that are processed by the believable league produce optimistically validated blocks. These blocks serve as input for sampling re-validation by the normal league who runs concurrently. The normal league also combines inputs from multiple optimistic processing groups. This could maximize the throughput of the system. If transactions are validated successfully, they will be included in a finalized block, added to the shard's blockchain, and finally included in the MSB. However, when the normal league detects any inconsistency, the corresponding validated transaction would be excluded from the blockchain and the validator who signed the invalid block would be detected and held accountable. We designed the punishment scheme to be powerfully harsh so that the validator has no incentive to misbehave under any circumstances. If a validator is detected as misbehaving, that validator will lose all tokens and reputation in the system and all its previously validated transactions will be re-checked. Given the minimal incentive to be at fault and the quantifiable confidence in the security of validation, clients can achieve real-time processing speed with assurance.

The normal league runs the Byzantine consensus scheme based on ByzCoin [7], because it scales efficiently to thousands of consensus group members. ByzCoin uses collective signature (CoSi) [25], a scalable cryptographic primitive that uses multi-signatures [20], to make traditional consensus algorithms such as PBFT [4] scale. ByzCoin distributes blocks using multicast trees for performance, and falls back to a star topology for fault tolerance. It ensures that all the honest members of a shard agree on a specific common sequence of actions, despite some malicious nodes are in the shard, while guaranteeing safety and liveness.

To ensure robustness, we use a fall-back scheme in Believable-first protocol. When a shard doesn't have enough believable validators to form the league, due to either temporary downtime or being in the bootstrapping phase of the ecosystem, two-league committees would fall back to one-league. All transactions are directly processed by the normal league following the PBFT consensus protocol.

# 9 Blockchain Storage Pruning

Another issue current blockchains are facing is the rapid expanding size of the blockchain storage [8], which puts on new validators heavy workload for bootstrapping. Blockchains follow the same pattern to store historical data from the beginning. However, this is a crucial concern for the high-throughput blockchain systems as the the storage will explode. To minimize the storage and bootstrapping costs for validators, we use a blockchain storage pruning approach to summarize the full state of a shard's blockchain. We use Micro State Blocks (MSB), which is based on the State Block [8]. We present the MSB generation protocol below.

## 9.1 Algorithm - MSB Generation Protocol

***Algorithm 6: MSB Generation Protocol***

Inputs:

1) $e$ is the current epoch
2) $j$ is the current shard

Output: the micro state block $msb_{j,e}$ for $j$ in $e$

1) When the epoch $e$ ends, the shard leader stores all transactions of $e$ in a Merkle tree [14].
2) The shard leader hashes the Merkle tree's root, denoted by $h$, and puts $h$ in $Header(msb_{j,e})$.
3) Validators run consensus on the $Header(msb_{j,e})$, while there is not any regular blocks pending.
4) If the correctness of $Header(msb_{j,e})$ is verified, the shard leader stores the approved header in the shard's blockchain.
5) At the end of epoch $e+1$, all the nodes drop the body of $msb_{j,e-1}$ and keep the regular blocks of $e$.

## 9.2 Analysis

Transactions are checked by references to past blockchains. Since each shard in IOSChain stores only the past MSB headers and blockchain state is distributed across multiple shards, a client cannot prove the existence of a past transaction by providing a check in the block. We mitigate this issue by moving the storage responsibility from past blockchains to

the client. Since latest epoch's blocks is retained, clients can ask the validators of the shard to create existence proof for transactions validated in epoch $e$ during the next epoch.

Validators are essentially creating a higher level chain for MSBs, that provides skips from an epoch's MSB to another. This MSB Chain keeps the latest MSB with its body and all the previous MSB headers. This is important as clients that want to verify a past transaction need to have a reference point. We remark that MSBs may contain several multi-hop backpointers to headers of regular blocks in order to reduce the size of their proofs.

With MSB, bootstrapping new validators and syncing crashed validators up-to-date become efficient, as validators start from the last valid MSB and replay only the last part of the blockchain, instead of replaying the full history from the first block or from the time they crashed. If Bitcoin is deployed on IOSChain, currently the bandwidth bootstrapping costs would be two orders of magnitude less. This is critical when new shards come in in IOSChain. Due to the random shard assignment mechanism, validators changes shards periodically and need to be updated frequently, which benefits a lot from the blockchain storage pruning technique.

## REFERENCES

[1] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. 2016. Cryptocurrencies Without Proof of Work. In *Lecture Notes in Computer Science*. 142–157.

[2] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. 2014. Proof of Activity. *ACM SIGMETRICS Performance Evaluation Review* 42, 3 (2014), 34–37.

[3] Vitalik Buterin. 2014. Slasher: a punitive proof of stake algorithm. Retrieved January 9, 2018 from https://blog.ethereum.org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm/

[4] Miguel Oom Temudo de Castro. 2000. *Practical Byzantine Fault Tolerance*.

[5] Nxt Community. Nxt Whitepaper. Retrieved January 9, 2018 from https://bravenewcoin.com/assets/Whitepapers/NxtWhitepaper-v122-rev4.pdf

[6] George Danezis and Sarah Meiklejohn. 2016. Centrally Banked Cryptocurrencies. In *Proceedings 2016 Network and Distributed System Security Symposium*. DOI:https://doi.org/10.14722/ndss.2016.23187

[7] E. Kokoris-Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford. 2016. Enhancing Bitcoin Security and Performance with Strong Consistency via Collective Signing. In *25th USENIX Conference on Security Symposium*.

[8] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser†, Nicolas Gailly, Ewa Syta, Bryan Ford. 2017. OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding.

[9] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. 2017. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles - SOSP '17*. DOI:https://doi.org/10.1145/3132747.3132757

[10] G Maxwell And. 2015. On Stake and Consensus. Retrieved January 9, 2018 from https://download.wpsoftware.net/bitcoin/pos.pdf

[11] Ian Grigg. EOS - An Introduction. *eos.io*. Retrieved from https://eos.io/documents/EOS_An_Introduction.pdf

[12] J. Kwon. 2014. Tendermint: Consensus without mining. Retrieved January 9, 2018 from http://tendermint.com/docs/tendermint.pdf

[13] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. 2016. A Secure Sharding Protocol For Open Blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security - CCS'16*. DOI:https://doi.org/10.1145/2976749.2978389

[14] Ralph C. Merkle. A Certified Digital Signature. In *Lecture Notes in Computer Science*. 218–238.

[15] Satoshi Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. *bitcoin.org*. Retrieved from https://bitcoin.org/bitcoin.pdf

[16] Rafael Pass, Lior Seeman, and Abhi Shelat. 2017. Analysis of the Blockchain Protocol in Asynchronous Networks. In *Lecture Notes in Computer Science*. 643–673.

[17] Rafael Pass and Elaine Shi. 2017. The Sleepy Model of Consensus. In *Lecture Notes in Computer Science*. 380–409.

[18] Phil Daian and Rafael Pass and Elaine Shi. 2016. Snow White: Provably Secure Proofs of Stake. (2016).
[19] Phil Daian Rafael Pass. Snow White: Robustly Reconfigurable Consensus and Applications to Provably Secure Proofs of Stake.
[20] C. P. Schnorr. 1991. Efficient signature generation by smart cards. *J. Cryptology* 4, 3 (1991). DOI:https://doi.org/10.1007/bf00196725
[21] Jagdeep Sidhu. 2017. Syscoin: A Peer-to-Peer Electronic Cash System with Blockchain-Based Services for E-Business. In *2017 26th International Conference on Computer Communication and Networks (ICCCN)*. DOI:https://doi.org/10.1109/icccn.2017.8038518
[22] Sunny King And. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. Retrieved 2012 from https://peercoin.net/assets/paper/peercoin-paper.pdf
[23] Scott Nadal Sunny King. 2012. Peercoin. Retrieved January 9, 2018 from https://peercoin.net/assets/paper/peercoin-paper.pdf
[24] Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J. Fischer, and Bryan Ford. 2017. Scalable Bias-Resistant Distributed Randomness. In *2017 IEEE Symposium on Security and Privacy (SP)*. DOI:https://doi.org/10.1109/sp.2017.45
[25] Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ismail Khoffi, and Bryan Ford. 2016. Keeping Authorities "Honest or Bust" with Decentralized Witness Cosigning. In *2016 IEEE Symposium on Security and Privacy (SP)*. DOI:https://doi.org/10.1109/sp.2016.38
[26] V Buterin And. 2015. Casper. Retrieved January 9, 2018 from https://blog.ethereum.org/2015/08/01/ introducing-casper-friendly-ghost/
[27] G. Wood. 2014. Ethereum: A Secure Decentralised Generalised Transaction Ledger. Ethereum Project Yellow Paper. (2014).
[28] Gavin Wood. 2018. ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER. *ethereum.github.io/yellowpaper*. Retrieved from https://ethereum.github.io/yellowpaper/paper.pdf
[29] 2017. Steem: An incentivized, blockchain-based, public content platform. *steem.io*. Retrieved from https://steem.io/SteemWhitePaper.pdf
[30] Whitepapers. *bitshares.org*. Retrieved from http://docs.bitshares.org/bitshares/papers/
[31] Proof of stake instead of proof of work. Retrieved January 9, 2018 from https://bitcointalk.org/index.php?topic=27787.0