This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966464, IEEE Access

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.

Segment blockchain: A size reduced storage mechanism for blockchain

YIBIN XU[1](Student Member, IEEE), YANGYU HUANG[2]
[1]School of Computer Science and Informatics, Cardiff University Cardiff CF10 3AT, UK
[2]School of Electronic Engineering and Automation, Guilin University of Electronic and Technology, Guilin 541004, China

Corresponding author: Yibin Xu (e-mail: work@xuyibin.top).

ABSTRACT The exponential growth of the blockchain size has become a major contributing factor that hinders the decentralisation of blockchain and its potential implementations in data-heavy applications. In this paper, we propose segment blockchain, an approach that segmentises blockchain and enables nodes to only store a copy of one blockchain segment. We use *PoW* as a membership threshold to limit the number of nodes taken by an Adversary—the Adversary can only gain at most $n/2$ of nodes in a network of n nodes when it has 50% of the calculation power in the system (the Nakamoto blockchain security threshold). A segment blockchain system fails when an Adversary stores all copies of a segment, because the Adversary can then leave the system, causing a permanent loss of the segment. We theoretically prove that segment blockchain can sustain a $(AD/n)^m$ failure probability when the Adversary has no more than AD number of nodes and every segment is stored by m number of nodes. The storage requirement is mostly shrunken compared to the traditional design and therefore making the blockchain more suitable for data-heavy applications.

INDEX TERMS Distributed processing, Edge computing, Content distribution networks, Distributed management, Blockchain, Blockchain Storage

I. INTRODUCTION

In the anonymous and autonomous society like a blockchain system, every record should be re-derivable. This feature forms the essential trust of the blockchain, and secures the blockchain; thus, keeping every history transaction is critical. However, the size of the Nakamoto blockchain (Bitcoin [1]) has grown from the ground to over 226 $Gbytes$ in the past ten years from January 2009 to June 2019; the size doubled since the February 2017 (at slightly over 100 $Gbytes$) [2]. If the exponential growth to be continued, we are expecting 1.5 to 1.6 $TBytes$ in Jan 2021, 3 to 3.2 $TBytes$ in Jan 2022. Parallel to the growth of blockchain size, the storage cost of being a full node (the node which stores all the blocks of the mainchain) in the Bitcoin network is also grown in exponential.

It holds the promise that, through the usage of blockchain, complex and data-demanding jobs can be distributed through predefined protocols (usually through smart contracts [3]–[5]) to the anonymous nodes throughout the network, and the majority consensus can secure the job results. Ideally, an alternative-finance system can be built upon—the job publisher pays the system to do tasks, while the anonymous nodes in the system get paid by generating the commonly recognised task results [3]. It is guaranteed by the decentralisation and anonymous nature of blockchain that as long as the security threshold—the Adversary not having more than half of the participated calculation power is sustained, the results recognised by the majority are the correct results [1]. However, the oversize problem increases the bar of storage requirement for participants, making the system hard to process data-heavy applications like training Artificial Intelligence video recognition models [6], [7] distributedly and decentralisedly if the system remains universally joinable. Ordinary devices simply do not have enough space to store the data, and the system becomes increasingly centralised if the system process those applications. A transaction in Bitcoin and other Distributed Ledgers [3], [8] sized only around several hundreds of bytes. Even with such little usage of data, disadvantaged nodes are gradually leaving the mining game of most Distributed Ledger systems. More and more devices are acting in lightweight mode [9], [10] or join in mining pools [11], [12] for the reason of both lacking calculation

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966464, IEEE Access

advantage and the space for storing the blockchain.

Many flavours of approaches like weighted models [9], [10], [13], off-chain [14], [15], blockchain sharding [16]–[18] are proposed in recent researches to improve the performance of the blockchain. Many approaches attempt to ease the burden of individual nodes and solve the dilemma among having the ability to process everything, maintaining the decentralised system and increasing the performance.

For weighted models, the winning chance in the mining game and the duty of a node are different by weights. The lightweight node system [9], [10] is an example of the weighted model. A lightweight node does not store any block but is the client of some full nodes. They require relevant transactions from the full node to verify a new transaction using Simple Payment Verification (SPV) inquires [19]. A lightweight node only takes up to 4.2M Bytes per year, regardless of the total size of blockchain [19], but it cannot verify the new blocks and can be misled by full nodes. In Delegated Proof of Stake (DPoS) [8], people elect a fixed number of representatives and contribute their stakes to these representatives; these representatives then compete in the game of PoS [20]. DPoS has excellent performance because the representative nodes usually have a superpower regarding calculation ability, storage, and network bandwidth. These models are now commonly used in many blockchain-powered *IoT* systems [21], [22], where lightweight nodes are at the edge, or the nodes contribute their stakes to DPoS to function the system. Because the use of authoritarian/superior nodes, the systems are potential-centralised and the system security highly depends on these representatives.

For off-chain approaches, the relevant persons publish a co-signed contract at the beginning and the end of the relationship. Then they do the trading securely through off-chain channels [23], [24] without publishing transactions to the blockchain. They only publish the transactions to the blockchain when one violates the off-chain transactions. They need to monitor the blockchain to detect any violations; thus, it is not desirable for users who may go offline. And, there are few usages of off-chain approaches in non-financial related applications. Instead of broadcasting the task and the task result to the network, the entities who use off-chain methods must communicate in private, and this also compromises the anonymous nature of the blockchain.

For blockchain sharding approaches, they distribute nodes into different shards and divide the storage as well as the jobs to different shards which runs in parallel so that generally the work demand for individual nodes are not increased with the increase of transaction per second globally. Blockchain sharding is designed for applications that require high concurrency and high transaction per second. The design of blockchain sharding approaches mainly focuses on lowering the chance for the Adversary to occupy the majority spots in a shard when the Adversary has taken a relatively large population of nodes but has not taken the majority nodes globally. It is possible for the Adversary not having a security threshold number of nodes globally but controlled a shard, then the security of the system as a whole is compromised. In order to maintain the security of the system, there are very strict requirements over the number of shards and the number of nodes inside a shard [18].

We see some blockchain-based storage systems proposed in recent years [25]–[33], for most cases, the blockchain is only used as the "contract-signing witness" between the *data publisher* and the *data keeper*. Approaches so far are not trying to reduce the size of the blockchain itself and thus cannot avoid the storage pattern of Bitcoin. Not to mention, the nodes not only need to store the transaction (the contract between the *data-publisher* and the *data-keeper*), they also need to keep some data published by the *data-publishers*.

In this paper, we show a new blockchain structure that cut the blockchain into parts. We use blocks as the input to update the states of the *ledger*. Once a block is accepted, the transactions inside are executed, and then a new state is derived basing on the old one. In Bitcoin case, a state is the balance of every wallet address. A fixed number of blocks are placed into a segment; a segment is stored by different nodes and is retrieved by a user only when the user wants to re-derive a state. The nodes keep the latest state so that they can verify the new transactions. They also need to keep a segment assigned by the system to participate in the mining game. In this model, the Adversary may attempt to cause a permanent loss of a specific blockchain segment by storing all the copies of a blockchain segment and then disappear from the network once succeed. The Adversary may also attempt to deceive the system by claiming it stores a part of the blockchain which it does not store.

Segment blockchain dynamically divides the blockchain into $h//s$ segments following the sequence of blockchain. It requires nodes to show a *PoW* (Proof of Work) [34], [35] when joining in the system as well as every time the nodes present the evidence of storing to the system. In this way, we avoid the Sybil attack [36] and make sure that the Adversary can only take up to $n/2$ of nodes if it has 50% of the overall calculation power. We use a hypothesis taken from our recent blockchain sharding research [18] to label different nodes in Segment blockchain. We categories the participated nodes into different classes and ensures every part of the blockchain is stored by one node per class. Nodes gain the reward for keeping the blockchain segments in the edited game of mining.

Instead of using that hypothesis to build a blockchain sharding system that is more focusing on the improvement of transaction per second, Segment blockchain concentrates solely on the reduce of blockchain size. In blockchain sharding systems, the honest nodes must be the majority of every shard to keep the security of the system as a whole. However, considering storage, when an Adversary fails to become the keeper of every copy of a Segment, its attack does not succeed. So that, in segment blockchain, we do not require the honest people to be the majority of the people who store a specific segment, we only need to ensure that every segment gets at least a reliable keeper. With the loose security

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966464, IEEE Access

threshold, we can assign less number of nodes to store a segment in order to keep that segment securely, compared to blockchain sharding systems where they need to secure the majority nodes are honest. That is why the storage can be much shrunken.

Segment blockchain is suitable for most blockchain applications nowadays including notation [37], identity control [38], multinational customs record exchange [39] which applications do not require a large transaction throughput (like ten thousand to 1 million per second), but get benefit from decentralisation. It also helps the implementation of blockchain in *IoT* environment where the edge devices are lacking storage capacity to keep the full record; meanwhile, the systems are not requiring significant transaction per second [40]. Segment blockchain can even be used to improve the blockchain sharding by separating transaction storage from transaction verification: the nodes in shards only keep the latest state in their shards, and the history transactions are placed into Segments and being handled separately by Segment blockchain.

In the following sections, we will show the Segment blockchain in detail, discuss why our method is secured and can provide a $(AD/n)^m$ failure probability; also, how the model prevents the spoof of storing. We will also show the data requirement compared to the traditional Nakamoto blockchain (Bitcoin).

II. THE *JURY* HYPOTHESIS AND ITS USAGE IN SEGMENT BLOCKCHAIN

We proposed the *Jury* Hypothesis as an analogy of an $n/2$ Byzantine-node tolerate blockchain sharding approach in [18], which turns the blockchain into multiple committees that run in parallel.

The *Jury* Hypothesis states that the member of the Jury of a court comes from the diverse background, so that when a verdict is reached, it can be seen as the decision reached from the whole society (every class of people). If it takes m different occupations to form a jury, then when there are a s number of court hearings run in parallel, there are s number of people in each one of the m occupation. Table 1 shows a court schedule; each court represents a shard, A is a person controlled by the Adversary while H is an honest person.

TABLE 1: Court Schedule

Ocp \ Court number	0	1	2	3
Occupation 1	A	A	A	A
Occupation 2	H	A	H	A
Occupation 3	A	H	A	H
Occupation 4	H	A	H	H
Occupation 5	H	H	H	H

It is ruled that a verdict is reached when a pre-defined T, $T > 0.5m$ number of people inside the jury reached a consensus. Assuming there exists a random assignment scheme that assigns people of the same occupation to different courtrooms where different court hearings are taken place in parallel. Then, the chance for the Adversary to gain T spots inside the target courtroom is (assuming the Adversary put all its nodes into the front T occupations)

$$Pr[T] = \prod_{i=1}^{T} \frac{A_i}{s} \tag{1}$$

where A_i is the number of people inside courtroom i who are controlled by the Adversary. To derive the maximised $Pr[T]$, we want $\prod_{i=1}^{T} A_i$ to be maximised because s is the same. Let the Adversary has AD number of people inside the system (Court Jury Schedule), then $AD = \sum_{i=1}^{m} A_i$. To maximise the value of $\prod_{i=1}^{T} A_i$, we consider

$$A_i = \lfloor AD/T \rfloor, i \in [1, T-1] \tag{2}$$

$$A_T = \lfloor AD/T \rfloor + AD \bmod T \tag{3}$$

This scenario is the maximised because, given any positive integer X,

$$X * X > (X-1) * (X+1) = X * X - 1 \tag{4}$$

Thus,

$$Pr[T]_{max} \approx (\frac{AD}{T * s})^T \tag{5}$$

A. THE *JURY* HYPOTHESIS FOR SEGMENT BLOCKCHAIN

Let the jury of a court stores part of the blockchain, then for the Adversary to control all the people ($T = m$) inside this jury is

$$Pr[T]_{max} \approx (\frac{AD}{m * s})^m \tag{6}$$

Let the court system has a n number of jury members in total ($n = m * s$). Then,

$$Pr[m]_{max} \approx (\frac{AD}{n})^m \tag{7}$$

If the Adverary has no more than 50% fraction of the people inside the system (Nakamoto blockchain threshold), then

$$Pr[T = m]_{max} \approx (\frac{1}{2})^m \tag{8}$$

Thus, the maximum chance for a failure to occur is $(\frac{1}{2})^m$.

B. CHALLENGE

The challenge of implementing *Jury* Hypothesis for the blockchain storage is (1) how to give different nodes different occupations. (2) how to randomly assign storage to a node. (3) how to prove that a node stores a data. (4) how to adjust the membership and reform the jury when nodes go offline.

III. SEGMENT BLOCKCHAIN

Segment blockchain cuts the blockchain into segments. The size and the number of blockchain segments are dynamically adjusted base on the quantity and the occupation of nodes in the system. Every node only stores one blockchain segment and the block header of every block in the mainchain.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966464, IEEE Access

A. BLOCK AS INPUT

Let the blockchain be a *state* machine where every block is the input of the current *state*; the machine reaches the next *state* after processing the current block. In Segment blockchain, every node keeps the latest *state* and all the block headers while storing some copies of the previous blocks. The blockchain is secured when a node can download all the blocks, run them as *inputs*, and derive the same *state*. Figure 1 shows an example of this design.

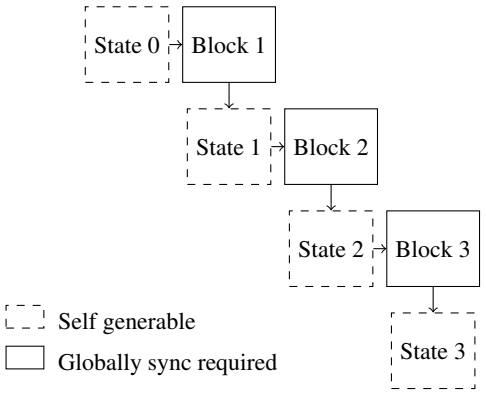

FIGURE 1: An example of the *block* and *state*

In Bitcoin case, the *state* can be a ledger that records the balance of every account.

B. BLOCKCHAIN SEGMENTS

Let there be s number of blockchain segments; every segment takes $h//s$ number of nodes following the index number of the blocks and a *state* derived from the latest block of the previous segment. h is the current length of the blockchain. Except the last segment, other segments are of equal length. The last segment would additionally contain $h \mod s$ blocks. Figure 2 shows an example of this design.

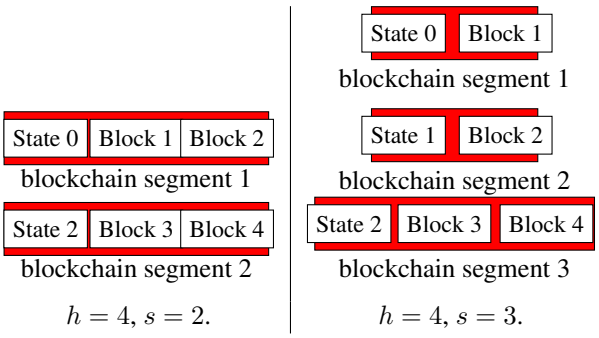

FIGURE 2: The example of the blockchain segments

C. NODE MEMBERSHIP

Let every block has a section which records the pending nodes. The pending nodes are nodes which reported to the system but has not yet been assigned with storage. When a node wants to join in the system, it checks the pending nodes information of the latest block and finds an occupation that is less affluence in number. It then claims that occupation. Let there be a threshold *PoW* difficulty P. The node needs to present a *PoW* of a $P \times s$ difficulty to the system before this node's info (the occupation and a public identity key) is written into the pending node section. Table 2 shows the structure of this *PoW*. Until the node has been instructed to store a specific blockchain segment, the node needs to additionally present a P difficulty *PoW* (of the same content) in every iteration of the mining game to the system to keep the spot in the pending node section.

TABLE 2: PoW for pending node

Name	Storage	Description
Hash_Prev_block	32 bytes	The hash of the block in block height *h-s*.
Occupation	4 bytes	The occupation claimed
Identity Key	32 bytes	Node's public identity key
Nonce	32 bytes	The number tried for this *PoW* to reach the required difficulty

Rank the nodes' info by the time their info is written into the pending section in ascending order into a list PN. Let $PN_{i,j}$ refers to the index j pending node of the occupation i. Every time when $min(len(PN_i)) >= 10, i \in [1, m]$, the storage of all nodes is re-assigned while the size of the blockchain segment is readjusted and 10 more blockchain segments are created $(s = s + 10)$ and $PN_{i,1..10}, i \in [1, m]$ are added to the system. Table 3 and 4 shows an example of the nodes in the system and the PN list; Table 5 and 6 show a possible situation of the nodes in the system and the PN list when the front 10 elements of every occupation in the PN list (Table 4) are added to the system. Because there is a queue for the pending nodes in every occupation, it is nature for the nodes to claim an occupation that is less affluence in number to join in the system quicker. In this way, we solve the challenge (1) of segment blockchain.

TABLE 3: Nodes in the system

Occupation \ blockchain segment	1	2	3	4
Occupation 1	A	A	A	A
Occupation 2	H	A	H	A
Occupation 3	A	H	A	H
Occupation 4	H	A	H	H
Occupation 5	H	H	H	H

TABLE 4: PN list

	PN										
Occupation 1	A	A	H	H	A	A	H	H	H	H	H
Occupation 2	H	A	A	A	H	H	A	A	A	H	H
Occupation 3	A	H	H	A	A	A	H	H	A	A	
Occupation 4	H	A	A	H	H	A	H	A	A	H	H
Occupation 5	H	H	A	H	A	A	H	H	A	H	H

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966464, IEEE Access

TABLE 5: Nodes in the system after adding

Ocp \ Bs	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Occupation 1	H	A	H	H	A	H	A	H	A	A	A	A	A	H
Occupation 2	H	A	H	A	A	A	H	H	A	A	H	A	H	A
Occupation 3	H	A	A	H	H	A	A	A	H	A	H	A	A	H
Occupation 4	A	A	H	A	H	H	A	H	H	H	A	H	H	A
Occupation 5	H	A	H	H	H	H	H	H	H	H	H	A	A	A

TABLE 6: PN list after adding

PN	
Occupation 1	H
Occupation 2	H
Occupation 3	
Occupation 4	H
Occupation 5	H

D. STORAGE ASSIGNMENT

When new nodes are added to the system right after block height h, let $ID_{i,j}$ refers to the identity key of the node of occupation i which stores j blockchain segment. Create a

$$RID_{i,j} = ID_{i,j}\ hash\ BH_h \tag{9}$$

link $RID_{i,j}$ with $ID_{i,j}$ and rank $RID_i,\ i \in [1, m]$ by the ascending order, then adjust ID_i according to the sequence of the ranked RID_i. BH_i is the hash of the block i, $hash$ is a hash function that returns an 256 $bits$ integer. After the procedure above, the assignment is completed. In this way, we solved the challenge (2) of segment blockchain.

E. PROOF OF STORAGE

Let every block header records the Merkle root of the transactions embedded in the block. Let BH_h be the block header hash of the latest block height (block height h). If the node j in occupation i stored the blockchain segment k, then let

$$CI_k = (BH_h\ hash\ ID_{i,j}\ hash\ i)\ mod\ len(k) + 1 \tag{10}$$

where CI_k is the index number of a transaction in blockchain segment k, $len(k)$ is a function that returns the number of transactions inside the blockchain segment k. When the nodes present the evidence of storing the blockchain segment k at the block height h (referred to as *Proof of Storage*), it should provide

- The transaction which CI_k refers to.
- A Merkle branch that can derive the Merkle root indicated in the block header of a block B in the blockchain segment k.
- The transaction which CI_k refers to is inside block B.

Figure 3 shows an example of the *Proof of Storage*. In this way, we solved the challenge (3) of segment blockchain.

F. MINING AND BLOCKCHAIN SEGMENT SIZE ADJUSTMENT

Segment blockchain runs the same mining rule as Nakamoto blockchain for the block creation. Nakamoto blockchain rules that the node which presents a valid block with the most

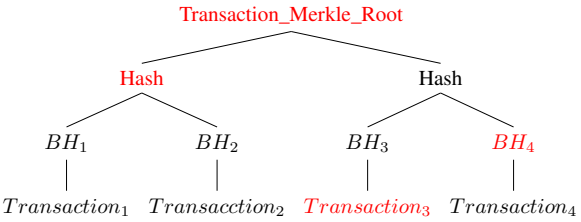

FIGURE 3: The example of a transaction Merkle tree of block i. Let $len(i) = 4$, $CI_i = 3$. The Merkle branch for Proof of Storage contains all the nodes in red.

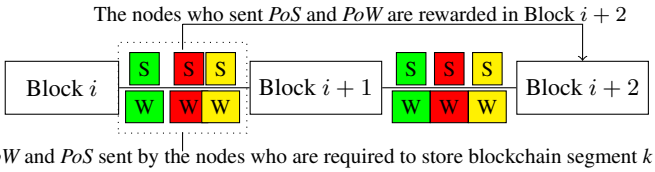

FIGURE 4: A mining procedure example. There are three occupations (red, yellow, green); *S* rectangle represents *Proof of Storage*; *W* rectangle represents a *PoW*

difficult *PoW* in an iteration wins that iteration of the mining game.

Let there be s number of blockchain segments existing in the system, and the current block height is h. Then, the nodes who were assigned to store the number $k = (h\ mod\ s) + 1$ blockchain segment should send the *Proof of Storage* to the network after the block of the block height h is created. They need to submit a *PoW* of $P \times s$ difficulty alongside the *Proof of Storage*. Table 7 shows the structure of *PoW* for these nodes.

TABLE 7: PoW for nodes who are required to store blockchain segment *k*

Name	Storage	Description
Hash_Prev_block	32 bytes	The hash of the block in block height *h-s*
Identity Key	32 bytes	Node's public identity key
Nonce	32 bytes	The number tried for this PoW to reach the required difficulty

The information of the nodes which stored blockchain segment k and presented *Proof of Storage* and *PoW* is embedded in the block of the second next block height. Figure 4 shows an example of this procedure.

When a node which stores the *blockchain segment* k, $k = (h\ mod\ s) + 1$ does not present the *Proof of Storage* and the fulfilled *PoW* at the block height h, the membership of this node is eliminated after the block height $h+2$. If this node is of occupation X, and there is a pending node in PN_X, then $PN_{X,1}$ replaces the eliminated node. If there is no pending node in PN_X, then $s = s - 1$, the number of blocks in every blockchain segment is adjusted accordingly. After that, all nodes storing the *blockchain segment* k before the adjustment are back to the pending node section. In this way, we solve the challenge (4) of segment blockchain.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966464, IEEE Access

G. STORAGE ADJUSTMENT DELAY

When $h//s$ is changed, the components of every segment is changed. Because of that, nodes need to adjust the segments they store by adding or deleting blocks, and sometimes it also needs to derive a new state from the old one. For example, in Figure 2, when s changed from 2 to 3, blockchain segment two would contain "State 1" instead of "State 2". The nodes store segment two would need to derive the State 1 by acquiring segment one and execute blocks since "State 0". It is required that the nodes need to keep the old version of their segments for one block iteration after $h//s$ is changed to avoid conflicts between different versions of segments. In this way, it is providing time for nodes to change the segments smoothly.

H. REWARD

There are two parts of reward in segment blockchain, one for creating a block, one for keeping the blockchain segments. The reward for creating the block is given using the same rule as Nakamoto blockchain (the reward starts from a large amount of currency at first, and cut in half in every fixed time window until reaching zero). The reward for keeping the blockchain segments is given using the following rules:

- When a node showed the *Proof of Storage* and the fulfilled *PoW* in the block height h, the reward is given to this node at the block of the block height $h+2$ (using the node's public identity key as the wallet address).
- The reward is equally divided to every node.
- The amount of the reward for every iteration comes from the system (as like the Nakamoto blockchain). After the reward from the system goes to zero, the reward then comes from the transaction fees.

I. POWER CONSTRAIN

As every node which stores the data are required to present $P \times s$ amount of difficulty per s block height, the node should at least be able to generate a P difficulty *PoW* per iteration. We say the node which can generate P difficulty *PoW* in one iteration has P power. Let $s = 1$, and there is $n \times P$ amount of power globally. The Adversary who has $(n/2) \times P$ amount of power can only keep $n/2$ of nodes in a system of n nodes because it needs to place P amount of power for every Adversary node. If $s > 1$, every node still needs place P amount of power to maintain the spot in the system in every iteration. Otherwise, the Adversary node will be expelled at the next time window (the time window is sized s) for not being able to provide a $P \times s$ difficulty *PoW*. If an Adversary who has P amount of power stop placing power for its node A and tries to gain a new node B, A and B cannot remain in the system at the same time. This is because B also needs $P \times s$ of power in order to become a pending node. Thus, regardless of the number of s, for an Adversary who has $(n/2) \times P$ amount of power, it can only keep $n/2$ of the nodes.

IV. COMBINING $N/2$ BLOCKCHAIN SHARDING WITH SEGMENT BLOCKCHAIN

For blockchain sharding approaches, nodes only store transactions in their shards, so that the storage globally is also divided. However, since blockchain sharding approaches aim to process the transactions in different shards in parallel to improve the transaction per second globally, the system is requiring the honest nodes to be the majority of every shard. As a result, the number of shards that the transactions as a whole can be divided into, in blockchain sharding approaches, is much smaller than the number of Segments that the transactions can be divided into in Segment blockchain.

For an $n/2$ blockchain sharding system which uses Segment blockchain, the nodes keep the latest state in their shards and store different segment of the blockchain (the segments may or may not be one that contains blocks within the nodes' shards). We need to consider two failure probabilities in this system: the chance for the Adversary to control a shard and the chance for the Adversary to take all the copies of a segment. Since the two attacks are unrelated, the maximum failure probability Pr_{max} for a blockchain sharding system in overall is

$$Pr_{max} = max(Pr_{shard_{max}}), Pr_{storage_{max}}) \tag{11}$$

We know that

$$Pr_{shard_{max}} = (\frac{AD}{T \times S})^T \tag{12}$$

and

$$Pr_{Storage_{max}} = \frac{1}{2}^m \tag{13}$$

Let

$$Pr_{max} >= [(\frac{AD}{T * s_1})^T \approx \frac{1}{2}^{(n/s_0)}] \tag{14}$$

where $0.5 * n/s_1 < T <= n/s_1$, T is a pre-defined setting [1]

We can use equation 14 to calculate the maximum number of shards s_1 required for the n/2 blockchain sharding approach to function securely and the maximum number of segments s_0 the Segment blockchain embedded to the system can have in order to maintain the same security threshold Pr_{max}.

Let $AD = \frac{n}{2}$ (this is the maximum AD, because if exceed this number the Adversary would be the majority). Then, we can derive:

$$s_0 = -\frac{n \times log(2)}{log(2^{-T} \times (\frac{n}{s_1 \times T})^T)} \tag{15}$$

Assumed we set the maximum failure probability to be $Pr_{max} = 10^{-6}$ (this bounds the number of s_1), then we derive $\frac{S_0}{S_1}$ which is shown in Figure 5.

As the data in Figure 5 suggested, the blockchain can be divided into $\frac{S_0}{S_1}$ more times of segments than shards, so that the nodes can store a much smaller number of blocks than store all the blocks of a shard. Thus, if a blockchain

[1]The smaller the T is, the less secure a shard is, the larger the T is, the easier a shard can be halted by the Adversary [18].

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2020.2966464, IEEE Access

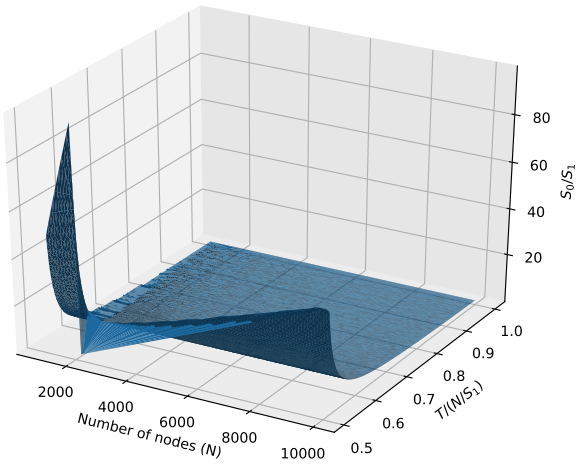

FIGURE 5: s_0/s_1

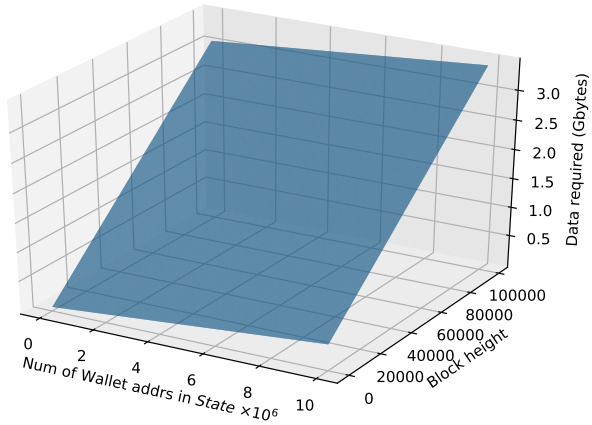

FIGURE 6: Segment blockchain data requirement

sharding system separates the transaction storage from transaction verification by embedding a segment blockchain, the storage requirement for an individual node can be reduced significantly.

V. DATA REQUIREMENT

Let a record in *State* sized $41\ bytes$ (a wallet address sized $33\ bytes$, the balance of a wallet address sized $8\ bytes$). Let a block of the Nakamoto blockchain sized $SB\ bytes$, a block of segment blockchain sized $SB + Size_{Pending\ node\ section}$. A record in $Pending\ node\ section$ sized $68bytes$ ($4\ bytes$ for the occupation of this node, $32\ bytes$ for the public identity key and $32\ bytes$ for the *PoW* it demonstrated). In reality, we can use persistent data structure [41] to store *States*. In this way, the size of storage can be further reduced.

Figure 6 shows the data requirement for the segment blockchain with different number of records in the *State* (let $SB = 1\ Mbytes$, $m = 256$), and with the changes of the block height h. Let there be 8000 nodes (the number of the Bitcoin nodes currently), 256 pending nodes are in the pending node section in every block. Figure 7 shows the differences in the amount of data stored in a node in the Nakamoto blockchain and the amount of data stored by a node in segment blockchain with the different number of accounts in the *state*. Segment blockchain largely shrank the data required while maintaining the full functions of Nakamoto blockchain.

As the analysis in Section 2 showed, the chance for the Adversary who has 50% of the overall calculation power to store all the copies of a block is $\frac{1}{2}^m$. When $m = 256$, the security of segment blockchain reached the security level of the standard public-private encryption systems. It is very safe to use an $m = 256$ segment blockchain to power financial systems because the encryption systems of the same security threshold are already widely tested by the public and used in many online banking systems.

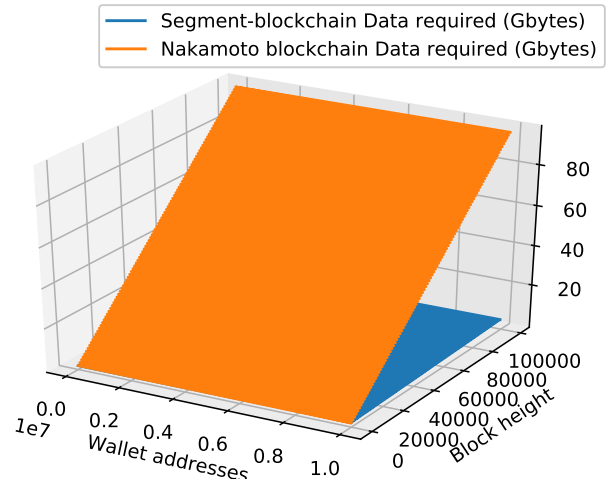

FIGURE 7: Segment blockchain data requirement compared to Nakamoto blockchain data requirement

VI. CONCLUSION

In this paper, we showed an approach to reduce the storage requirement of the blockchain system while keeping the decentralisation without compromising the security of the blockchain. The data analyses proved that segment blockchain largely reduced the data requirement compared to Nakamoto blockchain. Thus, it is of more advantage to using segment blockchain to power data-heavy blockchains.

REFERENCES

[1] Satoshi Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system. 2008.
[2] Bitcoin.com. Bitcoin.com Charts.
[3] Vitalik Buterin et al. A next-generation smart contract and decentralized application platform. white paper, 3:37, 2014.
[4] Christopher D Clack, Vikram A Bakshi, and Lee Braine. Smart contract templates: foundations, design landscape and research directions. arXiv preprint arXiv:1608.00771, 2016.
[5] Alexander Savelyev. Contract law 2.0:'smart'contracts as the beginning of the end of classic contract law. Information & Communications Technology Law, 26(2):116–134, 2017.
[6] Nathanael Rota and Monique Thonnat. Activity recognition from video

sequences using declarative models. In ECAI, pages 673–680. Citeseer, 2000.
[7] Wojciech Samek, Thomas Wiegand, and Klaus-Robert Müller. Explainable artificial intelligence: Understanding, visualizing and interpreting deep learning models. arXiv preprint arXiv:1708.08296, 2017.
[8] Daniel Larimer. Delegated proof-of-stake (dpos). Bitshare whitepaper, 2014.
[9] Ali Dorri, Salil S Kanhere, Raja Jurdak, and Praveen Gauravaram. Lsb: A lightweight scalable blockchain for iot security and privacy. arXiv preprint arXiv:1712.02969, 2017.
[10] Damian Gruber, Wenting Li, and Ghassan Karame. Unifying lightweight blockchain client implementations. In Proc. NDSS Workshop Decentralized IoT Security Stand., pages 1–7, 2018.
[11] Okke Schrijvers, Joseph Bonneau, Dan Boneh, and Tim Roughgarden. Incentive compatibility of bitcoin mining pool reward functions. In International Conference on Financial Cryptography and Data Security, pages 477–498. Springer, 2016.
[12] Yoad Lewenberg, Yoram Bachrach, Yonatan Sompolinsky, Aviv Zohar, and Jeffrey S Rosenschein. Bitcoin mining pools: A cooperative game theoretic analysis. In Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, pages 919–927. Citeseer, 2015.
[13] Sidra Khatoon and Nadeem Javaid. Blockchain based decentralized scalable identity and access management system for internet of things.
[14] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016.
[15] Jacob Eberhardt and Stefan Tai. On or off the blockchain? insights on off-chaining computation and data. In European Conference on Service-Oriented and Cloud Computing, pages 3–15. Springer, 2017.
[16] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. Rapidchain: Scaling blockchain via full sharding. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, pages 931–948. ACM, 2018.
[17] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. Omniledger: A secure, scale-out, decentralized ledger via sharding. In 2018 IEEE Symposium on Security and Privacy (SP), pages 583–598. IEEE, 2018.
[18] Yibin Xu and Yangyu Huang. An n/2 byzantine node tolerated blockchain sharding approach. In The 35th ACM/SIGAPP Symposium On Applied Computing, http://gofun.online/document/blockchain_sharding.pdf, 2020.
[19] Bitcoin Developer Guide. Simplified payment verification (spv).
[20] Pavel Vasin. Blackcoin's proof-of-stake protocol v2. URL: https://blackcoin. co/blackcoin-pos-protocol-v2-whitepaper. pdf, 71, 2014.
[21] Seyoung Huh, Sangrae Cho, and Soohyung Kim. Managing iot devices using blockchain platform. In 2017 19th international conference on advanced communication technology (ICACT), pages 464–467. IEEE, 2017.
[22] Xinxin Fan and Qi Chai. Roll-dpos: A randomized delegated proof of stake scheme for scalable blockchain-based internet of things systems. In Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services, pages 482–484. ACM, 2018.
[23] Conrad Burchert, Christian Decker, and Roger Wattenhofer. Scalable funding of bitcoin micropayment channel networks. Royal Society open science, 5(8):180089, 2018.
[24] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In Symposium on Self-Stabilizing Systems, pages 3–18. Springer, 2015.
[25] Yibin Xu. Section-blockchain: A storage reduced blockchain protocol, the foundation of an autotrophic decentralized storage architecture. In 2018 23rd International Conference on Engineering of Complex Computer Systems (ICECCS), pages 115–125. IEEE, 2018.
[26] Quanqing Xu, Khin Mi Mi Aung, Yongqing Zhu, and Khai Leong Yong. A blockchain-based storage system for data analytics in the internet of things. In New Advances in the Internet of Things, pages 119–138. Springer, 2018.
[27] Shawn Wilkinson, Jim Lowry, and Tome Boshevski. Metadisk a blockchain-based decentralized file storage application. Tech. Rep., 2014.
[28] David Vorick and Luke Champine. Sia: Simple decentralized storage. Nebulous Inc, 2014.
[29] Sheng Wang, Tien Tuan Anh Dinh, Qian Lin, Zhongle Xie, Meihui Zhang, Qingchao Cai, Gang Chen, Beng Chin Ooi, and Pingcheng Ruan. Forkbase: An efficient storage engine for blockchain and forkable applications. Proceedings of the VLDB Endowment, 11(10):1137–1150, 2018.
[30] Jiaxing Li, Jigang Wu, and Long Chen. Block-secure: Blockchain based scheme for secure p2p cloud storage. Information Sciences, 465:219–231, 2018.
[31] Muneeb Ali, Jude Nelson, Ryan Shea, and Michael J Freedman. Blockstack: A global naming and storage system secured by blockchains. In 2016 {USENIX} Annual Technical Conference ({USENIX}{ATC} 16), pages 181–194, 2016.
[32] Richard Dennis, Gareth Owenson, and Benjamin Aziz. A temporal blockchain: a formal analysis. In 2016 International Conference on Collaboration Technologies and Systems (CTS), pages 430–437. IEEE, 2016.
[33] Yongjun Ren, Yepeng Liu, Sai Ji, Arun Kumar Sangaiah, and Jin Wang. Incentive mechanism of data storage based on blockchain for wireless sensor networks. Mobile Information Systems, 2018, 2018.
[34] Martijn Bastiaan. Preventing the 51%-attack: a stochastic analysis of two phase proof of work in bitcoin. In Availab le at http://referaat. cs. utwente. nl/conference/22/paper/7473/preventingthe-51-attack-a-stochasticanalysis-oftwo-phase-proof-of-work-in-bitcoin. pdf, 2015.
[35] Marko Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. In International workshop on open problems in network security, pages 112–125. Springer, 2015.
[36] John R Douceur. The sybil attack. In International workshop on peer-to-peer systems, pages 251–260. Springer, 2002.
[37] Orlenys López-Pintado, Luciano García-Bañuelos, Marlon Dumas, and Ingo Weber. Caterpillar: A blockchain-based business process management system. In BPM (Demos), 2017.
[38] Quinn Dupont. Blockchain identities: Notational technologies for control and management of abstracted entities. Metaphilosophy, 48(5):634–653, 2017.
[39] Yotaro Okazaki. Unveiling the potential of blockchain for customs. WCO Research Pper, 45, 2018.
[40] YJ Ren, Yan Leng, YP Cheng, and Jin Wang. Secure data storage based on blockchain and coding in edge computing. Math. Biosci. Eng, 16:1874–1892, 2019.
[41] Takahiro Kurosawa, Masahiko Yoshimoto, Shigeki Shibayama, and Ryuhei Uehara. Method of managing data structure containing both persistent data and transient data, April 2 1996. US Patent 5,504,895.

YIBIN XU IEEE student member. Student at School of Computer Science and Informatics, Cardiff University Cardiff CF10 3AT, UK. Email:work@xuyibin.top.

YANGYU HUANG Student at School of Electronic Engineering and Automation,Guilin University of Elecctronic and Technology, Guilin 541004, China. Email:i@hyy0591.me.