Trade-off Between Security and Scalability in Blockchain Design: A Dynamic Sharding Mechansim

Kahina Khacef[1], Salima Benbernou[2], Mourad Ouziri[2], and Muhammad Younas[3]

[1] LIP6, Sorbonne Université, France
firstname.lastname@lip6.fr
[2] LIPADE, Université de Paris, France,
firstname.lastname@u-paris.fr
[3] Oxford Brookes University,UK
m.younas@brookes.ac.uk

Abstract. Security and scalability are considered as two major issues that are most likely to influence rapid deployment of blockchains in businesses. We believe that the ability to scale up a blockchain lies mainly in improving the underlying technology rather than deploying new hardware. Though recent research works have applied sharding techniques in enhancing scalability of blockchains, they do not cater for addressing the issue of both data security and scalability in blockchains. In this paper, we propose an approach that makes a trade-off between security and scalability when designing blockchain based systems. We propose an efficient block storage method, which creates dynamic sharding wherein blocks are stored in a varying number of nodes. The proposed method shows that the replication of blockchain over peer to peer network is minimized as the blockchain's length evolves according to a replication factor to preserve the security.

Keywords: Blockchain, security, scalability, sharding

1 Introduction

Blockchain is a distributed data structure that comprises a list of blocks which record transactions such that they are maintained by nodes without a central authority. It has a decentralized peer-to-peer (P2P) architecture which is inherently resilient, decentralized, and open. In it, blocks are (chained or) linked securely by hash pointers that require consensus among different nodes in order to approve every transaction in a block. Blockchain was originally proposed in the white paper [13] by mysterious Nakamoto in 2008, as a proof_of_work (PoW) consensus. It was based on a P2P network which is not required to abide by control of a trusted third party. In the beginning, it was used for bitcoin as a safer way to carry out financial transaction. It is worth noting that the number of blockchains went from one (Bitcoin) to several blockchains in 2021. Blockchain based-systems

have recently become appealing to several financial sectors and scientific communities. Currently there exist various blockchains such as Ethereum [17], Hyperledger [1], Tezos [6] etc. Each blockchain has its own operating mode, a different transaction validation consensus that makes it attractive to various applications.

A user broadcasts a new transaction to network and adds it to the blockchain. A set of nodes then verify the new transaction to ensure that it is correctly signed and that it has not been previously spent (or recorded) in the ledger. Although the nodes of this decentralized network are equal, they can have different roles depending on the functions they support, e.g., routing, database, miner, and wallet nodes. A node with all these functions is called a full node - which maintains a complete copy of the blockchain and contributes to network security. Others nodes, supporting only a subset of functions, verify transactions using a simplified payment verification (SPV) method, known as lightweight nodes - they allow to send and receive transactions without owning a full copy of the blockchain. But they download headers of blocks and transactions concerned depend on full nodes.
Moreover, blockchain has won its spurs in security, whether for user's governance of data, transparency, and immutability (this is not clear ???). However, security is achieved at a price of maintaining full nodes. Storage costs increase linearly with the increase in number of transactions and may become one of the bottlenecks that limit blockchain's scalability. Traditional blockchain is based on full replication, where nodes rely on all past transactions locally. They check the state to validate a new transaction and then store each transaction to maintain the system. This is to represent proof of correct state that consists of all block headers starting from the genesis's block. This processing is slow and requires a lot of storage capacity. Each node has to agree with that process. However, supporting a large number of users and transactions result in a serious scalability problem.

On one hand, the decentralization of blockchain on a peer-to-peer network and its replication on multiple nodes, provide extra security by making it more difficult for an attacker to compromise the system. On the other hand, these negatively affect the scalability of blockchain systems. It is believed that the ability to scale up a blockchain therefore lies mainly in improving the technology, and not in the deployment of new hardware. This paper presents a new blockchain design method in order to reduce the storage volume on each node and allow dynamic sharding with different local states from node to node but without compromising on security properties. This is based on proof-of-work that allows them to follow the evolution of the entire blockchain (I changed this text but it is not clear to me. Please check ???).

The main contributions of this paper are summarized as follow:

– It proposes to design a new blockchain based system that makes a trade-off between security and scalability in a blockchain. The proposed system is named as SecuSca.

- It formulates trade-off approach as a multi-objective and multi-constraint optimization problem. The proposed mechanism allows a dynamic sharding to save the replications in the network while maintaining appropriate security of the blockchain systems.
- It implements the proposed approach as a proof of concept and highlights its efficiency.

The remainder of this paper is organized as follows. Section 2 presents a motivating example as well as overview of the dynamic sharding approach. Section 3 discusses the related work. Section 4 provides background on blockchains. Section 5 describe the proposed design which enhances the scalability of the blockhain while maintaiing the security. Section 6 provides the sharding function to optimize the trade-off. Section 7 concludes the paper.

2 Overview of SecuSca approach and Motivation

This section presents a running example in relation to the proposed SecuSca framework that aims to overcome scalability limitation of blockchain based systems while maintaining their security. It also presents an overview of SecuSca model.

(approach, mechanism, method, framework - these are four different terms used for the proposed work. It would be useful to use consistent term - proposed approach or mechanism or method or framework ???)

2.1 Motivating example

The blocks that make up blockchain are replicated on all nodes. They maintain local copies of all blocks (including genesis block) for the following reasons:

i To verify a transaction, the nodes read the history of all past transactions locally.
ii To provide replication as it enhances security against attacks and tampering, and also improves availability of data.
iii To safeguard transactions - transactions are considered sufficiently safe from attacks when buried under enough blocks, and miners reach consensus by selecting the longest one. In proof_of_work cryptocurrencies, the longest chain is deemed honest by the network, regarded as the most invested chain [14].

We consider Alice, Bob and John as three participants among hundred nodes in the blockchain network which has a storage capacity of 50 GB that holds shared ledger. As shown in Fig.1, the blockchain is fully replicated on every node in the network. All nodes store whole blocks with all transactions, and the same block is replicated on all nodes.
Suppose that the size of a block is $1MB$, and that blocks are generated every $10minutes$, $1MB{*}6{*}24{*}30 = 4320MB$ per month. Since each block is replicated in all nodes, the three nodes can store up to $50 * 10^3$ blocks. These nodes with a

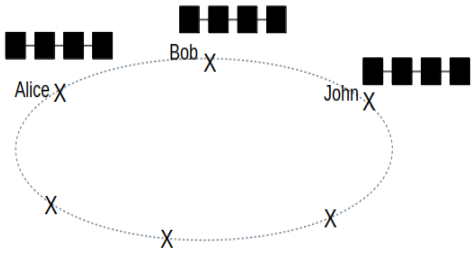
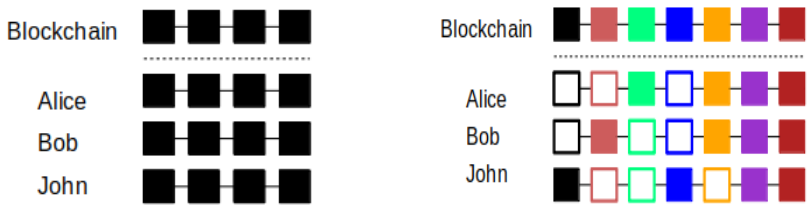


Fig. 1: Full Replication

capacity of $50GB$ containing the blockchain will be saturated in less than a year. This shows that current design of blockchain would result in major scalability issue.



(a) Traditional blockchain (b) New replication model

Fig. 2: Traditional blockchain *vs* the new replication model. In 2(a), each node maintains its chain which contains all previous transactions. Each given block is stored on the three nodes, Alice, Bob, and john. While in 2(b), the global blockchain shown at the top is sharded across the three nodes. Colored blocks represent the entire block containing transactions, and framed blocks only have the block header - so the block buried in the chain is held on a few nodes.

Reducing replication reduces storage space of nodes so they can continues to receive new blocks. Thus blockchain can contain more than 50,000 blocks. As shown in Fig. 2, blockchain distributed over networks is more scalable in storage than that of full replication.

2.2 The SecuSca approach in a nutshell

SecuSca aims to reduce storage load by reducing replication of each block in a distributed ledger. It takes into account fundamental characteristics of security and verification as mentioned above. The blockchain is distributed on nodes,

and all blocks make the overall state. To allow a node for verifying history of the blockchain, SecuSca removes transactions from blocks and keeps header block. The new block produced is cryptographically linked to the last block which is added to the longest chain. As blocks arrive in the system, they are stored on a higher number of nodes. The replication of blocks secured by chaining decreases when the blockchain becomes longer on each node. Each block is not entirely removed from the blockchain. As a result, it is possible to store a larger number of transactions than current blockchain systems by using same storage capacity. It also reduces network throughput (or delay. please check ???) and memory required for each node individually. Also, with an increase in transactions, users are not forced to store a large amount of data.

3 Related work

The most widely used blockchains of Bitcoin [13] and Ethereum [17] are based on a full replication system which cannot support a large number of transactions. With growing number of transactions, they result in system overload that does not allow to scale. In effect, the efficiency of blockchain decreases as more nodes join the network. In order to tackle this issue, many approaches have recently emerged that allow blockchain to continue functioning even when the number of users increases.

Sharding, generally used in databases, is proposed in cryptocurrency ledger. The network of N nodes is partitioned into committees k with a small number of nodes c, with replication, $c = N/K$ - i.e., to yield smaller full replication systems. The node in each committee K stores only validated blocks inside its committee and does not manage the entire blockchain ledger. As an example, [8, 11, 18] are sharding-based Proof_of_Work and Byzantine fault tolerance (BFT). *Elastico* [11] is the first sharding-based public blockchain proposed in 2016 that tolerates byzantine adversaries. It partitions the network into shards and ensures probabilistic correctness by randomly assigning nodes to committees, wherein each shard is verified by a disjoint committee of nodes in parallel. It executes expensive PoW to form a committee, where nodes randomly join different committees and run PBFT or Practical Byzantine Fault Tolerance for intra-committee consensus. In Elastico, all nodes maintain the blockchain ledger but cross-shard transactions are not supported. In addition, while running PBFT among hundreds of nodes decreases protocol's performance, but reducing the number of nodes within each shard increases the failure probability. The network can only tolerate up to 25% of malicious nodes.

Omniledger in [8] improved upon *Elastico*. It includes new methods to assign nodes into shards with a higher security guarantee, as *Elastico*. It uses both Pow and BFT; an atomic protocol for across-shard transactions (Atomix). The intra-shard consensus protocol of OmniLedger uses a variant of ByzCoin [7] and assumes partially synchronous channels to achieve faster transactions. The network tolerates up to 25% of faulty nodes and 33% of malicious nodes in each committee as in [11].

Cross-shard in *Rapidchain* [18] relies on an inter-committee routing scheme which is based on the routing algorithm of Kademlia [12]. It tolerates up to 33% of total resiliency and 50% of committee resiliency. *Rapidchain* also supports cross-shard transactions using Byzantine consensus protocols but requires strong synchronous communication among shards which is hard to achieve. There exist other approaches in the literature which are based on private blockchain [2, 3, 5, 15, 16]. Even though sharding improves storage and throughput, K increases linearly with N with a low-security level, thus, leading to malicious node errors.
Vault [9] introduces fast bootstrapping to allow new participants to join the network without downloading the whole blockchain by reducing the transmitted state. *Vault* is Account-based for Algorand [4], and does not require all nodes to store the whole blockchain state.
In [10], authors propose a superlight client design to allow a light client to relay full nodes to read blockchain with a low read cost to predict (non) existence of a transaction in a blockchain. Therefore, blockchains can hold a large amount of data. However, each node requires storage space. Thus, the cost of storage and the required memory increase with the number of transactions.

4 BACKGROUND

In this section, we present relevant background on blockchain systems in relation to the proposed approach.

4.1 Blockchain systems

Blockchain network. A blockchain network consists of nodes that can record all data in an immutable way. The data structure is collected in blocks that contain sets of transactions, and the consensus of most network participants verifies these transactions. Each block is identified by a hash, a unique identifier. In our proposed approach, we assume that all nodes always have same constant amount of resources with respect to CPU, storage, and network bandwidth.
Data model. Different blockchain uses different models for their states. The main models are Unspent Transaction Output (UTXO) and the account-based. Bitcoin adopts the UTXO model and many other cryptocurrencies, consisting of outputs of transactions that have not been considered inputs to another transaction. UTXO provides a higher level of privacy. More recent blockchains-based systems, such as Ethereum and Hyperledger, support general states that can be modified arbitrarily by smart contracts. They adopt an account-based data model, in which each account has its local states stored on the blockchain; it is similar to the record-keeping in a bank. Our proposed approach focuses on UTXO as it will involve reading history of transactions and verify the states.
Block structure. A block stores transactions and global states. A block header contains the following fields: (1) Previous Block -reference to the previous block in the chain, (2)Nonce-related to the proof_of_work, (3) Merkle root -enable

efficient verification of transactions and states, (4) A set of metadata related to the mining protocol; difficulty and timestamp.
Consensus. It is an agreement to validate correctness of a blockchain, which is linked to the order and timing of block. Its goal is to achieve consistency of the nodes participating in blockchain. All honest nodes accept the same order of transactions as long as they are confirmed in their local blockchain views. The blockchain is constantly updated as new blocks are received.

4.2 States management

States. The system ensures data availability and transparency for all participating members by maintaining all historical and current transactions at each node for blockchain verification. The main abstraction of reading blockchain requires user to maintain a full node to run the consensus protocol and maintain a local replica of a blockchain. An increase in number of participants makes blockchain system complex and leads to saturation of network. This leads to substantial transaction costs to process data with increased storage space which degrades network performance.
Censorship-resistance. Data stored in the blockchain cannot be tampered with during and after block generation. An adversary will fail to modify historical data stored on blockchain because of cryptographic techniques used in distributed blockchain storage: (1) Asymmetric Key - that each node uses to sign and verify the integrity of the transaction, and (2) Hash function - a mathematical algorithm that maps arbitrary size data to a unique fixed-length binary output. A hash function (e.g., $SHA-256$) is computationally infeasible to recover input from output hash. An attacker fails to tamper with a block after a size t of the blockchain sufficiently secure according to [14], with t is the number of blocs in the blockchain. Even if the adversary tries to cover up this tampering by breaking the hash of the previous block and so on, this attempt will ultimately fail when the genesis block is reached. It is complicated to modify data blocks across the distributed network. A small change in the original data makes the hash unrecognizable different, which secures the blockchain.

5 The proposed dynamic sharding mechanism

In this section, we first discuss an overall architecture of the proposed approach, SecuSca. We then describe the dynamic sharding approach that preserves the security and scalability of storage in the blockchain.

5.1 Architecture of SecuSca

An architecture of the proposed SecuSca approach is depicted in Fig 3. Users trigger transactions which are inserted into a block and are added and replicated in the chain of blockchain over the network. In order to preserve a maximum

security and scalability of the blockchain by decreasing its full replication, SecuSca creates a trade-off between security and sclability. The process comprises two steps that operates at the time. An optimization function is introduced in order to help the sharding process:

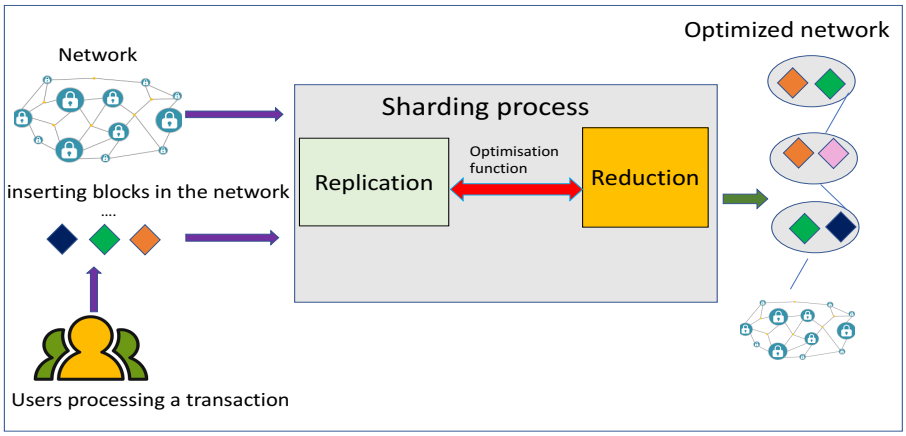


Fig. 3: An overview of functional architecture of SecuSca

– *Efficient replication*: The replication means efficient storage and security. This approach distributes/replicates the global state of the blockchain to some nodes of the network by dynamic sharding without sacrificing security. The fundamental goal is to preserve the security and scalability of the blockchain and store data in the future where blockchain applications are variants; even if the node has poor memory storage can participate in the protocol process.

– *Efficient reduction*: While the replication of an inserted block is in the process, a reduction step is operating in order to reduce the replication of some blocks over the network and then scaling up the blockchain. This allows more storage in the network of the blockchain. In the next section we will detail the SecuSca approach.

5.2 The process of SecuSca

Step 1: blocks. Unlike current blockchain systems, in SecuSca, a block keeps concatenated and hashed transactions in the Merkle root in the header (this is not clear ???). Transactions are explicitly represented in the block broadcast when it is newly added. But transactions are removed from the block with other blocks after it (after what ??? this is not clear ???). Indeed, transactions are proportional to the network load. Forcing a node to store all would affect scalability.

For each new block when it is first added to the blockchain, the transactions it contains are not confirmed until a specific size constitutes other blocks after this (new) block. During this phase, the block is entirely stored. When transactions in the block are confirmed by the network and the block buried in the blockchain, transactions are deleted from the block and kept only in the Merkle root. Thus, a minimum number of nodes keep all transactions of this block in the blockchain.

Step 2: replication model. In SecuSca, we are interested in storing transaction's history. Unfortunately, for scalability reasons, all nodes cannot store the whole state. The blockchain is distributed over a higher number of nodes that store b_i blocs (where $i = \{1, ..., B\}$); each node is represented by n_j (where $j = \{1, ..., N\}$). The blocks are propagated throughout the network and stored on nodes n with ($n <= N$). A node may not even store any state and can still participate in transaction confirmation and block process. The number of malicious nodes q that the system can tolerate cannot exceed half of the nodes in the network. The honest nodes p can be resilient under the presence of malicious nodes q if their computing power is greater than the computing power of malicious nodes. Therefore, the security of the blockchain is guaranteed by honest nodes, as so the replication of the blocks at the beginning of the process must be large enough to discourage malicious nodes from attacking the network. The number of nodes adding the block should be higher for security reasons, but, without replicating it across the network. t denotes the number of blocks contained in the blockchain that increases with the arrival of new blocks. The replication of the block is related to the size t of the blockchain in each node n_j. In the beginning, it is maximum because the size is small, then it decreases as the size increases.

Step 3: reduction model. In SecuSca, reducing block replication does not mean deleting the whole block, but only the explicit transactions and keeping the block header of all blocks which are added to the blockchain. Cryptographically, buried blocks in the blockchain are more secure. When the size t increases, the replication of the older block decreases. A transaction is only deleted when it is confirmed in the blockchain and is secured by chaining. The last blocks added to the blockchain are not reduced because they are less secure. For any new block b_i added to the blockchain with a high replication, the approach selects a replication of all past blocks; each node frees memory space by erasing transactions from the older blocks. Every node stores part of the blockchain state (a subset of all history) and header blocks, including the Merkle roots for the whole blockchain. SecuSca allows availability of the overall state of the blockchain, and keeps a minimum replica of the whole blocks distributed on nodes.

6 Block insertion in the dynamic sharding

When a new block is ready to be inserted in the blockchain and replicated over the network, the sharding process follows an optimization function $\mathcal{R}$. In this

section, we will define $\mathcal{R}$ allowing the trade-off between replication for security and scalability of the blockchain. Before we discuss the optimization function, let's introduce some useful definitions used in the setting of the function. We first introduce the notion of the depth of a block which is given by the number of blocks added to the chain after it. It is formally defined as:

Definition 1. ***(Block depth** d**).*** *Let's consider, t, the size of the blockchain and ,i, the position of a block b_i in the chain, we define the depth of a block b_i as follows: $d = t - i$.*

For example, if a blockchain contains three blocks [b1, b2, b3], the depth of b1 is 2, the depth of b2 is 1, and the depth of b3 is 0.

Definition 2. ***(The boundaries thresholds** α_N, α_0 **for security.)*** *Let's consider, N, as the number of nodes in the network that host the blockchain,*

- *the upper bound α_N in the replication phase is the estimated highest number of nodes where the blockchain should be fully replicated in all nodes such as $\alpha_N < N$ to ensure the security. (is this maximum security or just security ???)*
- *the lower bound α_0 in the replication phase is the estimated lowest number of nodes where the blockchain should be replicated to ensure a minimum of security such as $\alpha_0 < \alpha_N < N$.*

Definition 3. ***(The boundaries thresholds** γ_0, γ_B **for scalability.)*** *Let's consider B as the higher size of the blockchain. In order to ensure security and optimize scalability we define:*

- *the upper bound γ_B in the replication phase is the estimated highest depth of a block from which we can not go under to stop the reduction of the blocks in different nodes, where $\gamma_B < B$*
- *the lower bound γ_0 in the replication phase is the estimated lowest depth of a block (to be replicated) and from which the replication starts to be reduced while preserving the security and spread up the scalability, such as $\gamma_0 < \gamma_B$,*

By means of the definitions of block depth in a blockchain and the boundaries, let's define, the sharding function to ensure the trade off between security and scalability:

Definition 4. ***(The sharding optimisation function** $\mathcal{R}$**.)*** *Let's, d, be a depth of block b in a blockchain. According to the steps of SecuSca, the number of replications of any block over is defined as:*

$$\mathcal{R}(d) = \begin{cases} \alpha_N & \text{if } d < \gamma_0 \\ \frac{\alpha_0 - \alpha_B}{\gamma_B - \gamma_0} * (d - \gamma_B) + \alpha_0 & \text{if } \gamma_0 < d < \gamma_B \\ \alpha_0 & \text{if } \ d > \gamma_B \end{cases}$$

7 Implementation and discussion

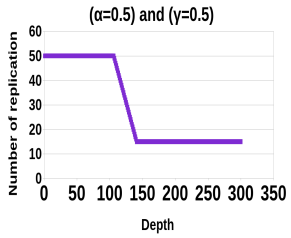
We simulate a blockchain in order to validate how our proposed SecuSca offers a trade-off between security and scalability.

7.1 Experimental Setup

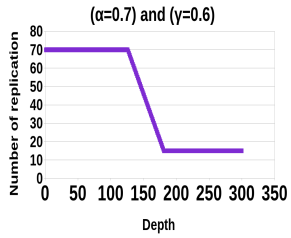
We evaluate our approach and compare it with traditional blockchain (bitcoin) in two experiments by varying the replication of block and block depth. We run multiple simulations with different values of α and γ parameters in order to define *upper bound* αN and *lower bound* $\gamma 0$. We then give the size of the whole blockchain sharded over the network.

7.2 Experiments studies

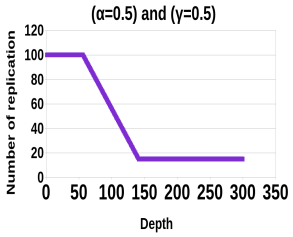
In the experiments we studied two aspects; the efficiency in terms of block replication; and the size of the blockchain.



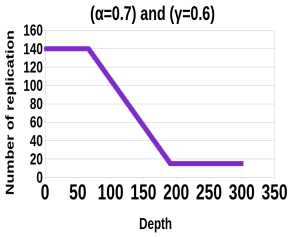
(a) 100 nodes with 200GB memory



(b) 100 nodes with 200GB memory



(c) 200 nodes with 100GB memory



(d) 200 nodes with 100GB memory

Fig. 4: The replication of blocks according to block depth

- Replication of blocks.

When a miner produces a new block, it is broadcasted over the network and added to the local disk of nodes. First, we simulate the function $\mathcal{R}(d)$ with 100 nodes with 200 GB storage capacity. See Fig.4 (a) and (b). Each node performs the sharding optimization function $\mathcal{R}$. At the start of the process, the new block has a depth of zero. No block is chained to it, and its replication is high. We fixed the parameter ($\alpha = 0.5$) and ($\gamma = 0.5$), and then ($\alpha = 0.7$) and ($\gamma = 0.6$). For the first simulation, the *upper bound* is given by 50, and the block is replicated on 50 nodes. This initial replication is constant until it reaches depth of γ_0.

A second simulation with 200 nodes with a storage capacity of 100 GB is given in Fig. 4 (c) and (d). The replication of each block depends on its depth in the blockchain. It decreases as the block's depth increases. Our system reduces the replication of block to α_0. It is the lower number of replication that will be stored in the blockchain for each block. For our experimental results, we consider that 15 replicas of a block are sufficient to maintain state of the blockchain as the size increases ($\alpha_O = 15$).

-Size of the network.

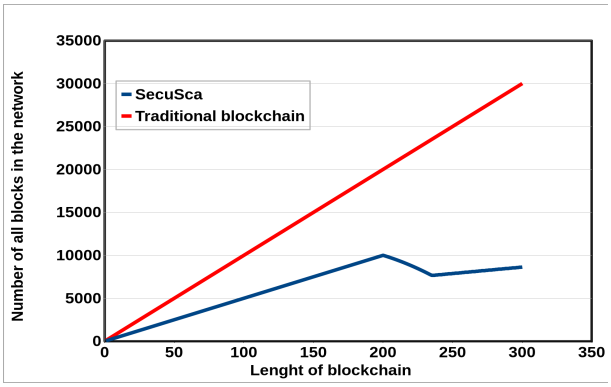


Fig. 5: Comparison of the size of blockchain between traditional blockchain and SecuSca

In the experiment that follows, we run the algorithm over 100 nodes that produce 300 blocks of $1MB$ size. Fig.5 gives an overview of the evolution of the size of blockchain in both traditional blockchain and SecuSca. The experiment reveals that the overhead of blockchain becomes significant in traditional blockchain traced in red. The size of SecuSca is the sum of all blocks stored on each node, from the first block to the last block. The sum of all transactions of all shards is traced in blue. From [0-200], every block is highly replicated, and the size grows linearly. After that, nodes start reducing replication until the lower bound of our approach.

The size t of the shard in each node n at different steps is given by:

$$t_n = \sum_{i=0}^{b} \frac{R(i,t)}{N} \tag{1}$$

where N is the number of nodes, i is the position of a block b_i is in the chain and t is the size of the blockchain.

-**Discussion.** This section discusses the effectiveness of our approach and certain aspects that are not analyzed. This work aims to design an optimal function for blockchain scalability that maintains security. The above analytical and numerical results show, how our proposed SecuSca approach, promotes scalability, and enables users to store more transactions by freeing up local disk. Nevertheless, SecuSca needs further improvements. For instance, it should allow the blockchain to continue to function even when some nodes delete transactions from their local chain. At each transaction verification, if the needed data is not on a local chain of each node, this node must retrieve them from another node despite sharing state between nodes. Inter-shard communications should be part of the consensus protocol.

8 Conclusion

In this paper, we proposed a new design of a blockchain which makes a trade-off between security and scalability that allow for more capacity in terms of storage in the whole blockchain. We develop a dynamic sharding approach which is formulated as an optimisation problem which is preserving security and scaling up the blockchain. We have conducted various simulation-based experiments. Experiments have shown promising results and significant improvement over traditional blockchain. As a future work, we aim to investigate querying our new design of the blockchain to access the data.

References

1. Hyperledger Architecture Volume 1. Introduction to hyperledger business blockchain design philosophy and consensus. *accessed on May 20, 2021 https://www.hyperledger.org/wp-content/uploads/2017/08/Hyperledger_Arch_WG_Paper_1_Consensus.pdf*.
2. Mohammad Javad Amiri, Divyakant Agrawal, and Amr El Abbadi. On sharding permissioned blockchains. In *IEEE International Conference on Blockchain, Blockchain 2019, Atlanta, GA, USA, July 14-17, 2019*, pages 282–285. IEEE, 2019.
3. Mohammad Javad Amiri, Divyakant Agrawal, and Amr El Abbadi. Sharper: Sharding permissioned blockchains over network clusters. *CoRR*, abs/1910.00765, 2019.
4. Jing Chen and Silvio Micali. Algorand: A secure and efficient distributed ledger. *Theor. Comput. Sci.*, 777:155–183, 2019.
5. Hung Dang, Tien Tuan Anh Dinh, Dumitrel Loghin, Ee-Chien Chang, Qian Lin, and Beng Chin Ooi. Towards scaling blockchain systems via sharding. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD*

Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019, pages 123–140. ACM, 2019.
6. L.M Goodman. Tezos a self amending crypto ledger white paper. *accessed on May 20, 2021 https://tezos.com/whitepaper.pdf*, 2017.
7. Eleftherios Kokoris-Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. *CoRR*, abs/1602.06997, 2016.
8. 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 2018, Proceedings, 21-23 May 2018, San Francisco, California, USA*, pages 583–598. IEEE Computer Society, 2018.
9. Derek Leung, Adam Suhl, Yossi Gilad, and Nickolai Zeldovich. Vault: Fast bootstrapping for the algorand cryptocurrency. In *26th Annual Network and Distributed System Security Symposium, NDSS 2019, San Diego, California, USA, February 24-27, 2019*. The Internet Society, 2019.
10. Yuan Lu, Qiang Tang, and Guiling Wang. Generic superlight client for permissionless blockchains. *CoRR*, abs/2003.06552, 2020.
11. Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, October 24-28, 2016*, pages 17–30. ACM, 2016.
12. Petar Maymounkov and David Mazières. Kademlia: A peer-to-peer information system based on the XOR metric.
13. Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *White paper*, 2008.
14. A. Pinar Ozisik, George Bissias, and Brian Neil Levine. Estimation of miner hash rates and consensus on blockchains (draft). *CoRR*, abs/1707.00082, 2017.
15. Harmony team. Harmony. *Technical White paper*, 16:Accessed May 2021: https://harmony.one/whitepaper.pdf, 2017.
16. ZILLIQA team and Others. The zilliqa. *Technical White paper*, 16, 2017.
17. V.Buterin. Ethereum, white paper. *accessed on May 20, 2021 https://ethereum.org/en/whitepaper/*, 2017.
18. Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. Rapidchain: Scaling blockchain via full sharding. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 931–948. ACM, 2018.