A Dynamic Sharding Model Aware Security and Scalability in Blockchain

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

[1]Sorbonne Universite, LIP6, Paris, France
[2]Universit´e Paris Cite, LIPADE, Paris, France
[3]Oxford Brookes University, Oxford, UK

Abstract

Security and scalability are considered as two major challenges in rapid and smooth deployment of blockchains in businesses, enterprises and organizations. Common belief is 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 joint issue of data security and scalability in blockchains. This paper proposes an approach that makes a trade-off between security and scalability when designing blockchain based systems. It designs an efficient replication model, which creates dynamic sharding wherein blocks are stored in a varying number of nodes. In order to maintain required level of security, the proposed approach shows that the replication of blockchain over peer-to-peer network is minimized as the blockchain's length evolves according to a replication factor.

Keywords Blockchain, Security, Scalability, Sharding

1 Introduction

Blockchain is a distributed data structure which consists of a list of blocks that record transactions. These are maintained by nodes which do not require a central authority or a coordinator. It uses a peer-to-peer (P2P) network that is open, decentralized and resilient network. Blocks are securely connected by hash pointers, which require consensus from multiple nodes when approving transactions in a block. Blockchain was initially proposed by an enigmatic person (named Nakamoto) in 2008 as a proof-of-work (PoW) consensus. It was first used for bitcoin in order to securely conduct financial transactions. It was deployed on a P2P network that was not bound by the control of a third party or a central authority (Nakamoto,- 2008). Blockchain has lately been adopted in numerous financial organizations and other industries. It is also used in Internet of Things (IoT), e-Health, self-sovereign identity management [14], Public Key Infrastructures (PKIs) and decentralized key management.

Each blockchain has its own operating mode with dedicated transaction validation consensus that makes it attractive to various applications. For example, in cryptocurrency, a transaction is a digitally signed statement (currency) that states the transfer of ownership of a digital property (or asset). In bitcoin, a transaction is signed cryptographically by one participant which is then sent to one or more anonymous participants who are only known by their public keys. Any node can join or exit blockchain P2P network at any time without authentication.

A user adds a new transaction to the blockchain whenever it is broadcasted to the network. The new transaction is then received and verified by a group of (computer) nodes in order to ensure that it is signed precisely and that it has not previously been spent (or recorded) in a general ledger. These are then grouped into valid transaction blocks.

Miners are compensated for their labor by receiving a fee each time a block is added and agreed upon. Despite the fact that all nodes in this decentralized network are equal, they can perform different functions, such as routing, database operations, mining, and wallet services. A full node is a node that performs all such services, contributes to network security and maintains a complete copy of a blockchain. A lightweight node performs a handful of functions and uses simplified payment verification (SPV) approach in verifying transactions. In addition, a lightweight node allows to transmit and receive transactions without owning a whole blockchain. They do, however, download block headers, and the transactions in question rely on full nodes.

In terms of data integrity, security, and immutability, blockchain has earned its stripes. An ideal blockchain system enables a decentralized control where no single party has an a priori privileged role. It also provides consensus on a single state (single source of truth), and tamperproof recording such that validated transactions cannot be deleted or updated. Furthermore, ideal blockchain preserves privacy of data using cryptographic techniques such as cryptographic hashing, private-public key cryptography.

In blockchain system, security comes at the cost of maintaining full nodes. Storage costs rise in lockstep with an increase in number of transactions, which become a barrier in blockchain's scalability. Full replication, in which nodes rely on all previous transactions locally, is the foundation of traditional blockchain, where nodes validate a new transaction by checking the state, and then store each transaction to keep the system running. This is used to convey proof of correct status, which is made up of all block headers beginning with the genesis block. This operation is slow and requires a large amount of storage space.

Based on the above discussion, it is observed that the decentralization, security, and scalability trilemma is the most pressing concern in blockchain. Though blockchain technology is fast evolving it still faces fundamental and practical challenges in gaining widespread adoption. On the one hand, the decentralization of blockchain on a P2P network and its replication on multiple nodes, provide extra security by making it more difficult for an attacker to compromise the system. It must be made secure [18, 19] to ensure that it can withstand specific types of assaults that could imperil its proper functioning. To achieve decentralization, solutions must allow independent participants with diverse assumptions about how participants might join or depart a network. 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 designs a new blockchain system in order to reduce storage volume on each node and to allow dynamic sharding with different local states from node to node but without compromising on security. The main contributions of this paper are as follows.
It designs a new blockchain system namely SecuSca that maintains an appropriate balance and trade-off between security and scalability. In addition, it formulates trade-off as a multi-objective optimization problem. The proposed approach allows a dynamic sharding to save replications in the network while maintaining appropriate level of security. Furthermore, it implements the proposed approach as a proof of concept and evaluates its efficiency.

The remainder of this paper is organized as follows. Section 2 presents the motivation through an example of the proposed approach, a useful background to make the paper understandable and an overview of the approach. Section 3 describes the dynamic sharding architecture and the approach that preserves balance between security and scalability. Section 4 presents the dynamic sharding calculus.- Section 5 gives an overview of the existing approaches and their drawbacks. The conducted experiments are presented in Section 6.

2 The SecuSca approach and a motivating example

Every node in all of the blockchain protocols with a backbone and production environment that we see today is responsible for storing all states and processing all transactions. This enhances security but restricts scalability. Furthermore, the blockchain network must maintain its own features, such as proof and traceability; which result in an increase in the amount of data that each node must store. This section first presents a running example in relation to the proposed SecuSca approach that aims to overcome scalability limitation of blockchain based systems while maintaining appropriate level of security. It then presents an overview of SecuSca model.

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:

1. To verify transactions - the nodes read the history of all past transactions locally.
2. To provide replication - it enhances security against attacks and tampering, and also improves availability of data.

3. 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, and is regarded as the most invested chain (Pinar Ozisik et al.,2017).

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.

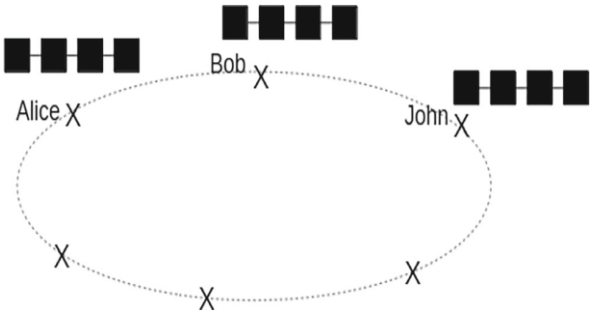


Fig. 1 Full Replication

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 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.

In Fig 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 Fig 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. Decreasing the replication reduces the storage space of the nodes so they can keep receiving new blocks. Thus blockchain can contain more than 50,000 blocks. Consequently, as shown in Fig 2, the blockchain distributed over the network is more scalable in storage than that of full replication.

2.2 An overview of SecuSca approach

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 but keeps the header block. The new block, produced cryptographically, is linked to the last block and 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 the same storage capacity. It also reduces the memory required for each node individually. Also, with an increase in transactions, users are not forced to store a large amount of data. Below we present some important concepts and definitions which are used in the design and development of the proposed approach.

2.2.1 Distributed ledger vs blockchain

A distributed ledger is a database that exists across several locations or among multiple parties which is decentralized in a way that does not need a central authority to process, validate or authenticate

transactions. All files in the distributed ledger are timestamped and are given unique cryptographic signature (Ioini & Pahl, 2018; Lux et al., 2020).

Blockchain is one of the types of Distributed Ledger Technology (DLT). It is essentially a shared database filled with entries that must be confirmed and encrypted. Transactions are entered in the DLT in chronological order and are recorded as a series of blocks in the general ledger. An interconnected chain is built between the blocks with each block referring to the block before it, thus forming a blockchain. The major distinction is that blockchain is a sequence of blocks, but DLT do not require such a chain. Furthermore, DLT do not need proof of work and offer, which provide better scaling options. Moreover, blockchain requires global consensus across all nodes, but DLT can establish consensus without validating the full blockchain. Blockchain requires a peer-to-peer network with consensus techniques so as to assure replication between nodes. But DLT is dispersed over numerous nodes (devices). Each node replicates and saves an identical copy of the general ledger and updates itself independently. Once a consensus is reached, all other nodes are updated with the new, correct version of the ledger.

2.2.2 Blockchain

A blockchain is a type of distributed ledger maintained by a group of validators that stays correct even if some of the validators are malicious. When each party receives blocks of transactions, they save a copy of the ledger on their computer and update it according to the protocol's rules. The purpose of blockchain technology is to ensure that the ledger is correctly replicated, which means that at any given time, each honest party sees the same version of the ledger. A blockchain network is made up of nodes that can maintain an immutable record of all data. The data structure is organized into blocks, each of which contains a set of transactions that are verified by the majority of network participants. The messages sent by the nodes are signed, and a hash, or unique identifier, is used to identify each block. Transactions and blocks need time to transfer from one participant to another as they pass through the protocol. Depending on the environment in which they are designed to operate, different protocols use different network models. While communications can be arbitrarily delayed or even omitted altogether, the consensus mechanism must be designed to accommodate this.

Definition 1 (**Network model**) The parameter by which nodes communicate with each other is called the network model. Nodes communicate by sending messages through Gossip protocol and form a P2P network topology.

The adversary's ability to delay messages is limited by the network model. Networks can be synchronous, asynchronous or partially synchronous.

– *Synchronous*: In synchronous networks, the maximum network delay of message propagation delays between nodes has a predetermined upper bound known to the participants in which messages are sent to all nodes. In semi-synchronous, the message delivery time can be defined by a random variable whose probability distribution is known to participants.

– *Partially-synchronous*: In this network, there is a predetermined maximum network delay that is unknown to network participants. However, they are certain that all messages will eventually reach their intended recipients. There is also an unknown global stabilization time (GST) in a second scenario, which ensures that all communications between two honest nodes arrive on time after GST (the network will then become synchronous). Partial synchrony provides good adaptability to the real network dynamics, while also simplifying network modeling.

– *Asynchronous*: This is most challenging scenario to reach consensus. There is no maximum network delay in this configuration. That is, communications could take an unlimited amount of time to reach their intended receiver.

2.2.3 Data model

Different blockchains adopt different data models for their states. For example the states of Bitcoins are coins and modeled as Unspent Transaction Output (UTXO). The two most common models:

1. UTXO model: In this model transaction spends output from prior unspent transactions and generates new outputs that can be spent in the future. The unspent transactions are kept in each node; which consists of transaction outputs that have not considered inputs to another transaction. A higher level of privacy is provided by UTXO. In this paper we focus on UTXO model.

2. Account/Balance Model: More contemporary blockchain-based systems, such as Ethereum (Buterin, 2017) and Hyperledger (Hyperledger Architecture Volume 1, 2021), allow smart contracts to modify general states arbitrarily; the model where consensus takes place upon state (result). Furthermore, the Account/Balance model is more efficient because each transaction just needs to verify that the sending account has sufficient funds to complete the transaction. State replication completely overcomes the concept of transaction input and output. More precisely, blockchain state is an outcome of a transaction. The Balance model keeps track of the balance of each account as a global state. The balance of an account is checked to make sure it is larger than or equal to the spending transaction amount. Once a transaction is executed the states of the accounts involved in the transferring are updated.

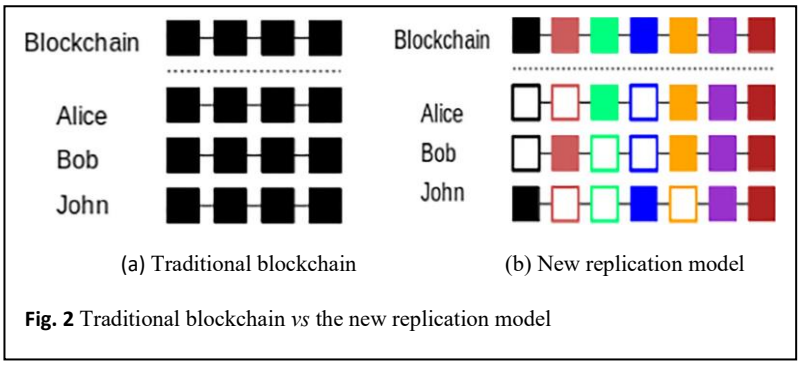


Fig. 2 Traditional blockchain *vs* the new replication model

2.2.4 Block structure

In bitcoin, a transaction is a transfer of bitcoins from one or more source accounts to one or more destination accounts. In essence, an account is a public/private key pair. The account is identified by using an address obtained from the public key. To send bitcoins to a specific account, a transaction is produced using account's address as a destination. To send bitcoins from one account to another, a transaction must be signed using the sending account's private key. Generally, a block stores transactions and global states.

The block header contains the following fields.

- Previous Block - it is reference to the previous block in the chain
- Nonce - it is related to the proof of work,
- Merkle root - it enables efficient verification of transactions and states (a block stores transactions)
- A set of metadata related to the mining protocol; difficulty and timestamp.

2.2.5 Blockchain layers

There are several components in a blockchain system which are categorized as follows (Ferdous et al., 2021).

- Meta-Application Layer. Its aim is to provide an overlay on top of the application layer
- Application Layer. It defines a semantic interpretation of a blockchain system. For example, in the cryptomoney blockchain, it defines a crypto-currency and then set up protocols in order to determine, how such a currency can be exchanged between different parties.
- Consensus Layer. It manages distributed consensus mechanisms that allow distributed systems (networks of computers) to work together and stay secure.

- Network Layer. It is handling network functionalities, for instance, protocol network as well as propagating and receiving transactions and blocks in the P2P network.

2.2.6 Consensus

A common order on transactions must be agreed between the nodes in order to ensure consistency of the nodes participating in the blockchain. In a distributed system, deciding on a common order is not easy. This is an agreement used to verify correctness of blockchain, which is related to the order and timing of a block. 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, e.g. bitcoin overcomes this challenge by tentatively committing transactions before synchronizing at regular intervals and broadcasting a block created by one of the nodes (Al-Saqqa & Almajali, 2020; Ferdous et al., 2021; Du et al., 2021). A block b contains collection of transactions $T\ x$ that the node that produced the block has committed since the previous block. The block is subsequently delivered to all nodes in the network. Each receiving node rolls back any tentatively committed transactions from the previous block and applies the transactions from the current block. At this time, all of the nodes have agreed that all of the transactions in the block are valid. Transactions committed as part of the block are confirmed, so they do not need to be applied again. The rolled-back transactions will be validated once more and reapplied on top of the new base state. Transactions that are now invalid due to conflicts with other transactions in the block are rejected.We show the major consensus algorithms in Table 1. Unlike, our proposed SecuSca, majority of existing distributed consensus algorithms are not handling security and scalability at the same time.

2.2.7 Blockchain Characteristics

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 in terms of processing data and increased storage space which in turn 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 - each node uses this key 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 (Pinar Ozisik et al., 2017); where t is the number of blocks 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 unrecognizably different, which secures the blockchain.

Table 1 Main distributed consensus algorithms in blockchain

Algorithm	Description
Practical Byzantine fault tolerance (PBFT.)	Replication algorithm able to tolerate Byzantine fault. it works in asynchronous environments. But it does not scale (Castro & Liskov, 1999)
Proof of Work(PoW)	The blockchain validators must take data from a block header as an input, and continuously run it through a cryptographic hash function (Kiayias & Zindros, 2018).
Proof Of Stake (PoS)	Blocks are validated based on the stake of the network participants (Fan & iching, 2017).
Delegated Proof-of-Stake (DPoS)	Users of the network vote and elect delegates to validate the coming block (Wang et al., 2020).
Proof of Authority (PoA)	It is based on the reputation of trusted parties in a blockchain network. It provides high performance and fault tolerance (Ekparinya et al., 2020)

2.2.8 Sharding Blokchains

Sharding aims to split nodes into different partitioning (shards) that process transactions simultaneously, while- maintaining consistency. In a sharded blockchain, (honest or malicious) nodes are partitioned into various shards, as shown in Fig. 3. Each shard stores its own blockchain, which contains transactions. A transaction might involve a single shard (an intra-shard transaction) or multiple shards (a cross-shard transaction).

Communication, storage, and computational overhead are three primary resources of traditional blockchains that might prevent system from scaling up. All nodes incur such overhead due to data replication and established consensus, which requires a large mass of message exchange. However, this can be avoided if each node participates in sharding, as each node in a shard only supports part of the total blockchain.

The following are the main steps in sharding with orthogonal functionalities: (1) Shard formulation: It involves two processes. First, the ledger is divided into distinct shards. Second, nodes are assigned to different shards in different ways based on verification methods being applied (shard allocation). (2) An intra-shard transaction: It describes how each shard processes local transactions; different protocols utilize different consensus, e.g., BFT, POW, PBFT, etc. (3) A cross-shard protocol: It defines how shards process cross shard transactions, which is only guaranteed by certain protocols. Concurrency Control and Atomic Commit both capture the cross-shard layer (Han et al., 2021). The sharded blockchain has four layers (Han et al., 2021): (1) *data layer* defines format of the ledger (2) *membership layer*, ensures that nodes are partitioned into shards (3) *intra-shard layer* -transactions are validated inside the shard (4) *cross-shard layer* - transactions are validated between the shards.

3 The Proposed Dynamic Sharding Approach

In this section, we first discuss the architecture of the proposed dynamic sharing approach, SecuSca. We then describe the process involved in the dynamic sharding approach that preserves balance between security and scalability.

3.1 Architecture of SecuSca

An architecture of the proposed SecuSca approach is depicted in Fig 4. 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 maximum security and scalability of the blockchain by decreasing its full replication, SecuSca creates a tradeoff between security and sclability. The process comprises two steps that operates at the same time. An optimization function is introduced in order to help the sharding process:

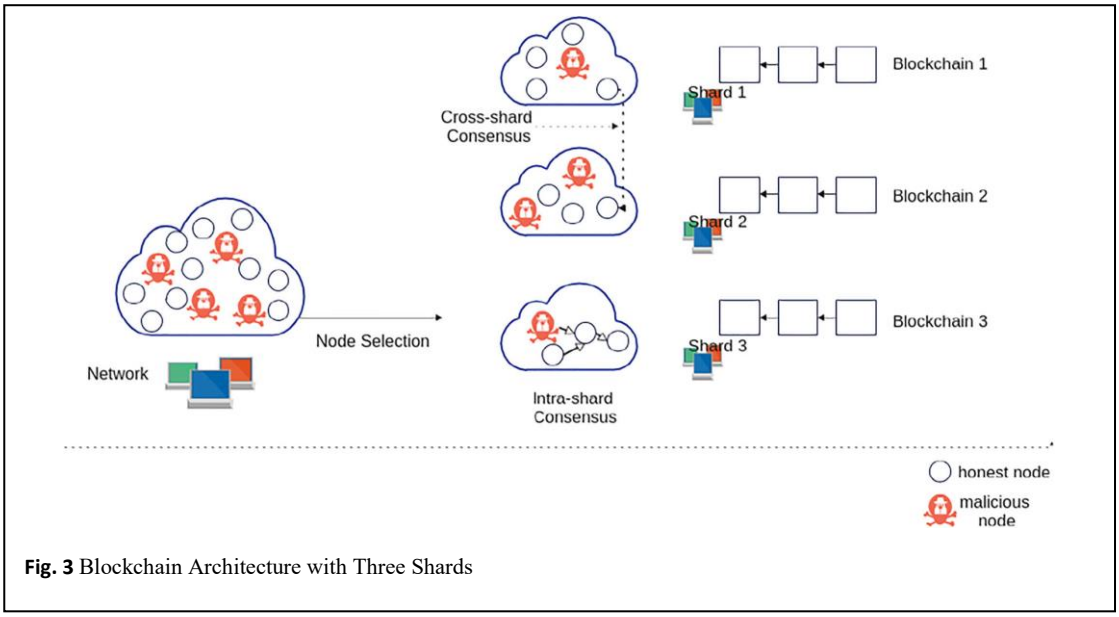


Fig. 3 Blockchain Architecture with Three Shards

– *Efficient replication*: The replication means efficient storage and security. This approach distributes and replicates global state of the blockchain to some nodes of the network by dynamic sharding but 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 a node has poor memory storage it can participate in the protocol process.

– *Efficient reduction*: During the process of replication of an inserted block, a reduction step is operating in order to reduce the replication of some blocks over the network. This process then scales up the blockchain. This also allows more storage in network of the blockchain.

3.2 The process of SecuSca Approach

The efficient replication and reduction steps, involved in the proposed SecuSca approach, are explained as follow.

Step 1: **Replication model:** In SecuSca, we are interested in storing transaction's history. Unfortunately, all nodes cannot store the whole state due to scalability reasons. The blockchain is distributed over a higher number of nodes that store blocs, b_i (where i = {1, ...,B}). Each node is represented as n_j; where j = {1, ...,N}. The blocks are propagated throughout the network and stored on nodes, n, (n <= N). A node may not even store any state and can still participate in transaction validation 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, security of the blockchain is guaranteed by honest nodes. Replication of 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 this should be carried out without replicating them 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 as the size is small. It then decreases as the size increases.

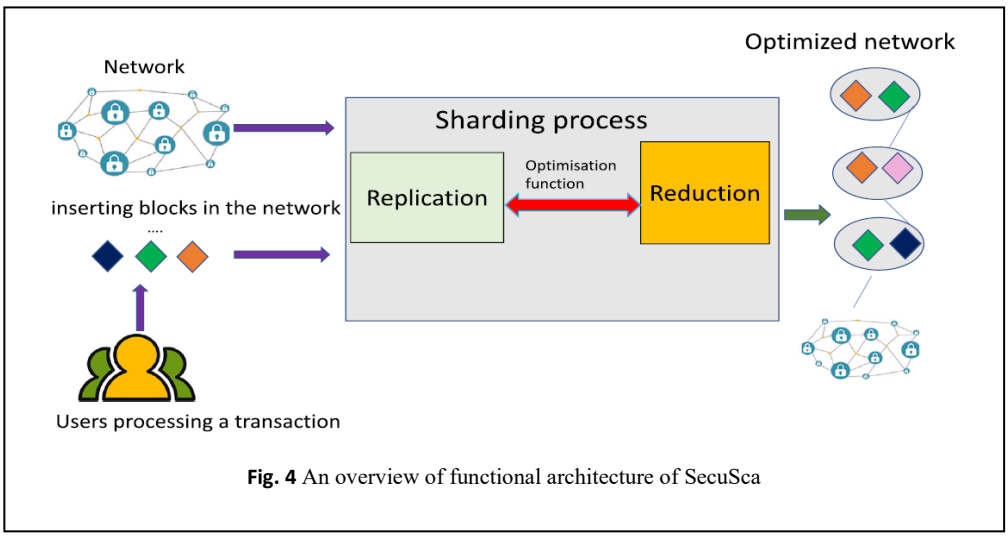


Fig. 4 An overview of functional architecture of SecuSca

Step 2: **Reduction Model.** In SecuSca, reducing block replication does not mean removing the entire block. Instead, it only reduces replication of explicit transactions in each block and keeps the header of all blocks in the blockchain. Transactions are explicitly represented in the block when it is newly added. For each new block added to the blockchain, the transactions it contains are only confirmed when a specific size constitutes other blocks after that new block. During this phase, the block is entirely stored. Indeed, transactions are proportional to the load of the network. Forcing a node to store everything would affect scalability. When transactions of block are confirmed by the network and that block is buried in the blockchain, the transactions are deleted.

Cryptographically, blocks buried in the blockchain are more secure. As the size t increases, the replication of the old blocks decreases. The last blocks added to the blockchain are not reduced because they are less secure. For each new block, b_i , added to the blockchain with a high replication, the approach selects a replication of all previously confirmed blocks in the blockchain on all nodes. Each node frees memory space by erasing the transactions of these blocks until reaching a minimal replica. The replication then becomes constant, even if the blockchain size increases. SecuSca allows availability of the overall state and keeps a minimum replica of the entire blocks distributed on nodes. Every node stores part of the blockchain state (a subset of all history) and header blocks, including Merkle roots for the whole blockchain.

4 The Dynamic Sharding Calculus

This section describes the calculus and the process of inserting a new block into the blockchain. When a new block is ready to be inserted in the blockchain and is replicated over the network, the sharding process follows an optimization function R. In the following, we define and explain such an optimization function (R) in order to ensure trade-off between replication for security and scalability of the blockchain.

4.1 Dynamic Sharding Function

In order to define and explain the optimization function, we introduce some useful definitions, which are used in the setting of such 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 a specified block. It is formally defined as:
Definition 2 (Block depth d). Consider, t , as the size of the blockchain and ,i, as 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 3 (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 maximum 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 4 (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 an estimated highest depth of a block from which we can not go under to stop reduction of 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 increasing 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 5 (The sharding function 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 the network is defined as follows:

$$\mathcal{R}(d) = \begin{cases} \alpha_N & \text{if } d < \gamma_0 \\ \frac{\alpha_0 - \alpha_B}{\gamma_B - \gamma_0} * (d - \gamma_B) + \alpha_0 & \text{if } \gamma_0 < d < \gamma_B \\ \alpha_0 & \text{if } d > \gamma_B \end{cases} \quad (1)$$

Input: $B = \{b_1, b_2, ..., b_n\}$: a set of blocks in the blockchain B.
$N = \{n_1, n_2, ..., n_N\}$: set of nodes in the network.
b_{n+1}: the next block, d : the depth of the blockchain in a node, Q: the Quorum
Output: NR_j: The number block replication in the blockchain

1: **For each** bj *where* $j \geq n+1$
2: **//step1:** *Calculate the number of replications to ensure the security*
3: $NR_j \leftarrow \mathcal{R}(d)$
4: **//step2:** *Verification of the Quorum agreement*
5: **While** $Verfcation(Q, NR_j) = False$
6: Go to step 1
7: $return\ NR_j$

Algorithm 1 Replication-validation.

4.2 The Validation Algorithm

This section illustrates the validation process when a block is replicated over the network. We devise an Algorithm 1 that presents the main steps involved in the process.

Step 1. *Step 1. Entry of a new block:* When a new block is entering in the blockchain, the sharding function *R(d)* will be triggered in order to calculate replication of the new block over the network to ensure scalability and security.

Step 2. *Validation by the Quorum:* The insertion of the block in the blockchain and its replication should be validated by the quorum *Q* over the network *N*. The block will not be inserted until the quorum is reached.

5 Related Work

This section reviews existing work and provides insights into related methods and solutions. The most popular blockchain systems, such as Bitcoin and Ethereum, have a full replication process that cannot manage a large number of transactions. This is because, with an increase in number of transactions, systems become overloaded, which make them difficult to scale. Efficiency of such systems also declines as more nodes join the network. Some solutions (e.g., on-chain and off-chain) have lately been emerged in order to overcome this issue, i.e., to enable blockchain to work even with the growth in number of transactions. Off-chain scalability options may come with their own set of security problems. The decentralization and security of blockchain networks should not be compromised. However, in order to maintain the security of blockchain systems under the continuous growth of transactions and blockchain architecture, scalability features must be maintained.

In off-chain blockchains, transactions are executed off-chain in order to overcome scalability limitation in blockchain systems. Untrusted peers can build direct payment channels on the Lightning network (Poon & Dryja, 2008). This allows them to make off-chain micropayments without committing each transaction to the underlying blockchain. A payment channel is made up of a blockchain-based contract that stores funds of peers involved in a transaction. Signatures are used in micropayments to ensure that peers accept a transaction. Furthermore, hashlocks and timelocks are used in micropayments to ensure that a malicious node does not take advantage of a cooperative participant.

In addition to the above, on-chain blockchains are proposed in order to address scalability issue. In on-chain, sharding is commonly employed in databases. The aim is to improve scalability of a traditional blockchain. This section reviews current approaches based on their basic qualities, and examines operational specifics of each method from several perspectives. It also analyzes potential challenges of each solution, such as security, throughput, and latency as shown in Table 2.

In a sharding blockchain system, number of nodes, N, in a network are divided into committees, K (*shards*), with a small number of nodes, c, which are responsible for managing their own blockchain, and replication of $c = N/K$. This results in smaller full replication systems. Each committee K keeps just validated blocks and does not handle the whole blockchain ledger. The purpose is to eliminate overhead of duplicating communication, storage, and computation in each full node.

The work presented, in (Luu et al., 2016; Kokoris-Kogias et al., 2018; Zamani et al., 2018), is based on shardingbased Proof of Work and Byzantine fault tolerance (BFT). Authors in (Luu et al., 2016) proposes, *Elastico*, which is the first sharding-based public blockchain that is capable of tolerating byzantine adversaries. It partitions the network into shards and ensures probabilistic correctness by randomly assigning nodes to committees. In it, each shard is verified, in parallel, by a disjoint committee of nodes.

Table 2 Comparison between existing sharding blockchains in academia and industry based UTXO model and Account model

	UTXO			Account		
System	Elastico	Omniledger	Rapidchain	Ethereum 2.0	Monoxide	Zilliqa
Intra-Shard Consensus	PBFT	BFT (BizCoinX)	Sync BFT	BFT	POW	PBFT
Adversary Model	$\leq \frac{1}{4}$	$\leq \frac{1}{4}$	$\leq \frac{1}{3}$	$\leq \frac{1}{4}$	$\leq \frac{1}{2}$	$\leq \frac{1}{4}$
Cross-shard Consensus	-	2PC	Split	RT	RT	-
Performance	$O(n^2)$	$O(n)$		$O(1)/O(n)$	$O(n)$	$O(n)$
Decentralized	PBFT	BFT	Sync BFT	BFT	POW	PBFT
Security	$3f+1$	$3f+1$	$2f+1$	$3f+1$	$2f+1$	$3f+1$
(Fault tolerance)	33%	33%	50%	33%	50%	33%
Scalability	No	No	Yes	Yes	No	No

[a]RT: Relay Transaction

[b]The notation "-" means that the property does not apply to the system

[c]f is the number of admissible failures

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 blockchain ledger but cross-shard transactions are not supported. In addition, running PBFT among hundreds of nodes decreases protocol's performance. But reducing the number of nodes within each shard increases failure probability. The network can only tolerate up to 25% of malicious nodes.

The *Omniledger* (Kokoris-Kogias et al., 2018) is designed in order to improve upon *Elastico*. It includes new methods for assigning nodes to shardswith a higher security guarantee, as *Elastico*. It uses both Pow and BFT as atomic protocols for across-shard transactions (Atomix). The intra-shard consensus protocol of OmniLedger uses a variant of ByzCoin (Kokoris-Kogias et al., 2016) 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 (Luu et al., 2016).

Cross-shard in *Rapidchain* (Zamani et al., 2018) relies on an inter-committee routing scheme which is based on the routing algorithm of Kademlia(Maymounkov & Mazi`eres, 2002). It tolerates up to 33% of total resiliency and 50% of committee resiliency. *Rapidchain* also supports crossshard 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 (ZILLIQA team & et al, 2017; Harmony team, 2017; Amiri et al., 2019; Dang et al., 2019; Amiri et al., 2019). Though sharding improves storage and throughput, *K* (shards) increases linearly with *N* (nodes) with a low-security level. This could lead to malicious node errors.

The *Vault* approach (Leung et al., 2019) introduces fast bootstrapping that allows new participants to join the network without downloading the whole blockchain, thus reducing the transmitted state. *Vault* is Account-based for Algorand (Chen & Algorand, 2019), and does not require all nodes to store the whole blockchain state. In (Lu et al., 2020), authors propose a superlight client design that allows a light client to relay full nodes to read blockchain with a low read cost so as 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. In contrary, the proposed SecuSca approach does not replicate the blocks over the entire network.

Moreover, according to the results reported in [56], [57], it is critical that a sharding mechanism can handle crossshard verification and cross-shard transactions for validators assigned to distinct shards. This shows that the probability of cross-shard transactions reaches 100 percent as the total number of shards increases. Maintaining a separate global root chain could be one solution to verification. However, it does not permit cross-shard transactions without the use of an additional mechanism, such as lock/unlock operation in synchronous networks or lock-free operation in asynchronous networks. The demand for a safe cross-shard transaction protocol progressively outweighs the desire for a naive

mechanism that does not permit cross-shard transactions; even it can achieve a high improving factor N. In summary, most of the approaches being reviewed are focused only on storage issue. They fall short of simultaneously handling security and scalability issues, as addressed in our SecuSca approach.

6 Simulation and discussion

6.1 Experimental Setup

In order to validate our proposed SecuSca to ensuring a trade-off between security and scalabilit, we conducted simulation using python on a server with 75GB of RAM and two Intel Xeon E5-2650 v4 2.2GHz CPUs..We evaluate our approach and compare it with traditional bitcoin blockchain. We conduct multiple 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* α_N and *lower bound* γ_0. We then define the size of the whole blockchain sharded over the network.

6.2 Simulation Results

The experiments studied two main aspects which include: the efficiency in terms of block replication; and the size of the blockchain.

– **Replication of blocks**

When a miner produces a new block, it is broadcasted over the network and is added to the local storage of nodes. First, we simulate the function *R(d)* with 100 nodes with 200 GB storage capacity as shown in Fig. 5(a) and (b). Each node performs the sharding optimization function *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 (α = 0.5) and (γ = 0.5), and then (α = 0.7) and (γ = 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 experiment is conducted using 200 nodes with a storage capacity of 100 GB. The results are shown in Fig. 5 (c) and (d). The replication of each block depends on its depth in the blockchain. Replication 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 is to 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 (α_O = 15).

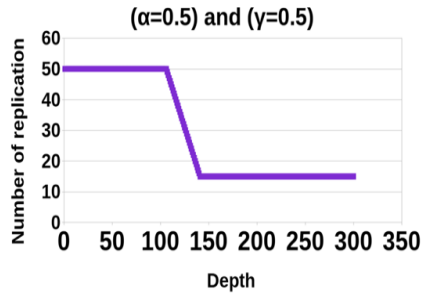
– **Size of the network**

In the experiment that follows, we run the algorithm over 100 nodes that produce 300 blocks of 1*MB* size. Fig. 6 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 more significant in traditional blockchain. The size of SecuSca is the sum of all blocks stored on each enode, i.e., from the first block to the last block. The sum of all transactions of all shards is shown in blue (or SecuSca legend). From [0-200], every block is highly replicated. The size also grows linearly in this range. After that, nodes start reducing replication until the lower bound is reached.

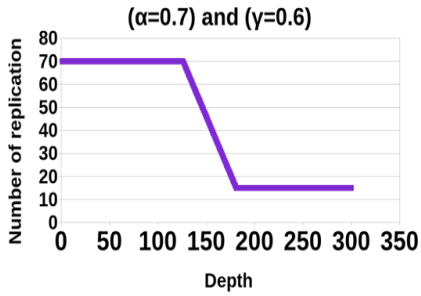
The size *t* of the shard in each node *n* at different steps is given by:

$$t_n = \sum_{i=0}^{b} \frac{R(i,t)}{N} \qquad (2)$$

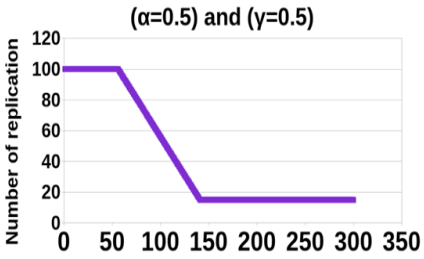
In the above equation, *N*, is the number of nodes, *i* is the position of a block b_i is in the chain and *t* is the size of the blockchain.



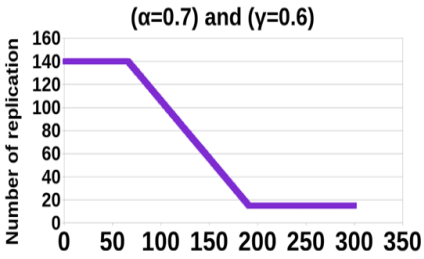
(a) 100 nodes with 200 GB memory



(b) 100 nodes with 200 GB memory



(c) 200 nodes with 100 GB memory



(d) 200 nodes with 100 GB memory

Fig. 5 The replication of blocks according to block depth

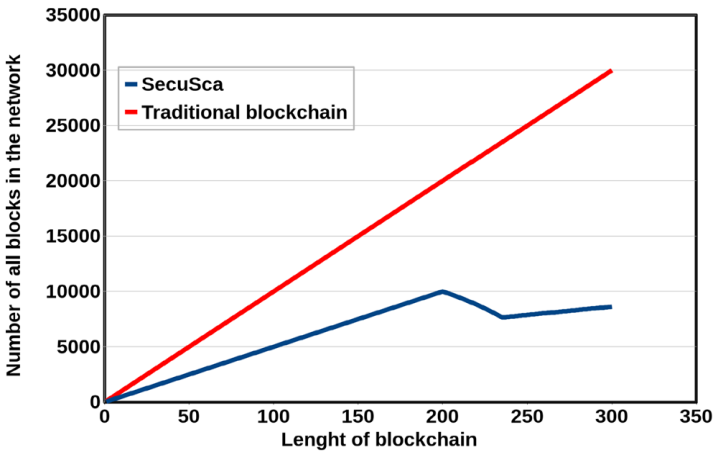


Fig. 6 Comparison of the size of blockchain between traditional blockchain and SecuSca

– Discussion

This section evaluates and analyzes the effectiveness of SecuSca approach. It is clearly observed that our approach has designed an optimal function for blockchain that ensures an appropriate tradeoff between scalability security of blockchain. The analytical and numerical results show, how our SecuSca approach, promotes scalability, and enables users to store more transactions by freeing up local disk. However, certain aspects of 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 also be part of the consensus protocol.

7 Conclusion

Security and scalability are two crucial factors in the successful implementation and operation of blockains. Existing research either focuses on security or a scalability of blockchain but they fall short of taking both into account when designing blockchain.

In this paper, we have designed a new approach, SecuSCa, for blockchain which makes an effective tradeoff between security and scalability. SecuSca allows for more capacity in terms of storage in the whole blockchain. We developed a dynamic sharding approach which is formulated as an optimisation problem that preserves security and scales up the blockchain. We have conducted various simulation-based experiments. This experiments have shown positive results and significant improvement over traditional blockchain, bitcoin. The analysis shows how our proposed approach promotes scalability and enables users to store more transactions by freeing up local disks while ensuring security of the blockchain. Nevertheless, SECUSCA needs further improvements, for instance, it should allow the blockchain to keep working even when some nodes are saturated.

References

Al-Saqqa, S., & Almajali, S. (2020). Blockchain technology consensus algorithms and applications: a survey. *Int. J. Interact. Mob. Technol.*, *14*(15), 142–156.

Amiri, M. J., Agrawal, D., & Abbadi, A. E. (2019). On sharding permissioned blockchains. In *IEEE International Conference On Blockchain, Blockchain 2019, Atlanta, GA, USA, 14-17, July 2019, pp. 282–285. IEEE*.

Amiri, M. J., Agrawal, D., & Sharper, A. E. A. (2019). Sharding permissioned blockchains over network clusters. CoRR, arXiv:1910.00765.

Buterin, V. (2017). Ethereum, white paper. Accessed on 20 May 2021. https://ethereum.org/en/whitepaper/.

Castro, M., & Liskov, B. (1999). Practical byzantine fault tolerance. In M. I. Seltzer, & P. J. Leach (Eds.) *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI), 22-25, February 1999, pp. 173–186. USENIX Association, New Orleans, Louisiana, USA*.

Chen, J., & Algorand, S. M. (2019). A secure and efficient distributed ledger. *Theoretical Computer Science*, *777*, 155–183.

Dang, H., Dinh, T. T. A., Loghin, D., Chang, E.-C., Lin, Q., & Ooi, B. C. (2019). Towards scaling blockchain systems via sharding. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, 30 June- 5 July, 2019, pp. 123–140. ACM, Amsterdam, The Netherlands*.

Du, M., Chen, Q., Chen, J., & Ma, X. (2021). An optimized consortium blockchain for medical information sharing. *IEEE Trans Engineering Management*, *68*(6), 1677–1689.
Ekparinya, P.arinya., Gramoli, V.incent., & Jourjon, Guillaume (2020). The attack of the clones against proof-of-authority. In *27th annual network and distributed system security symposium, NDSS 2020, San Diego, California, USA, 23-26, February 2020. The Internet Society*.

Fan, L.ei., & iching, H.-S. Z. (2017). A scalable proof-of-stake blockchain in the open setting (or, how to mimic nakamoto's design via proof-of-stake). IACR Cryptol. ePrint Arch.:656.

Ferdous, Md. S., Chowdhury, M. J. M., & Hoque, M. A. (2021). A survey of consensus algorithms in public blockchain systems for crypto-currencies. *J. Netw Sadek Comput. Appl.*, *182*, 103035.

Han, R., Yu, J., Lin, H., Chen, S., & Ver´ıssimo, P. J. E. (2021). On the security and performance of blockchain sharding. IACR Cryptol. ePrint Arch.:1276.

Harmony team (2017). Harmony. Technical White paper, Accessed 16 May 2021. https://harmony.one/whitepaper.pdf.

Hyperledger Architecture Volume 1 (2021). Introduction to Hyperledger business blockchain design philosophy and consensus. https://www.hyperledger.org/wp-content/uploads/2017/08/ Hyperledger Arch WG Paper 1 Consensus.pdf. accessed on 20 May 2021.

Ioini, N. E., & Pahl, C. (2018). A review of distributed ledger technologies. In H. Panetto, C. Debruyne, H. A. Proper, C. A. Ardagna, D. Roman, & R. Meersman (Eds.) *On the Move to Meaningful Internet Systems. OTM 2018 Conferences - Confederated International Conferences: CoopIS, C&TC, and ODBASE 2018, Valletta, Malta, 22-26 October, 2018, Proceedings, Part II, vol. 11230 of Lecture Notes in Computer Science, pp. 277–288. Springer*.

Kiayias, A., & Zindros, D. (2018). Proof-of-work sidechains. IACR Cryptol. ePrint Arch.:1048.

Kokoris-Kogias, E., Jovanovic, P., Gailly, N., Khoffi, I., Gasser, L., & Ford, B. (2016). Enhancing bitcoin security and performance with strong consistency via collective signing. CoRR, arXiv:1602.06997.

Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Syta, E., & Omniledger, B. F. (2018). 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, pp. 583–598. IEEE Computer Society*.

Leung, D.erek., Suhl, A.dam., Gilad, Y.ossi., & Vault, Nickolai Zeldovich (2019). Fast bootstrapping for the algorand cryptocurrency. In *26Th annual network and distributed system security symposium, NDSS 2019, san diego, california, USA, 24-27 February 2019. The Internet Society*.

Lu, Y., Tang, Q., & Wang, G. (2020). Generic superlight client for permissionless blockchains, CoRR, arXiv:2003.06552.

Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., & Saxena, P. (2016). A secure sharding protocol for open blockchains. In E. R. Weippl, S. Katzenbeisser, C. Kruegel, A. C. Myers, & S. Halevi (Eds.) *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 24-28 October 2016, pp. 17–30. ACM, Vienna, Austria*.

Lux, Z. A., Thatmann, D., Zickau, S., & Beierle, F. (2020). Distributed-ledger-based authentication with decentralized identifiers and verifiable credentials. In *2Nd conference on blockchain research & applications for innovative networks and services, BRAINS 2020, 28-30 September 2020, pp. 71–78. IEEE, Paris, France*.

Maymounkov, P., & Mazi`eres, D. (2002). Kademlia: a peer-to-peer information system based on the XOR metric. In P. Druschel, M. F. Kaashoek,&A. I. T. Rowstron (Eds.) *Peer-to-Peer Systems, First International Workshop, IPTPS 2002, 7-8 March 2002, Revised Papers, vol. 2429 of Lecture Notes in Computer Science, pp. 53–65. Springer, Cambridge, MA, USA*.

Nakamoto, S. (2008). Bitcoin: a peer-to-peer electronic cash system. White Paper.

Pinar Ozisik, A., Bissias, G., & Levine, B. N. (2017). Estimation of miner hash rates and consensus on blockchains (draft). CoRR, arXiv:1707.00082.

Poon, J., & Dryja, T. (2008). The bitcoin lightning network: scalable off-chain instant payments. White Paper.

Wang, B., Li, Z., & Li, H. (2020). Hybrid consensus algorithm based on modified proof-of-probability and dpos. *Future Internet*, *12*(8), 122.

ZILLIQA team, & et al (2017). The zilliqa. Technical White Paper, vol. 16.

Zamani, M., Movahedi, M., & Raykova, M. (2018). Rapidchain: scaling blockchain via full sharding. In D. Lie, M. Mannan, M. Backes, & X. F. Wang (Eds.) *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, 15-19 October 2018, pp. 931–948. ACM, Toronto, ON, Canada*.