Internet of Things 24 (2023) 100955

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Distributed scalability tuning for evolutionary sharding optimization with Random-equivalent security in permissionless Blockchain

Hamza Baniata [a,*], Ahmad Anaqreh [b], Attila Kertesz [a]

[a] Department of Software Engineering, University of Szeged, Hungary
[b] Department of Computational Optimization, University of Szeged, Hungary

ARTICLE INFO

Keywords:
Blockchain
Optimization
Security
Sharding
Scalability
Genetic algorithms

ABSTRACT

In spite of multiple advantages of the adoption of blockchain (BC), it still faces some integration challenges in modern applications, such as the Internet of Things. These challenges include low throughput rates in permissionless settings. To solve this challenge, several state-of-the-art works proposed sharding (i.e., partitioning) the BC infrastructure. Sharding the network into smaller shards improves the total system throughput, regardless of the node-shard assignment criteria. Most previous work applied a Random Sharding (RS) approach, i.e., randomly allocating nodes into predefined shards, to satisfy the required unpredictability property of node-shard allocation. In this paper, we propose a Blockchain Optimized and Secure Sharding (BOSS) protocol that aims to optimize the node-shard allocation resulting in increased throughput using a variant of the evolutionary Genetic Algorithm. The RS-equivalent levels of security and unpredictability are guaranteed by deploying a distributed random tuning mechanism for the intra-shard weight. We designed BOSS as an extension of the well-defined RS-based RapidChain protocol. We show that the proposed methods can be adapted to other sharding protocols that originally used RS techniques. We implemented and tested our protocol with more than 362,880 cases that covered seven configurable system and optimization parameters. Our evaluation revealed ≈17% average enhancement in scalability, along with a negligible <0.5% mean absolute difference in security levels. To the best of our knowledge, this is the first work that optimizes inter- and intra-shard scalability, with publicly verifiable solutions in permissionless BCs, while maintaining RS-equivalent security and unpredictability.

1. Introduction

Blockchain (BC) is a Distributed Ledger (DL) technology that is used to replicate, share, and synchronize data spread over different geographical locations such as multiple sites, countries, or organizations [1]. The permissionless category of BCs implies a main property of no central administrator or data storage mechanism. Starting in 2009, several variants of a reliable consensus algorithm were proposed to govern this peer-to-peer decentralized BC network, namely Proof of Work (PoW) [2]. Numerous benefits and applications of BCs have promoted their popularity among a broad spectrum of businesses, including governance, industry, decision-making processes, management, the Internet of Things (IoT), information security, and energy [3]. The advantages of BC utilization in these businesses include reliability, complete and secure decentralization, traceability, and immutability [4]. However,

* Corresponding author.
E-mail address: baniatah@inf.u-szeged.hu (H. Baniata).

https://doi.org/10.1016/j.iot.2023.100955
Received 15 June 2023; Received in revised form 22 August 2023; Accepted 23 September 2023
Available online 25 September 2023

the adoption of BC-related technologies also posed several challenges, related to, for example, scalability [5] and latency [6]. Specifically, the throughput of a BC system, defined as the number of processed transactions per second (TX/s), is far from current practical requirements. Thus, throughput has become a crucial limitation that prevents BC from being adopted in applications that require real-time response [7].

Sharding is a type of database/network partitioning technique that separates a very large set of operating elements into smaller, faster and more easily managed parts called shards [8]. These shards process disjoint sets of data in parallel. The very act of partitioning the network into such smaller shards results in a higher throughput level of the network compared to its non-sharded version. That is, each node does not have to store and validate the entire BC, but only a subset of it. Thus, sharding can reduce network congestion and increase transaction speed, as well as lower the storage and computational requirements for each node. The increasement in throughput and scalability levels is guaranteed, regardless of the node criteria used during its assignment to its shard(s).

Since, by definition, permissioned BCs consist of trusted parties, sharding can be easily adopted/integrated in such systems. These trusted parties are expected to perform data/node partitioning, propose trusted data configurations, and act as PKIs. In permissionless BCs, however, sharding is not an easy task to do, as several issues challenge its integration. For instance, untrusted leaders, or groups of nodes, must be anonymously elected in a costly distributed fashion. The election methods deployed must be formally proven correct and safe in a strictly challenging trustless environment. On the other hand, objectives, including communication costs, should be considered during BC sharding as they would increase the total scalability [9].

Few proposals could prove the applicability of proper methodologies providing secure *and* scalable BC-based permissionless systems. Specifically, interesting yet highly complex hybridization approaches (e.g. of different consensus mechanisms, each of which is utilized for a different layer of data) were proposed. The computational or communications costs associated with the identity establishment phase in permissionless BC sharding protocols play an important role in calibrating the security-scalability trade-off. In particular, this leads to either overly unscalable or insecure systems [10].

Since sharding provides higher scalability regardless of node-shard assignment criteria [11], Randomized Sharding (RS) was used by all previous works that addressed sharding in permissionless BCs (e.g. Elastico [12], Omniledger [13], RapidChain [14] and Ethereum 2.0). That is, RS was found to be the most secure approach against adaptive adversaries, potentially performing Sybil-attacks [13,15]. This is because RS satisfies the critical needed unpredictability property.

The aforementioned scalability and energy efficiency needs can be neglected when using unrestricted (i.e., in terms of e.g. computational or communications costs) BC infrastructure. That is, high security guarantee with non-optimized scalability is usually considered more acceptable than lower security guarantees with optimized scalability. This is true as long as the system can tolerate high computational and/or communications costs. However, in the case of real-time application requirements (e.g. Internet of Vehicles (IoV)) or very limited computational abilities (e.g. IoT), scalability and network optimization methods must be considered.

Genetic Algorithm (GA) is a type of evolutionary computation that uses natural selection and genetic operators to search for optimal solutions in a given problem domain. Genetic algorithms can be applied to various types of problems, such as optimization, feature selection, machine learning, and artificial intelligence. However, GA faces some challenges when dealing with large and complex problems, such as high computational cost, slow convergence, premature convergence, and loss of diversity. These challenges can limit the scalability of GAs and affect their performance and quality of solutions. Therefore, researchers have proposed various techniques and strategies to improve their scalability, such as parallelization, distributed computing, hybridization, niching, elitism, adaptive parameters, and problem decomposition.

In this paper, we address these complexities in permissionless BCs targeting the provision of secure and scalable node-shard assignments. Specifically, we propose a novel Blockchain Optimized and Secure Sharding (BOSS) protocol as our main contribution, in which we implement distributed scalability tuning for a niched Pareto GA-based optimization. The designed distributed tuning mechanism of the GA variables results in secure assignments in terms of RS-equivalency, unpredictability, and public verifiability.

In particular, we collectively show that our proposed BOSS protocol provides:

1. RS-equivalent security (i.e., in terms of shard-level security and unpredictability of node-shard assignments),
2. Higher scalability (i.e., in terms of average intra-shard data propagation),
3. Public verifiability of node-shard assignments.

The remainder of this paper is organized as follows: Section 2 discusses state-of-the-art solutions that attempt to address sharding problems in permissionless BCs. Section 3 provides the necessary background information related to the methods we use, the formal definitions, problem statement, and threat models. In Section 4, we explain the proposed methods, including the niched Pareto evolutionary sharding optimization, the distributed scalability tuning mechanisms, and the experimental setup. After that, we present, analyze, and discuss the scalability and security results of our proposed protocol in Section 5. Finally, we conclude our work in Section 6.

2. Related work

Several state-of-the-art works utilized different sharding mechanisms in BC-IoT integrated systems. However, critical security limitations of those proposals can be found. For example, Wang et al. [16] proposed sharding satellite-based IoT devices and Gao et al. [17] proposed sharding Industrial IoT devices. Both works provided security analysis for double-spend attack tolerance, yet failed to show their tolerance against the shard compromisation attack (aka 1% Attack) [18]. This attack could be addressed for

permissionless BC-IoT integrated systems by Cai et al. [19] only with unrealistic assumption that adversary nodes are known prior to running the sharding protocol.

Halgamuge and Mapatunage [20] assumed a TTP was present for the IoT system to perform the sharding tasks, which deviates from the claimed permissionless BC-based system towards centralization. Nevertheless, the proposed sharding protocol was also not evaluated against the shard compromisation attack. Mehraein et al. [21] used PoW to determine the members of a parent shard that runs the sharding protocol. Using such an approach, the authors could guarantee the tolerance of the parent shard against the shard compromisation attack. However, for other shards, they proposed a reputation-based mechanism according to which, the nodes are identified as honest or adversary.

As detailed in the full details in [11], Elastico, Omniledger, RapidChain, and Zilliqa all satisfy allocation-randomness, as all nodes are shuffled for each epoch. Furthermore, all of these protocols assume no trusted component. However, Omniledger and RapidChain were found to be the most satisfying of the unbiasibility property. To validate our approach, we compare the proposed protocol with the RapidChain sharding protocol [14] to represent previously proposed RS-based protocols. The reason we specifically selected RapidChain from other available protocols is its usage of the Commensal Cuckoo rule [22] which does not satisfy the public verifiability property [11].

2.1. Randomized sharding in RapidChain

RapidChain [14] is a recently proposed solution that hybridizes an improved version of the communication-based PBFT [23] consensus algorithm and the computation-based PoW [2] consensus algorithm. Specifically, it uses an expected constant-round algorithm for Byzantine intra-shard consensus in a synchronous, authenticated network, claiming a tolerance of up to $\Psi_R = 1/2$ fraction of shard nodes to be corrupt. Furthermore, RapidChain provides a relatively high inter-shard fault-tolerance (up to $\Psi_N = 1/3$ fraction of its network nodes) with reduced communication costs and decreased bootstrapping complexities.

In RapidChain, a reference committee in the protocol partitions the set of all nodes at ***random***. This partitioning method (i.e., Randomized-Sharding (RS)) is claimed to guarantee a majority of honest nodes per shard, resulting in leaders (individually elected later in shards) being corrupt with a probability less than 1/2. However, this guarantee was not supported by evidence and was only claimed arbitrarily.

To address this issue, a reliable mechanism is proposed to solve the probably corrupt elected leaders. That is, randomly elected leaders of later iterations are asked to verify similar blocks previously proposed by their former leaders. Additionally, the intra-shard protocol was designed such that a Byzantine leader can prevent progress but cannot violate safety, meaning that some honest nodes might not terminate at the end of the iteration but all honest nodes who terminate in that iteration will output the same value, called the safe value. Those two mechanisms, indeed, solve a corrupt intra-shard leader issue even with no solid guarantee of an honest intra-shard majority. A side effect of these mechanisms has been reported to be a reduction in throughput by roughly half.[1]

In summary, RS is used in permissionless RapidChain BCs to achieve $\Psi_R = 1/2$ ***with high probability*** but ***not always***, which we have experimentally proven in [24]. However, there are ways around this limitation that would still guarantee the safety of the system, such as the measures described above. Since having an insecure shard is solvable but costly, it is preferred to use RS, which still gives a much better scalability compared to the non-sharded version. Meanwhile, RS provides the lowest probability of cases where costly safety maintenance measures are required.

We list below the main steps of the classical RapidChain protocol to be able to precisely indicate where BOSS modifies:

1. The reference committee generates fresh random numbers to be used as seeds of PoW-based new public IDs of nodes wanting to participate in the following epoch. Furthermore, fresh random numbers are used later by the reference committee itself as seeds to randomly allocate the final list of nodes into shards using the Cuckoo rule.
2. Once nodes receive the new random seed, they start working individually on their new PoW-based public IDs. If a valid PoW is found by the cutoff time, nodes send their new IDs to the reference committee. Otherwise they wait for the next seed which will be used to generate new PoW IDs for the next epoch.
3. Once a cutoff time is reached, the reference committee uses the available valid IDs to allocate the nodes into shards using the Cuckoo rule.
4. Once the node-shard allocation scheme is agreed on by the members of the reference committee, it is shared with all nodes using the inter-shard communication protocol.
5. Accordingly, shard members send their TXs (which have been submitted by end-users during the past epoch) to the Output Committee which is elected by all shards during the ID generation phase. The output committee is elected using another random number generated earlier by the reference committee specifically for this purpose.
6. The output committee generates a valid block after verifying all TXs within, and saves this new block into its ledger.
7. Once the block is added to the ledger, go back to step 1.

The Cuckoo rule states that when a new node wants to join the system, place it at a random $x \in [0, 1)$ and move all nodes in a unique region containing x to other random locations in [0, 1). Obviously, this rule randomly allocates new nodes to given shards and relocates other similar nodes from their current shards to other randomly selected shards. The specific technical details, on how this rule is adapted into RapidChain, are out of the scope of our research, as we suggest to partially eliminate it.

[1] See Section 4.2.3 in the full RapidChain paper [14]

Table 1
Key symbols used in our formulations.

Symbol	Description
$G(V, E)$	Undirected graph representing a network
V	Set of nodes in a graph G
N	Number on nodes in the set V
E	Set of edges in a graph G
w_{ij} or w_e	Weight on $e \in E$ connecting nodes $i, j \in G$
K	Faulty/Adversary nodes in G
Ψ	The upper bound fraction of faulty nodes tolerated by a named distributed system to maintain its security
R	Set of network shards
d	Number of shards in the set R
r_i	Shard number i
n_{r_i}	Nodes in the ith shard
k_{r_i}	Adversary nodes in the ith shard
α_{ri}	Min. no. of nodes in r_i
β_{ri}	Max. no. of nodes in r_i
λ	intra-shard weight importance
ρ	Pearson correlation coefficient

2.2. Optimum sharding using linear programming

Following the notations described previously, and the Mixed Integer Program (MIP) model proposed in [25], we can model the BC network sharding as an MIP problem with the objective of minimizing the total weights only at the intra-shard level. The mathematical formulas of this program are provided in Appendix A. We can also find other graph partitioning approaches that only targeted the inter-shard optimization, such as [26]. Formulated as such, we obtain standard LP problems that have been well studied and can be solved by, e.g., branch-and-bound type of algorithms [27]. However, solving this problem to achieve *optimal* solutions is much more costly as the problem is NP-Complete [28]. Furthermore, our goal is to provide slightly optimized, yet still readable as randomized solutions. That is, whenever optimum solutions are targeted by a given sharding protocol, the unpredictability property cannot be fulfilled, as an attacker can reverse-engineer the sharding algorithm.

3. Model

In this section, we elaborate on the preliminary information, from which we build up our proposal. Specifically, we give a brief formal description of the BC and Sharding foundations. Furthermore, we state our research problem and the accompanying threat models considered in the later discussion. To facilitate the comprehension of our manuscript, we define the key symbols we used in Table 1. Further notation will be defined and introduced whenever necessary along the manuscript.

3.1. Blockchain

A BC-based system is characterized by its infrastructure, data structures, networking model, and Consensus Algorithm (CA). The infrastructure can be formally described as follows: Let $G = (V, E, w)$ be connected, undirected and weighted graph, where $V = \{1, \ldots, N\}$ is the set of nodes, usually termed as miners. Each edge $e \in E = \{(i, j) : i, j \in V, i \neq j\}$ is associated with a distinct non-negative real value $w : E \rightarrow \mathbb{R}^+$, namely weight. Weights can be defined in several ways depending on the model application. For example, weights can represent the transmission time needed to deliver 1 bit of data from node i to node j or vice versa, computed in ms.

Data shared between elements of the set V is described according to the application of the system. For example, TXs are submitted by end users to the BC network so that they are processed and added to its DL. Usually, TXs are shared with all miners triggering them to generate new blocks of data. A block usually consists of a header and a body. The header may consist of data such as the type of block, the type of CA, the timestamp, the hash of the body and, most importantly, the proof of block validity. The body, on the other hand, usually includes a group of TXs and the hash of the previous block body.

As BC nodes form a distributed system, these nodes exchange data through a P2P network and communicate by message passing via directly connected lines. The described graph consists of at least one sub-graph $G' \in G : (V', E', w')$, such that $V' \subseteq V$ and $E' \subseteq E$. G' is also undirected and weighted as it inherits these properties from the original graph.

Every BC-based system must operate a CA in order to maintain the consistency of its DL. As tens of CAs were proposed in the literature, a CA is usually considered *valid* if it was proven secure under specific formalized circumstances. One of the main

benchmarks used to describe the security level of a given CA is its tolerance for K faulty/adversarial nodes where $K < N$. For example, the most famous CA, known as Proof-of-Work (PoW) algorithm, was proven secure[2] as long as $K < N/2$. Similarly, the Delegated Byzantine Fault Tolerance (dBFT) algorithm was proven secure as long as $K <= \frac{N-1}{3}$. The upper bound fraction Ψ_N, tolerated by a given CA in order to maintain the system secure, is formalized in Eq. (1).

$$\Psi_N = \frac{K}{N} \tag{1}$$

3.2. Blockchain sharding

Sharding is a type of database partitioning technique that separates a very large database into much smaller, faster, more easily managed parts called data shards [8]. Technically, sharding is a synonym for horizontal partitioning, which makes a large database more manageable and efficient. The key idea of BC sharding is to assign each node in the set V, to a partition. We can simply assume that all nodes are initially assigned to one giant shard if the network is not sharded. Let d be an integer representing the exact number of shards to partition the graph, where $R = [d]$ is the set $\{r_1, r_2, ..r_d\}$. Each shard r_i, $i <= d$ consists of $n_{r_i} \in N$ nodes until Eq. (2) is satisfied.

$$\sum_{i=1}^{d} n_{r_i} = N \tag{2}$$

Furthermore, let α be the minimum number of nodes in each shard, and β is the maximum number of nodes in each shard. β can be either fixed or dynamically defined as in Eq. (3).

$$\beta = N - (d-1)\alpha. \tag{3}$$

Each shard processes a disjoint set of TXs, yet all shards utilize the same CA, leading to increased overall system throughput. As described in [29], d grows linearly with both the total computational power of the network, and with N.

Assume all $v_i \in V$ have the same computational power, and a fraction Ψ_N of which is controlled by a Byzantine adversary. All nodes have access to an externally specified constraint function(s) $C \rightarrow \{0,1\}$ to determine the validity of each TX submitted to or confirmed by the network. A sharding protocol outputs a set R where each shard $r_i \in R$ is trivially expected to contain a subset $k_{r_i} \in K$ adversary nodes leading to a state represented by Eq. (4).

$$\sum_{i=1}^{d} k_{r_i} = K \tag{4}$$

Subconsequently, a shard $r_i \in R$ is considered secure if $k_{r_i}/n_{r_i} <= \Psi_R$, where Ψ_R is the upper bound fraction of adversary/faulty nodes, tolerated in each shard, to maintain the correctness of the shard consensus. In the case where the CA used between shards to agree on a piece of data is the same as the CA used within shards, the intra-shard tolerance Ψ_R should be equal to inter-shard tolerance Ψ_N. Otherwise, each will have a different value.

Studying the literature, we can find several works that attempted to address the need for secure sharding in permissionless BCs. For example, Zhang et al. [30] and Huang et al. [31] assume a leader in each shard, while Manuskin et al. [32] and Naresh et al. [33] used external coordinators (TTPs) to perform the sharding tasks of the BC network. Both [30,31] assumed that $\Psi_N = \Psi_R = 1/3$. However, Zhang et al. [30] deploys a BFT-like method to elect a leader, while Huang et al. [31] uses a reputation-based approach for election. Luu et al. [12] built on even stricter assumptions, specifically that $\Psi_N = 1/4$ and $\Psi_R = 1/3$.

SSHC [34] adopts a sequential committee selection instead of a concurrent method and claims that $\Psi_R < 1/3$ by design. That is, each shard consists of an elected leader and the set R is generated by an elected shard r_{leader}. Additionally, digital signatures and verification were used to verify the honesty of nodes. Alon et al. [35] claimed a tolerance of $\Psi_N < 1/2$ under the condition that this system only works in very large systems. Some other works did approach the problem only for permissioned BCs or with further relaxed initial assumptions. For example, Zhang et al. [36] assumed that all nodes were honest, and evaluated the proposed scheme on the inter-shard level while varying d. We encourage the reader to refer to our comprehensive survey [24], which provides detailed technicalities related to state-of-the-art sharding solutions in permissionless BCs.

3.3. Problem statement

By definition, a BC sharding protocol enhances the BC-based solution, in which the BC is deployed, in terms of scalability. That is, the scalability of a BC-based solution is usually benchmarked by the overall throughput of the system, while increasing N. On the other hand, the security-related *agreement* property is benchmarked with reference to a security parameter which is satisfied in a sharded BC if Condition (5) holds:

$$\forall r_i \in R : \frac{k_{r_i}}{n_{r_i}} <= \Psi_R \tag{5}$$

[2] Assuming all nodes control equal computational powers. Otherwise, Ψ is represented as $q < T/2$, where q is the max collective computational power controlled by the adversary and T is the total hash power of the network.

That is, each shard consists of a fraction of at most Ψ_R faulty/adversary nodes out of all nodes belonging to that shard. Each shard, then, shall provide a level of agreement that is equivalent to the level of agreement of the same BC if it were not sharded.

In most sharding protocols, RS is run regularly in order to maintain a high probability that Condition (5) holds. However, such a sharding approach takes away the optimal data propagation within, and among, shards. Specifically, this approach results in relatively low throughput in exchange for a high level of security.

The trade-off between scalability and security in sharded permissionless BCs is an open optimization problem [37]. Our research goal is to propose a BC sharding protocol that addresses this problem. Specifically, we target a protocol that always results in one of the following two objectives:

1. Find optimal distribution of V among R in terms of scalability, such that Condition (5) is equivalently satisfied as in RS protocols, OR
2. Find RS-equivalent distributions of V among R in terms of scalability, such that Condition (5) is satisfied with higher probability compared to RS protocols.

We assume that Condition (5) is always satisfied when $d = 1$ (i.e., the BC is not sharded). Note that if the adversary nodes were known prior, or while sharding, they must be distributed equally among shards. A protocol such as MaOEA-DRP [19] even optimized the distribution of adversary nodes assuming that they are known before sharding. In our case, we assume that adversary nodes are *not* known to the system prior to or while sharding. However, we assume that an adversary/faulty node can be detected and immediately reported by honest nodes, which might trigger a new round of BC sharding (as Condition (5) may not hold anymore). The detection mechanism of adversary nodes is trivially dependent on whether the proposed solutions are reliably verifiable, a property that we show later to be satisfied in BOSS.

3.4. Overview of the BOSS protocol

Referring to the RapidChain protocol (see Section 2.1), the proposed BOSS protocol adds/modifies two main sub-steps to the original steps as follows:

- First, the reference committee is asked to generate an additional random float number between 0 and 1, namely (λ), during step 1. Note that generating several random numbers during each epoch is a main step of the classical RapidChain protocol. Thus, there is no overhead caused by generating λ.
- Second, λ is used in Step 3 by the reference committee itself to assign nodes to shards using the GA algorithm instead of the Cuckoo rule. To guarantee the public verifiability property, λ is shared with all nodes at the beginning of each epoch so that the node-shard allocation scheme (later generated in step 4) becomes reproducible, i.e. verifiable.

We can highlight the difference between the original Rapidchain approach and the proposed BOSS approach as follows. A new optimization target (either inter- or intra-shard scalability, represented by the value of λ) is randomly selected in a distributed unpredictable way. This value is integrated into a niched Pareto GA-based algorithm, along with other variables such as the weights on the connection links between nodes, that optimizes the new network. The advantage of such approach is enhancing the scalability of the sharded network in terms of data propagation on the inter or intra -shard level. Meanwhile, the proposed BOSS protocol maintains an unpredictable output as the randomness requirement of sharding protocols is *delegated* from the node-shard allocation step in RapidChain to the selection of λ step in BOSS.

In the case of BOSS, if we fix the parameter λ an attacker can collect all its nodes either the closest (in case the target is announced to be intra-shard scalability optimization) or the farthest (in case the target is announced to be inter-shard scalability optimization) from each other.

However, running only a few generations of the GA algorithm by a few nodes is sufficient to address our optimization target. The key security ingredient in our protocol is not hiding λ, nor fixing its value as a function of N. Rather, only the property of λ value being randomly selected for each round oscillates the protocol's goal between inter and intra -shard optimization. This shall, trivially always, produce an optimized sharded network with higher scalability compared to non-sharded and RS-based sharded networks with unpredictable optimization target for any given attacker.

In summary, BOSS could combine the advantages of both RS- and LP-based sharding, namely, the unpredictability and the optimization, respectively. Meanwhile, it could address the sharding problem in permissionless BCs without the limitations of RS and LP, namely the high data propagation in the sharded network and the high costs to compute the sharded network, respectively.

3.5. Threat models

As described earlier, a reference committee in RapidChain is responsible for sharding the network. Participants in this committee are regularly changing using a network-level permissionless voting that we consider secure. That is, the initial assumption is that Ψ_N does not exceed the tolerated level (i.e., 1/3) and thus the participants of the reference committee are guaranteed randomly elected. Specifically, the system nodes will keep terminating with the so-called *safe-value*, until all nodes agree on the same value, indicating a correct random election result.

As a proof-of-concept, we extend the original RapidChain protocol. Thus, our protocol inherits the RapidChain correctness properties of *Fairness*, *Liveness*, and *Unbiasibility*. However, we still need to show the correctness of our protocol in terms of *Verifiability* and *Unpredictability*. To address this, we consider two threat models as follows:

Table 2
Parameters used in our experiments, with the corresponding minimum and maximum tested configurations, and number of test cases per each parameter.

Index	Parameter	Min.	Max.	Step	No. of test cases	Accumulated total
1	No. of Generations	10	100	10	9	9
2	Population size	10	50	10	4	36
3	Nodes to be mutated(%)	0.1	0.5	0.1	4	144
4	Allowed sol. repetitions	3	10	1	7	1008
5	Net Size	10	50	10	5	5040
6	λ	0.1	1	0.1	9	45360
7	K/N	0.1	0.45	0.05	8	**362880**

3.5.1. Reference committee is compromised by an adversary

In this case, correctly and randomly electing the members of the reference committee resulted in the majority of these members being already controlled by an adversary. Accordingly, the adversary can generate the set R so that at least one shard consists of a majority of dishonest nodes. This might obviously lead to manipulated TXs being appended correctly to this shard's chain, accepted by other shards and, consequently, appended to the public DL. This threat is not addressed in the original RapidChain protocol because the resulting set R is not publicly verifiable [11]. We show in our security analysis that our protocol addresses this threat as the reference committee is enforced to propose a publicly verifiable R.

3.5.2. Node-shard assignment is predictable

The node-shard assignment criteria in the original RapidChain protocol is purely random. This means that an adversary that controls K cannot predict or control the next assignment of its nodes. The case of a shard consisting of more than Ψ_R, due to the randomness unpredictability, was solved in RapidChain with the exchange of halving the throughput (as described in Section 2.1).

Our protocol, however, modifies on this RS criterion as it prefers node assignments that provide either optimized inter- or intra-shard propagation delay. Accordingly, the adversary can manipulate the algorithm to have all its K nodes in one shard, if the protocol strictly prefers minimized inter- or intra-shard propagation delays. That is, K can be located very close to each other (in case strictly intra-shard propagation was preferred to be minimized), or very distant from each other (in case strictly inter-shard propagation is preferred to be minimized).

Assuming realistic slowly-adaptive faulty/adversary nodes, we show in our analysis that the adversary may be able to predict the outcome of the protocol, once λ is published, but cannot manipulate it. Furthermore, we show that our protocol satisfies the unpredictability property of the set R. That is, the adversary may be able to change the characteristics of its K nodes, yet it would be impossible for it to predict the outcome of the protocol before the generation of the next λ value. Therefore, changing the characteristics of the nodes in K would be only arbitrary.

4. Optimization methods & algorithms

4.1. Evolutionary sharding optimization

In this paper, we implement a Genetic Algorithm (GA) for solving the underlying MIP problem, as such approach has several advantages in the context of our research. First, GA is inherently heuristic and is not guaranteed to produce globally optimal results. Since we are not seeking such results, there is no need to waste much lengthy time for optimization using LP. At the same time, GA has been proven to provide good enough results for practical use. Second, GA can be designed to run for a predefined number of iterations or a predefined amount of time. Thus, GA is controllable in terms of computational costs. Third, and most importantly, we can improve on the classic GA approach to tune up and down the optimization level, which is the core security factor in our proposal as will be discussed later in the manuscript.

Specifically, GAs work on generations of individuals [38]. In our case, an individual is a list of lists, each list corresponds to a shard. Each generation has predefined number of individuals, where it is a system-wide configurable parameter, namely population size. Computation proceeds in iterations, where both the input and output of each iteration is a generation consisting of the configured population size. There are several other implementation details to fill in the basic algorithm, such as the crossover strategy and mutation strategy. Since these parameters depend on each other non-linearly, making the optimal choices is a highly non-trivial task.

For our experiments, we ran tests with the configurations provided in Table 2. Specifically, we used a simple nested FOR loop to test all cases with each other. We tested for different numbers of generations, or iterations per test case, from 10 to 100. In each iteration, we tested for different population sizes from 10 to 50. We applied only crossover operations under two strict conditions. First, relocating the selected nodes (i.e., each into the other's shard) should result in the mutated shards to be connected. Otherwise, the crossover is reverted. Second, the selection criteria of nodes to be relocated is defined as the first node(s) in a sorted set of node IDs. For instance, if the number of nodes to be relocated per shard per mutation operation is 2, nodes 1, 5 in a shard r_i with the nodes 1, 5, 29, 51 will be tested for mutation. They will be relocated into shard r_{i+1}. The node(s) that satisfy the first condition will remain in their new shards. We further tested for different numbers of nodes to be relocated per shard per mutation operation. This configuration is calculated as a percentage of nodes in the shard. We tested percentages ranging from 0.1 to 0.5.

Note that uniform selection between parent genes is rather easy to achieve by generating random selection vectors. This is typically used to satisfy the *unpredictability* requirement. However, this would lead to *unverifiable* solutions as seen in the original RapidChain protocol.

As we propose a different approach to solve for the *unpredictability* requirement, there was no need to adopt the original node selection strategy. The crossover of a population terminates once the end of the specified number of iterations is reached, or if the same similar solution was obtained for a specific number of allowed repetitions. We tested for different allowed repetitions configurations from 3 to 10.

There are three more configuration parameters that we oscillated in our experiments: Network size, for which we tested sizes ranging from 10 to 50, with a moving $d = \max(\lceil N/10 \rceil, 1)$. The intra-shard weight importance λ will be discussed and justified in the following subsection. Finally, the adversary fraction K/N to be tested against the RapidChain's fixed $\Psi_{r_i} = 1/2$, $\forall r_i \in R$. This was necessary to evidently evaluate our approach, against the RS approach, under different compromising scenarios. Note that RapidChain claimed a system-level tolerance of adversary/faulty nodes $\Psi_N = 1/3$, resulting in a shard-level tolerance $\Psi_R = 1/2$. However, we configured our experiments to test system-level adversary fractions Ψ_N ranging from 0.1 to 0.45, and later assisted the shard-level security with reference to the original $\Psi_R = 1/2$.

4.2. Scalability tuning

To understand the proposed intra-shard weight importance (notated as λ), one should notice that the previously described MIP model takes into consideration only the total weight of the connections within shards. That is, the program will optimize, such that nodes with the least weights on their connection links will be allocated in similar shards. This would lead to predictable allocations and the adversary can place its K at one geographical location to guarantee that all its nodes will be assigned into one shard.

We propose that the importance of such allocation strategy can be varied. That is, the importance of the scalability on the inter-shard level can be, more or less, than the importance of the scalability on the intra-shard level. Note that both directions of scalability enhancements improve the collective system throughput as data is indeed being shared on the shard level (e.g. TXs assigned to this shard) *and* the network level (e.g. Blocks to be verified and permanently appended to the DL). To address this need, we add one binary variable to the previously described MIP model, namely g_{ij}^{cl}. This variable takes the value of one if the edge (i, j) connects nodes $i, j \in V$, which are assigned into different shards $c, l \in R$, and zero otherwise. Accordingly, we modify on the Objective Function (A.1) leading to a niched Pareto [39] Objective Function (6).

$$\min \sum_{(i,j)\in E} (1-\lambda) w_{ij} g_{ij} + \lambda w_{ij} x_{ij} \tag{6}$$

Using Objective Function (6) instead of (A.1) as a fitness function for our GA, equal optimization opportunities to the traded-off inter- and intra-shard scalability can be obtained. That is, increasing the importance of one results in instant increment in the importance of the other. Using such approach, we gain several benefits. First, we achieve a fair scalability optimization between inter- and intra-shard connectivity. Second, in both cases, the scalability level is controllable. Meaning that on the intra-shard level, we can drastically increase/decrease the optimization direction. We can also simply define $\lambda = 0.5$ which would lead to an equal optimality on both levels, or define $\lambda = 1$ to obtain a maximum intra-shard optimization. Third, if λ is guaranteed to be unpredictable, an adversary may in no means predict how the optimization algorithm would behave and, thus, satisfying both scalability optimization and solution unpredictability.

We need to highlight the Pareto optimality that our objective function is targeting. Assuming that w_{ij} represents the connection latency in ms, utilizing the highest value of λ (i.e., $\lambda = 1$) results in shards mostly consisting of miners with very low connection latency. Meanwhile, this latency between shards would be maximized. Accordingly, scalability in terms of data propagation within shards would be near optimum, resulting in the highest throughput possible on the shard level. Data propagation between shards would then be at its worst, resulting in the lowest throughput possible on the network level.

Taking the lowest value of λ (i.e., $\lambda = 0$) would, similarly, result in the highest possible inter-shard throughput with the lowest possible intra-shard throughput. In other words, the spectrum of λ values (i.e., $\lambda \in [0, 1]$) practically defines our algorithm's optimization convergence direction (i.e., Pareto optimality) towards a best shard level or network level throughput, which solves for the trade-off between the two objectives.

4.3. Distributed scalability tuning

Following the so far described methods, we still need to define a secure, unpredictable, and distributed tuning mechanism for λ. That is, the scalability tuning approach integrated into the evolutionary GA-based sharding optimization can be used in both; permissioned and permissionless BCs. However, such sharding approach is unnecessarily too complex for permissioned BCs as a trusted entity is assumed. Thus, only faulty (instead of adversary) nodes can appear throughout the network. As a result, the trusted entity(s) are able to securely optimize the node-shard allocation without concerns of an adversary compromising the network.

Generating λ in a permissionless setting is critical yet simple task. As can be found in the literature, there are several approaches to generate random seeds in a distributed permissionless settings with sufficiently high security guarantees. For example, we can use the RandHound [40] for permissionless BCs and RandHerd [40] or RANDCHAIN [41] for permissioned BCs. High security guarantees include formal proofs that those protocols satisfy the consistency, liveness and unpredictability requirements [42].

In RapidChain, the RandHound protocol is used to securely generate agreed-on random seeds, which in turn are used for building the set R. Since the set R is the output of the proposed GA-based protocol, there would be no need for the classical random seed.

That is, we can instead configure the RandHound sub-protocol to generate random λ values that range between 0 and 1. This value is assumed public and can be accessed by any entity in the network, including an adversary that does not belong to the reference committee. Using this value, reference committee members can work in two ways. They can either wait for an elected leader to canonically optimize the set R, and then verify its output is correct even if the leader was compromised. The other way is performing a Distributed Genetic Algorithm (DGA) [43–45] optimization operations to collaboratively propose the next set R. In both cases, all the algorithm parameters are predefined except for λ.

DGA means that distinct optimization nodes are maintained. Search proceeds in each of these distinct nodes, similarly to any other canonical version. However, in addition to running several independent genetic searches in parallel, copies of the best individuals in each of the sub-populations are migrated to selected neighboring sub-populations. Accordingly, the distributed populations maintain some independence yet occasionally get affected by other solutions, which is the expected case in the centralized version. Several researchers, such as Belding [46] and Patel et al. [47], showed that even with a massive amount of migration, DGA performs better than a canonical GA version. On the computational efficiency and quality of search issues, Adeli and Kumar [48] showed that similar to canonical results can be obtained using DGA in a smaller number of population generations or design iterations. However, we perform our experiments using a canonical approach for simplicity. Further investigations are required regarding the secure utilization of DGA, in particular, which we leave as an open issue for future research.

4.4. Experimental setup

We specified previously the exact values of parameter configurations that we tested using RS and the proposed BOSS protocol. We implemented our code using Python 3.9 and we made it publicly available.[3] Our experiments were carried out on a DELL PC with an Intel i5-8265U CPU (8-Cores, 3.8 GHz) with 12 GB DDR4-SDRAM, 500 GB of SSD and Windows-10 OS. As provided in Table 2, we ran a total of 362880 different test scenarios. With each loop, only one parameter is increased by its corresponding 'step'. Thus, we could abstractly track and capture the general effect of each parameter on both the scalability and security. We used the Networkx library[4] to build random graph connection models in our experiments, namely Erdos-Renyi model [49]. To realize the attack scenario mentioned in Section 4.2, we configured adjacent nodes with the least weights on their connection edges, as to be controlled by an adversary. The number of nodes to be defined as such is determined according to the parameter K/N of the running scenario. To reliably emulate the RS in RapidChain, we used the well documented pymetis[5][6] library.

For each test case, our code shards a randomly generated network twice, once using RS and once using our proposed BOSS protocol (i.e., generates two solutions R_{RS} and R_{BOSS}). Then, it calculates the percentage of scalability enhancement and the security level of each resultant set R using Eqs. (7) and (8), respectively.

$$Scalability = (1 - \frac{Avg\_Shard\_Diameter}{Network\_Diameter}) \times d \tag{7}$$

$$Security\_Level = \frac{Number\_Of\_Secure\_Shards}{d} \tag{8}$$

Note that in both cases, we only consider the intra-shard status as we aim to analyze the generated R sets from the perspective of the adversary (i.e., to show if the adversary can capture a difference between RS results and GA-based sharding results). We assume that a shard is secure if $k_{r_i}/n_{r_i} <= \Psi_R$, and insecure otherwise.

5. Results & discussion

The full database of test cases, along with their corresponding individual scalability and security results, is publicly accessible at [50]. In this section, we analyze this database in order to evaluate the scalability enhancement and the security of the proposed protocol.

5.1. Scalability

To simplify the demonstration of the obtained measures, we present in Fig. 1(a) the upper part of a full scalability comparison figure, for all test cases, starting from the enhancement value '200%'. The minimum enhancement in the full figure is trivially zero. In Fig. 1(b), we provide all scalability enhancement differences (i.e., $R_{BOSS} - R_{RS}$) for all test cases. We can notice that the differences mostly fluctuate between approximately −50% and 80%, leading to an expected average scalability enhancement difference of about $(80 + (-50))/2 = 15\%$. Indeed, averaging the difference in scalability enhancement between R_{BOSS} and R_{RS}, for all test cases, revealed a positive average intra-shard scalability enhancement $> 16.9\%$.

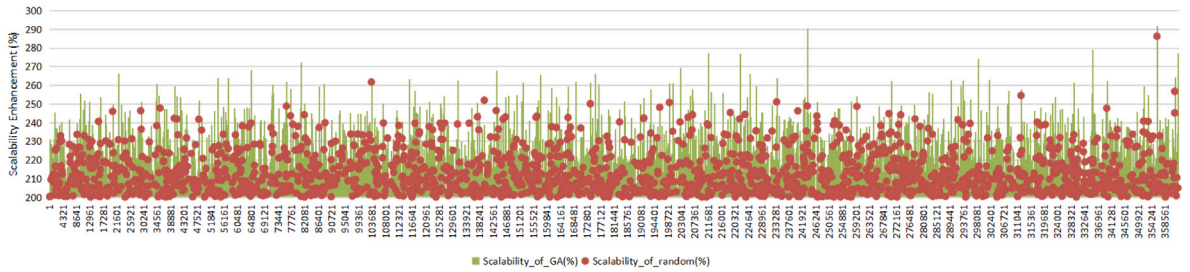
From our full database, we selected the fields at which parameters 1, 2, 3, 4, and 7 are fixed to 50, 20, 0.2, 7, and 0.45, respectively, while parameters 5 and 6 change in their ranges (see Table 2). As discussed previously, network size (N) determines the number of shards (d), which in turn justifies the observed direct correlation of the scalability enhancement with N throughout

[3] https://github.com/HamzaBaniata/Sharding

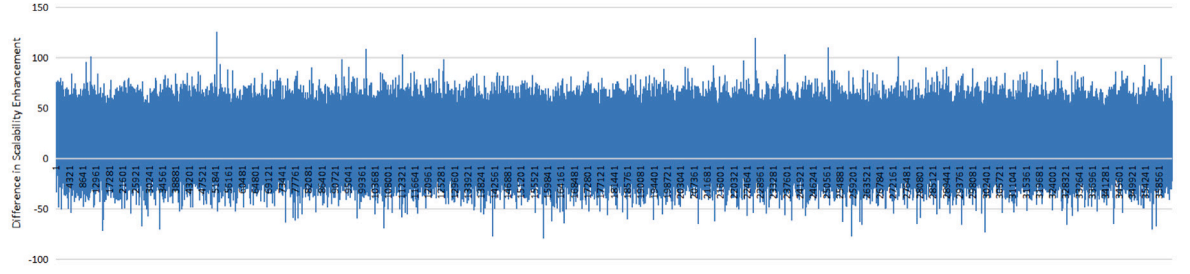
[4] networkx.org

[5] metis.readthedocs.io/en/latest

[6] Github.com/inducer/pymetis



(a) Scalability enhancement measures of our GA-based sharding protocol (BOSS) and the RS protocol, relatively to the scalability of the corresponding non-sharded version of the tested network.



(b) Scalability enhancement differences

Fig. 1. Scalability enhancement measurements (compared to non-sharded) of the proposed GA-based sharding protocol (BOSS) and the RS protocol (a), and individual differences between those scalability measures (b).

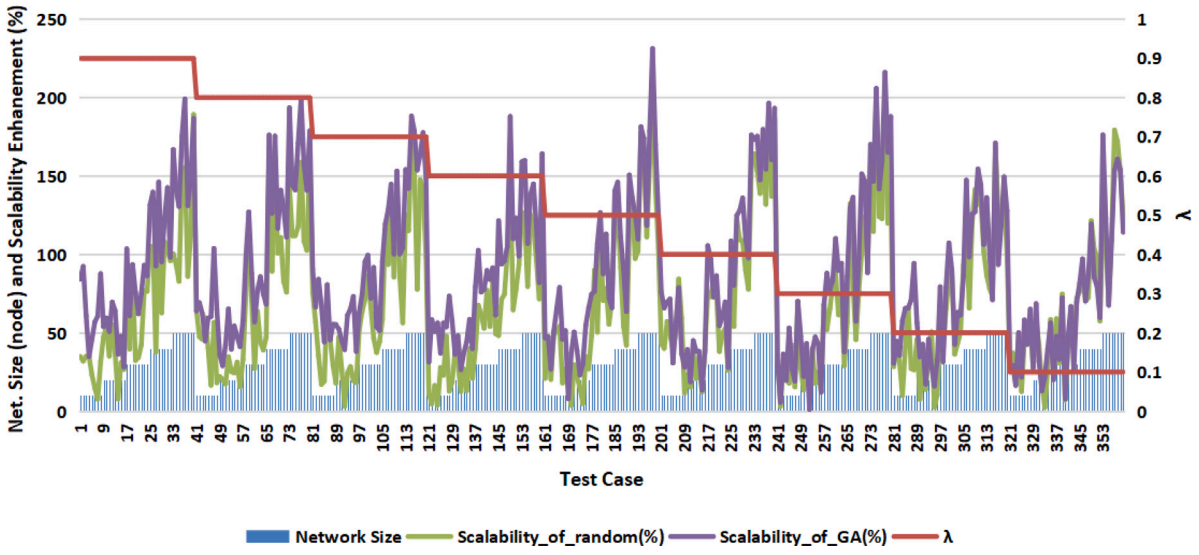


Fig. 2. Scalability enhancement measures, while changing the network size and the intra-shard weight importance (λ), of Randomized Sharding (RS) and our proposed Evolutionary sharding protocol (BOSS). (All values are correlated with the main y-axis on the left, except for λ which is correlated with the secondary y-axis on the right).

our results. This can be seen in Fig. 2, in which we present the scalability enhancements of R_{BOSS} and R_{RS} compared to the corresponding, randomly generated non-sharded network at each case. Our statistics showed that BOSS positively enhanced the scalability of the tested networks in 83.3% of the test cases (304,276 out of 362,880).

We also expected that, in theory, decreasing λ should result in lower intra-shard scalability enhancement. However, we could not easily observe, throughout our results, an obvious correlation between the selected value of λ and the intra-shard scalability enhancement. As can be observed in Fig. 2, the average (and pattern) of intra-shard scalability enhancement using our proposed protocol remains unchanged regardless of the value of λ. We speculated that the expected decrement in scalability was compensated by the effect of another parameter in the experiment. To accurately respond to this speculation, we systematically investigated the correlation between our inputs and results as follows.

Table 3
Pearson correlation coefficient (ρ) between each of the input variables and our resultant scalability and security measures.

Input \ Output	Scalability BOSS	Scalability RS	Scalability diff	Security RS	Security BOSS	Security diff
Net Size (N)	0.799486607	0.791243778	0.126345923	0.080064283	0.05869829	−0.018550198
λ	0.150089933	N/A	0.410773345	N/A	−0.073728401	−0.067350578
No. Generations	−0.001022806	N/A	−0.00130156	N/A	−0.000651157	−0.001110344
Nodes to be mutated(%)	0.00381835	N/A	0.012262811	N/A	0.000985198	0.000286706
Population size	0.032441252	N/A	0.088256124	N/A	−0.004183775	−0.002124442
Allowed sol. repetitions	0.000390606	N/A	0.002407999	N/A	0.000325748	0.000841723
K/N	−0.000108202	−0.000705094	0.001534178	−0.485068285	−0.501122788	−0.022353882
Number of Shards (d)	0.853833329	0.838531281	0.151793186	0.041397203	0.025334008	−0.01423635

We calculated the Pearson correlation coefficient[7] (notated as ρ) between each of the input variables and our resultant scalability and security measures.[8] The closer the calculated coefficient to zero, the lower the correlation between the tested variables. On the other hand, the further the coefficient is from zero, the better the fit and the greater the correlation. Specifically, the values can range from −1 (perfect negative/inverse correlation) to +1 (perfect positive/direct correlation). The calculated correlation results are provided in Table 3. As it can be seen in the table, the parameters that majorly affect the scalability enhancement of our protocol are the network size, with a positive $\rho \approx 0.8$, and the number of shards (d), with a positive $\rho \approx 0.85$. Additionally, λ with a positive $\rho \approx 0.15$ does indeed affect the scalability of our protocol as expected, yet not as strongly as d and N. Note that our protocol's correlation coefficients with these parameters is almost equal to the correlation coefficients between these parameters and the RS protocol.

Since a value of $\rho < 0.2$ typically indicates a low level of correlation, one can argue that it might be probable that such correlation with λ (i.e., $\rho \approx 0.15$) is purely related to the, generally uncontrollable, grade of unexpected randomness in GA [51,52]. However, the highest correlation coefficient value for the difference ($R_{BOSS} - R_{RS}$) is $\rho \approx 0.41$, which relates it with λ. This trivially refutes the argument. To clarify, increasing λ will lead to higher scalability on the intra-shard level. Since we do not consider the inter-shard scalability level in calculating the scalability enhancement (check Eq. (7)), with the intra-shard scalability using the RS protocol being almost constant, the difference is indeed expected to increase while increasing λ.

We highlight that this unexpected weakness in the correlation between the scalability enhancement and λ is a beneficial property as it gives two main advantages. First, this observation is an experimental proof that our protocol could successfully utilize GA to *almost always* generate higher scalability than RS, regardless of the randomly selected value of λ. Second, even with guaranteed higher average scalability enhancement, high λ values do not imply that nodes with the lowest weights on their adjacency connection links would be allocated in similar shards. The first advantage is beneficial in terms of scalability, while the second is beneficial in terms of security. We will further analyze the security of our protocol in the following subsection.

It is also worth noting that ρ between the scalability difference and both the network size N and the number of shards d is $\rho < 0.15$. This indicates that increasing the network size will not majorly increase the difference in scalability between R_{BOSS} and R_{RS}. However, if it did, our protocol is expected to provide better scalability enhancement compared to RS since ρ is positive.

Lastly, regarding the time overhead of the adopted GA over the Cuckoo rule, we could not find specific benchmark of the number of reallocated nodes to compare our GA with. This is because this number is flexibly configurable at the time of deployment. However, there are similar parameters in the BOSS protocol, which are the number of generations, the population size, the percentage of nodes to be mutated, and the allowed solution repetitions. Those parameters have the same effect on the time overhead as the number of reallocated nodes in the Cuckoo rule. As our results showed that all of those parameters imply no practical effect on the scalability and security of the protocols' outputs (see Table 3), we can set these parameters to the minimum values stated in Table 2. During our experiments, we found the time overhead using such minimum configuration to be negligible (i.e., less than a second for a typical PC) compared to the total epoch time (i.e., >600 s).

5.2. RS-equivalency

Referring to the data presented in Table 3, both our protocol and the RS protocol, are mainly, and almost equivalently, affected by K/N, which is a trivial expected observation. That is, regardless of the network size or the number of shards, the more nodes controlled by an adversary in a given network, the higher the probability that the set R will consist of non-secure shards. On the other hand, the correlation coefficient between λ and the security level of the proposed BOSS protocol is very close to zero. This practically means that the value of λ contributes negligibly to the level of security in our protocol.

We demonstrate these observations in Fig. 3, where we plot the security level measurements of both protocols, which we obtained from our tests, while λ and K/N values were changing. Obviously, it is very hard to determine a major difference in the resulted security levels while changing λ and fixing K/N. However, decreasing K/N value indeed decreases the probability of obtaining non-secure shards regardless of the value λ, and thus increases the security level.

[7] Also termed product-moment correlation coefficient

[8] The detailed calculation method is described in Appendix B

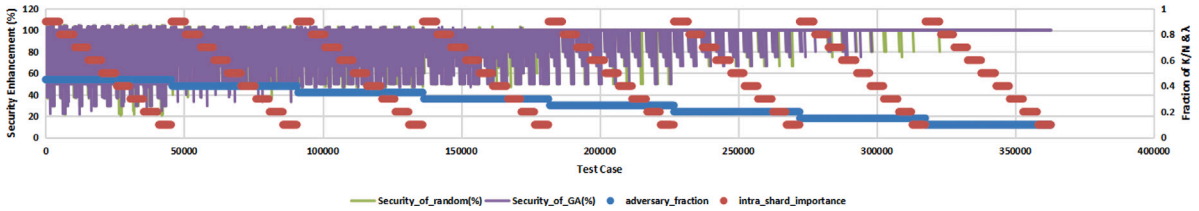


Fig. 3. Security level measurements of our GA-based sharding protocol and the RS protocol (correlated with the main y-axis on the left), while changing intra-shard weight importance (λ) and the actual adversary fraction (K/N) (correlated with the secondary y-axis on the right).

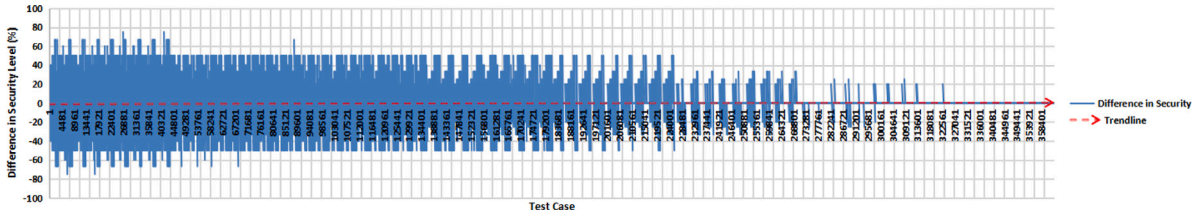


Fig. 4. Difference in the security of our GA-based sharding vs. RS protocols, with trendline for all tested cases.

In Fig. 4, we plot the differences in security values (i.e., $R_{BOSS} - R_{RS}$) while decreasing the fraction K/N. Although we can at a first glance argue that K/N is directly correlated with the difference in security, this is not an accurate description. Specifically, we can see in Table 3 that the security difference correlation with all test parameters, including K/N, is very close to zero. However, decreasing K/N generally decreases the probability of having non-secure shards in the set R regardless of the protocol used for sharding. This can be seen in Fig. 3 which is supported by the correlation data provided in Table 3. As a result, decreasing K/N leads to statistically equivalent increment of secure shards in both sets R_{BOSS} and R_{RS}. In other words, as K/N decreases, the security level of both protocols gets closer to 100% security level, resulting in a less total difference readings of the security levels. We can notice that the differences presented in Fig. 4 mostly fluctuate between approximately −50% and +50%, leading to an expected average security difference of $(50 + (-50))/2 = 0\%$. Indeed, averaging the difference in security levels between R_{BOSS} and R_{RS}, for all test cases, revealed a negligible mean absolute difference $< 0.5\%$.

The results presented thus far clearly prove that our proposed BOSS protocol is able to shard BC networks with an RS-equivalent security level and enhanced scalability. Thus, the proposed BOSS protocol successfully addressed the first research objective declared in Section 3.3. We further need to show that the BOSS protocol fulfills the unpredictability and verifiability properties, demonstrating that this protocol is robust against the threat models listed in Section 3.5.

5.3. Unpredictability

Since the value of λ in our protocol is defined randomly, we take a case where the adversary collects all its miner nodes at one location with almost zero latency. The adversary will honestly adhere to the protocol's rules until λ, by luck, serves for its shard controlling purpose.

As the description of the protocol states, the output of the protocol (i.e., the set R) can only be generated when the value of λ is known. However, the protocol also requires the list of nodes and their corresponding meta data (e.g. identities, neighbors, edge weights, etc.). Now we analyze the steps of the protocol and the expected actions to be made by an adaptive adversary. As in most realistic scenarios [53], we assume a slowly-adaptive adversary.

Let t_s, $t_{s''}$, and t_{s+1} be the instants when the sharding protocol is triggered, terminated, and re-triggered, respectively. We shall show that an adversary cannot manipulate the protocol, during the period $t_{s''} - t_s$ [37], such that its nodes would be included in a selected shard. To facilitate following up on our described timeline, we show the defined instants and periods in Fig. 5.

During the small period of time $t_{s'} - t_s$, the adversary can indeed predict the outcome of the protocol but cannot manipulate it. That is, λ is assigned a randomly selected value, which is generated and agreed-on by the end of the sharding round $s-1$ (i.e., at the instant t_s). Only at t_s, λ can be known by the adversary. Any node that wishes to participate in round s, has to establish an identity by solving a fresh PoW puzzle. This node has to submit a valid PoW solution to the reference committee before a "cutoff time" which is roughly 10 min in RapidChain. The cutoff time then equals precisely $(t_s + 600)$ seconds which is less than $t_{s'}$. Once the cutoff time has passed, each valid solution defines the corresponding node's identity, which is added to the list of active participants. From that point, no puzzle solutions are accepted (even if valid) and the consensus mechanism starts till the arrival at an agreed-on list of participants. After the consensus takes place to agree on the valid participants and their corresponding identities, this list of participants is publicly accessible. Let the period of time needed to reach a consensus on the list of participants, and the instant at which a consensus is reached be T_c and $t_{s'}$, respectively.

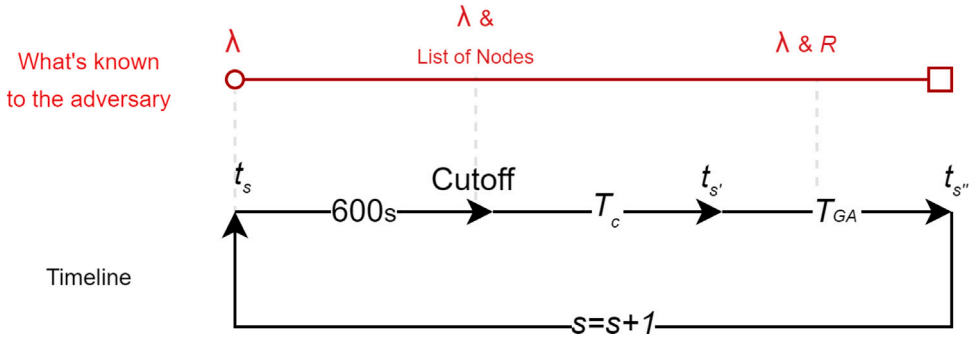


Fig. 5. Timeline of the BOSS protocol and the corresponding expected knowledge of the adversary at each instant of the protocol workflow.

Once the list of participants and the value of λ is known (at instant $t_{s'} = t_s + 600 + T_c$ seconds), any node, including the adversary, can predict which node is going to be assigned to which shard by running our GA-based BOSS protocol. The resultant set R will be generated at instant $t_{s''}$, forcing the adversary nodes to be included in a seemingly randomly-selected shards, and further making the current value of λ no more valid for the following sharding rounds. The time needed to optimize the set R depends mainly on all input parameters mentioned in Table 3, except for K/N, and the computational power of the node that runs the GA algorithm. Assuming the adversary controls stronger resources than the reference committee, the adversary may be able to generate R before the reference committee does. However, the adversary cannot manipulate the distribution in the set R in order to assign its nodes into one shard, except if the optimization is canonically run by a leader node that was originally controlled by the adversary at instant t_s. This issue can be easily solved by the public verifiability guaranteed by our protocol (discussed in the following subsection). Thus, the adversary may only be able to manipulate the protocol during the period $t_{s'} - t_s$.

All parameter configurations are predefined on the system level as we have previously shown that they do not affect the output of the protocol. Nonetheless, the adversary cannot by any means change the value λ nor the list of participants as they were both generated, and agreed-on, in a distributed fashion. Thus, knowing these pieces of information does not provide any help for the adversary regarding its nodes assignment. The adversary, however, can keep track on the submitted PoW solutions by other honest nodes during the period (Cutoff- $t_s = 600$ s). Since the solutions submission times of these honest nodes are distributed along the period (Cutoff- $t_s = 600$ s), only by the end of this period can the adversary disconnect nodes that do not provide high attack probability and reconnect nodes that serve the purpose of the adversary. However, this would be too late as all honest nodes do track the cutoff time and the adversary would not be able to connect the new nodes and submit PoW solutions of these nodes in negligible time.

To clarify, once the instant t_{s+1} is reached, the new value of λ is generated and the adversary would have exactly 600 s to relocate its nodes such that the weights on their corresponding edges (i.e., latency or RTT on the connection link) manipulates the protocol to assign the adversary's nodes into one shard. Let us assume the algorithm was already reverse-engineered and the adversary knows the exact values of edge weights per each λ value. Additionally, let us assume that the adversary had already many nodes that were distributed in several distinct physical locations. The adversary had those nodes distributed in such a way that for each λ value, there is a sub-group with edges and weights that gives high probability of those nodes being assigned to the same shard. In this case, we need to consider several issues as follows:

- The time needed to connect those nodes to the network. Even with a paralleled and automated approach, connecting nodes to the network takes at least several seconds and up to several minutes per node. This was clearly tested in the RapidChain paper[9] indicating an average of 200 s per node to join a network of only 500 nodes.
- The time needed to generate PoW solutions for all these nodes before the cutoff time. For this, the adversary shall control highly strong nodes in terms of computational capacity, such that PoW solutions using the newly generated seed (i.e., λ) can be generated in less than the time needed by most of the network nodes. This practically contradicts the initial assumption that the adversary controls a maximum of $\Psi_N < 1/3$.
- The probability that the adversary gets all nodes in this sub-group peered with each other (i.e., $x_{ij} = 1\ \forall i, j \in K$). The adversary is not able to select peers in permissionless BCs [54], and further is randomly peered when joining a RapidChain protocol [14]. Thus, the probability that the adversary will get its second node peered with its first node is $1/N$. For the third node to be connected to one of the two nodes, the probability would be $2/N$. While the probability would be $1/N^2$ if the third node is to be connected to both two nodes. Assuming the adversary is able to instantly connect and peer its nodes, the probability that the adversary gets all its nodes (in the selected sub-group) peered with each other equals $1/N^K$.

[9] See Section 6.6 in the full RapidChain paper [14]

Finally, during the period of time $t_{s+1} - t_{s''}$, the adversary may be able to change the characteristics of its K nodes, yet it would be impossible for an adversary to predict the outcome of the protocol and, thus, changing the characteristics of the nodes in K is arbitrary. Not to mention, of course, that at this time the list of participants of the sharding round $s+1$ is still not available as well.

In summary, only when the Cutoff instant is reached, the adversary can know both λ and the list of nodes. After that, the adversary can reorganize its nodes which can serve for its attack purpose, which takes a period of time T_A. Since the consensus starts immediately after the Cutoff instant is reached, there is no time window for the adversary to perform its attack tasks. As $T_A > 0$, the tasks performed by the adversary then can serve the attack only in the next sharding round, assuming that all nodes in the current round remain active and connected, and that λ is the same for the next round. The probability of this happening approaches zero with N increasing and, thus, the adversary has no chance to manipulate the protocol. Furthermore, if the adversary could predict its outputs during the period $t_{s''} - t_{s'}$, such predictions are useless for the current and future rounds.

5.4. Verifiability

We are left with one property evaluation for the proposed BOSS protocol, namely the public verifiability. We show in this subsection that this property is satisfied in the proposed protocol. Accordingly, we confirm that this protocol is robust against the yet unaddressed threats mentioned previously.

As mentioned in the previous sections, the set R is generated at instant $t_{s''}$ by the reference committee. After that, the list of nodes and the value of λ, along with the set R are all known to all nodes in the network. Although these will not help in the prediction of the outputs of the next sharding round, they can allow the verification of the most recently generated set R. Simply, the generated set R is reproducible by any node in the network as the initial node-shard allocation is readable. That is, the consensus on the list of participants implies an immutable block, consisting of those participants' identities, with agreed-on order according to their registration timestamp. This block is appended to the publicly accessible DL. Since all parameters (described in Table 2) are unified on the system-level, the initial allocation and λ values can be used to reproduce R. If the regenerated R does not match the originally generated R, further measures can be forced including the re-evaluation of the generator's reputation, re-election of a new reference committee, etc. Thus, even if the protocol deploys a canonical GA optimization and the optimizer node was already compromised by an adversary, the adversary can only delay the arrival of an agreed-on set R as a new sharding round shall be triggered. However, such situation does not affect the safety of the system as at least one node in the network will announce that the set R could not be reproduced. This would trigger all (honest) receiving nodes to try for themselves and take configured countermeasure.

6. Conclusion

In this paper, we proposed a Blockchain Optimized and Secure Sharding (BOSS) protocol for permissionless Blockchain (BC) with the aim of securely increasing the total system throughput. BOSS utilizes a Genetic Algorithm (GA) to shard BC networks, which enhances the system in terms of scalability compared to the typically used Randomized Sharding (RS). Meanwhile, a niched Pareto optimization approach was integrated as a fitness function of the GA program, such that the program will randomly oscillate between inter- and intra-shard oriented optimality. We defined, implemented, and comprehensively tested our protocol with more than 362,880 cases, covering several system and optimization parameters. The evaluation revealed approximately 17% average enhancement in scalability, along with a negligible $< 0.5\%$ mean absolute difference in security level. We have further shown that our proposed protocol satisfies both the public verifiability property, and the unpredictability property. The fulfillment of those properties confirmed the applicability of the proposed protocol in permissionless BCs. To the best of our knowledge, this is the first work that optimized inter- and intra-shard scalability, with publicly verifiable solutions in permissionless BCs, while maintaining an RS-equivalent security and unpredictability.

Declaration of competing interest

The authors whose names are listed immediately below certify that they have NO affiliations with or involvement in any organization or entity with any financial interest (such as honoraria; educational grants; participation in speakers' bureaus; membership, employment, consultancies, stock ownership, or other equity interest; and expert testimony or patent-licensing arrangements), or non-financial interest (such as personal or professional relationships, affiliations, knowledge or beliefs) in the subject mater or materials discussed in this manuscript.

Data availability

Data is publicly accessible and the link to full database is provided within the manuscript.

Acknowledgments

The research leading to these results has received funding from the National Research, Development and Innovation Office within the framework of the Artificial Intelligence National Laboratory Programme, and from the national project TKP2021-NVA-09 implemented with the support provided by the Ministry of Innovation and Technology of Hungary from the National Research, Development and Innovation Fund, financed under the TKP2021-NVA funding scheme, and from the University of Szeged Open Access Fund under the grant number 6534.

Appendix A. Blockchain sharding as a mixed integer program

Blockchain sharding can be modeled for intra-shard optimization, considering three variables as follows: variable y_i^c is a binary variable that takes the value of 1 if the node i is allocated into shard c, for $c \in R$, and zero otherwise. The binary variable x_{ij} takes the value of one if the edge (i, j) connects nodes i and j in the same shard, and zero otherwise. Finally, the continuous variable f_{ij} representing the flow over the edge (i, j). Moreover, $\mathbb{A} = \{n + 1, \ldots, n + d\}$ is a set of artificial nodes, one for each shard in R. The model, denoted as $\mathbb{M}$, can then be formulated as follows:

$$\min \sum_{(i,j)\in E} w_{ij} x_{ij} \tag{A.1}$$

subject to

$$\sum_{c\in R} y_i^c = 1 \qquad \forall\, i \in V \tag{A.2}$$

$$y_i^c + y_j^c - x_{ij} \le 1 \qquad \forall\, (i, j) \in E, c \in R \tag{A.3}$$

$$y_i^c + y_j^l + x_{ij} \le 2 \qquad \forall\, (i, j) \in E, c, l \in R, c \ne l \tag{A.4}$$

$$\sum_{j\in V} x_{ij} = 1 \qquad \forall\, i \in \mathbb{A} \tag{A.5}$$

$$\sum_{j\in V} f_{ij} = \sum_{j\in V} y_j^c \qquad \forall\, i \in \mathbb{A} \tag{A.6}$$

$$\sum_{(i,j)\in E} f_{ij} - \sum_{(j,i)\in E} f_{ji} = 1 \qquad \forall\, j \in V \tag{A.7}$$

$$f_{ij} + f_{ji} \le \beta x_{ij} \qquad \forall\, (i, j) \in E \tag{A.8}$$

$$\alpha x_{ij} \le f_{ij} \le \beta x_{ij} \qquad \forall\, i \in \mathbb{A}, j \in V \tag{A.9}$$

$$y_i^c \in \{0, 1\} \quad \forall i \in V, c \in R$$
$$x_{ij} \in \{0, 1\} \quad \forall (i, j) \in E$$
$$f_{ij} \in \mathbb{R}^+ \quad \forall (i, j) \in E$$

Constraint (A.2) guarantees that each node is allocated to exactly one shard. Constraint (A.3) guarantees that if node i and j are in the same shard then the variable x_{ij} set to 1, while constraint (A.4) ensures if nodes i and j are in different shards then x_{ij} has to be zero. Constraints (A.5)–(A.7) are flow conservation constraints. Constraint (A.5) guarantees that each artificial node in $\mathbb{A}$ is exactly connected to one node in the graph. Constraint (A.6) guarantees that the outflow from artificial node i to node j is equal to the number of nodes in shard c that node j belongs to. Constraint (A.7) guarantees that the outflow from node j is equal to the inflow into node j. Constraints (A.8) and (A.9) guarantee that the number of nodes in a given shard is bounded by α and β.

Appendix B. Correlation coefficient

Generally, the Correlation Coefficient is defined as a statistical measure of the strength of a linear relationship between two variables. The closer the calculated coefficient to zero, the lower the correlation between the tested variables. On the other hand, the further the coefficient is from zero, the better the fit and the greater the correlation. Specifically, the values can range from −1 (perfect negative/inverse correlation) to +1 (perfect positive/direct correlation). The Pearson coefficient ρ uses a mathematical statistics formula to measure how closely the data points combining the two variables (with the values of one data series plotted on the x-axis and the corresponding values of the other series on the y-axis) approximate the line of best fit. The line of best fit can be determined through regression analysis.

To calculate ρ, each variable's standard deviation needs to be calculated as well as the covariance between the considered variables. Subconsequently, ρ is the covariance divided by the product of the two variables' standard deviations. The mathematical formula for such calculations is given in Eq. (B.1).

$$\rho_{xy} = \frac{\mathrm{Cov}(x, y)}{\sigma_x \sigma_y} \tag{B.1}$$

where:

- ρ is the Pearson product-moment correlation coefficient,
- $\mathrm{Cov}(x, y)$ is the covariance of variables x and y,
- σ is the standard deviation.

Standard deviation is a measure of the dispersion of data from its average. Covariance shows whether the two variables tend to move in the same direction, while the correlation coefficient measures the strength of that relationship on a normalized scale, from −1 to 1.

References

[1] S. Perera, S. Nanayakkara, M. Rodrigo, S. Senaratne, R. Weinand, Blockchain technology: Is it hype or real in the construction industry? J. Ind. Inform. Integr. 17 (2020) 100125.

[2] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, Decentralized Bus. Rev. (2008) 21260.

[3] C. Bernardino, C.J. Costa, M. Aparício, Digital evolution: Blockchain field research, in: 2022 17th Iberian Conference on Information Systems and Technologies, CISTI, IEEE, 2022, pp. 1–6.

[4] D.D.F. Maesa, P. Mori, Blockchain 3.0 applications survey, J. Parallel Distrib. Comput. 138 (2020) 99–114.

[5] A.A. Monrat, O. Schelén, K. Andersson, A survey of blockchain from the perspectives of applications, challenges, and opportunities, IEEE Access 7 (2019) 117134–117151.

[6] H. Baniata, T. Pflanzner, Z. Feher, A. Kertesz, Latency assessment of blockchain-based SSI applications utilizing hyperledger indy, in: Proceedings of the 12th International Conference on Cloud Computing and Services Science - Volume 1: CLOSER, SciTePres, INSTICC, 2022, pp. 264–271, http://dx.doi.org/10.5220/0011082300003200.

[7] G. Yu, X. Wang, K. Yu, W. Ni, J.A. Zhang, R.P. Liu, Survey: Sharding in blockchains, IEEE Access 8 (2020) 14155–14181.

[8] Y. Liu, Y. Wang, Y. Jin, Research on the improvement of MongoDB Auto-Sharding in cloud environment, in: 2012 7th International Conference on Computer Science & Education, ICCSE, IEEE, 2012, pp. 851–854.

[9] Y.-F. Ge, Z.-H. Zhan, J. Cao, H. Wang, Y. Zhang, K.-K. Lai, J. Zhang, DSGA: A distributed segment-based genetic algorithm for multi-objective outsourced database partitioning, Inform. Sci. 612 (2022) 864–886.

[10] G. Kim, M. Franz, J. Kim, The ticket price matters in sharding blockchain, in: Data Privacy Management, Cryptocurrencies and Blockchain Technology: ESORICS 2022 International Workshops, DPM 2022 and CBT 2022, Copenhagen, Denmark, September 26–30, 2022, Revised Selected Papers, Springer, 2023, pp. 185–202.

[11] R. Han, Scaling Permissionless Blockchains Via Sharding (Ph.D. thesis), Monash University, 2022.

[12] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, P. Saxena, A secure sharding protocol for open blockchains, in: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, 2016, pp. 17–30.

[13] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, B. Ford, Omniledger: A secure, scale-out, decentralized ledger via sharding, in: 2018 IEEE Symposium on Security and Privacy, SP, IEEE, 2018, pp. 583–598.

[14] M. Zamani, M. Movahedi, M. Raykova, Rapidchain: Scaling blockchain via full sharding, in: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018, pp. 931–948.

[15] T. Rajab, M.H. Manshaei, M. Dakhilalian, M. Jadliwala, M.A. Rahman, On the feasibility of sybil attacks in shard-based permissionless blockchains, 2020, arXiv preprint arXiv:2002.06531.

[16] B. Wang, J. Jiao, S. Wu, R. Lu, Q. Zhang, Age-critical and secure blockchain sharding scheme for satellite-based internet of things, IEEE Trans. Wireless Commun. 21 (11) (2022) 9432–9446.

[17] N. Gao, R. Huo, S. Wang, T. Huang, Y. Liu, Sharding-hashgraph: A high-performance blockchain-based framework for industrial internet of things with hashgraph mechanism, IEEE Internet Things J. 9 (18) (2021) 17070–17079.

[18] R. Han, J. Yu, H. Lin, S. Chen, P. Esteves-Veríssimo, On the security and performance of blockchain sharding, Cryptol. ePrint Arch. (2021).

[19] X. Cai, S. Geng, J. Zhang, D. Wu, Z. Cui, W. Zhang, J. Chen, A sharding scheme-based many-objective optimization algorithm for enhancing security in blockchain-enabled industrial internet of things, IEEE Trans. Ind. Inform. 17 (11) (2021) 7650–7658.

[20] M.N. Halgamuge, S.P. Mapatunage, Fair rewarding mechanism for sharding-based blockchain networks with low-powered devices in the internet of things, in: 2021 IEEE 16th Conference on Industrial Electronics and Applications, ICIEA, IEEE, 2021, pp. 504–509.

[21] E. Mehraein, Z. Ahmadian, R. Nourmohammadi, IGD-ScoreChain: A novel lightweight-scalable blockchain based on nodes sharding for the internet of things, Cryptol. ePrint Arch. (2023).

[22] S. Sen, M.J. Freedman, Commensal cuckoo: Secure group partitioning for large-scale services, Oper. Syst. Rev. 46 (1) (2012) 33–39.

[23] L. Ren, K. Nayak, I. Abraham, S. Devadas, Practical synchronous byzantine consensus, 2017, arXiv preprint arXiv:1704.02397.

[24] H. Baniata, A. Kertesz, Approaches to overpower proof-of-work blockchains despite minority, IEEE Access 11 (2023) 2952–2967.

[25] M. Cordero, A. Miniguano-Trujillo, D. Recalde, R. Torres, P. Vaca, Graph partitioning in connected components with minimum size constraints via mixed integer programming, 2022, arXiv. http://dx.doi.org/10.48550/ARXIV.2202.11254.

[26] A. Henzinger, A. Noe, C. Schulz, ILP-Based local search for graph partitioning, J. Exp. Algorithmics (JEA) 25 (2020) 1–26.

[27] B. Borchers, J.E. Mitchell, An improved branch and bound algorithm for mixed integer nonlinear programs, Comput. Oper. Res. 21 (4) (1994) 359–367.

[28] I.S. Dhillon, Co-clustering documents and words using bipartite spectral graph partitioning, in: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2001, pp. 269–274.

[29] G. Wang, Z.J. Shi, M. Nixon, S. Han, Sok: Sharding on blockchain, in: Proceedings of the 1st ACM Conference on Advances in Financial Technologies, 2019, pp. 41–61.

[30] M. Zhang, J. Li, Z. Chen, H. Chen, X. Deng, Cycledger: A scalable and secure parallel protocol for distributed ledger via sharding, in: 2020 IEEE International Parallel and Distributed Processing Symposium, IPDPS, IEEE, 2020, pp. 358–367.

[31] C. Huang, Z. Wang, H. Chen, Q. Hu, Q. Zhang, W. Wang, X. Guan, Repchain: A reputation-based secure, fast, and high incentive blockchain system via sharding, IEEE Internet Things J. 8 (6) (2020) 4291–4304.

[32] A. Manuskin, M. Mirkin, I. Eyal, Ostraka: Secure blockchain scaling by node sharding, in: 2020 IEEE European Symposium on Security and Privacy Workshops, EuroS&PW, IEEE, 2020, pp. 397–406.

[33] V.S. Naresh, V.D. Allavarpu, S. Reddi, P.S.R. Murty, N.L. Raju, R.J. Mohan, A provably secure sharding based blockchain smart contract centric hierarchical group key agreement for large wireless ad-hoc networks, Concurr. Comput.: Pract. Exper. (2022) e6553.

[34] Y. Liu, J. Liu, Q. Wu, H. Yu, H. Yiming, Z. Zhou, SSHC: A secure and scalable hybrid consensus protocol for sharding blockchains with a formal security framework, IEEE Trans. Dependable Secure Comput. (2020).

[35] N. Alon, H. Kaplan, M. Krivelevich, D. Malkhi, J. Stern, Scalable secure storage when half the system is faulty, in: International Colloquium on Automata, Languages, and Programming, Springer, 2000, pp. 576–587.

[36] Z. Zhang, X. Wang, G. Yu, W. Ni, R.P. Liu, N. Georgalas, A. Reeves, A community detection-based blockchain sharding scheme, in: Blockchain–ICBC 2022: 5th International Conference, Held As Part of the Services Conference Federation, SCF 2022, Honolulu, HI, USA, December 10–14, 2022, Proceedings, Springer, 2022, pp. 78–91.

[37] A. Mariani, G. Mariani, D. Pennino, M. Pizzonia, Blockchain scalability and security: Communications among fast-changing committees made simple, in: 2023 IEEE 20th International Conference on Software Architecture Companion, ICSA-C, IEEE, 2023, pp. 209–215.

[38] D. Bogdanov, K. Emura, R. Jagomägis, A. Kanaoka, S. Matsuo, J. Willemson, A secure genetic algorithm for the subset cover problem and its application to privacy protection, in: Information Security Theory and Practice. Securing the Internet of Things: 8th IFIP WG 11.2 International Workshop, WISTP 2014, Heraklion, Crete, Greece, June 30–July 2, 2014. Proceedings. Vol. 8, Springer, 2014, pp. 108–123.

[39] J. Horn, N. Nafpliotis, D.E. Goldberg, A niched Pareto genetic algorithm for multiobjective optimization, in: Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence, Ieee, 1994, pp. 82–87.

[40] E. Syta, P. Jovanovic, E.K. Kogias, N. Gailly, L. Gasser, I. Khoffi, M.J. Fischer, B. Ford, Scalable bias-resistant distributed randomness, in: 2017 IEEE Symposium on Security and Privacy, SP, Ieee, 2017, pp. 444–460.
[41] G. Wang, M. Nixon, Randchain: Practical scalable decentralized randomness attested by blockchain, in: 2020 IEEE International Conference on Blockchain, Blockchain, IEEE, 2020, pp. 442–449.
[42] S. Das, V. Krishnan, I.M. Isaac, L. Ren, Spurt: Scalable distributed randomness beacon with transparent setup, in: 2022 IEEE Symposium on Security and Privacy, SP, IEEE, 2022, pp. 2502–2517.
[43] D. Whitley, T. Starkweather, Genitor II: A distributed genetic algorithm, J. Exp. Theor. Artif. Intell. 2 (3) (1990) 189–214.
[44] Y.-J. Gong, W.-N. Chen, Z.-H. Zhan, J. Zhang, Y. Li, Q. Zhang, J.-J. Li, Distributed evolutionary algorithms and their models: A survey of the state-of-the-art, Appl. Soft Comput. 34 (2015) 286–300.
[45] M. Rodríguez, D.M. Escalante, A. Peregrín, Efficient distributed genetic algorithm for rule extraction, Appl. Soft Comput. 11 (1) (2011) 733–743.
[46] T.C. Belding, The distributed genetic algorithm revisited, 1995, arXiv preprint adap-org/9504007.
[47] R. Patel, E. Rudnick-Cohen, S. Azarm, M. Otte, H. Xu, J.W. Herrmann, Decentralized task allocation in multi-agent systems using a decentralized genetic algorithm, in: 2020 IEEE International Conference on Robotics and Automation, ICRA, 2020, pp. 3770–3776, http://dx.doi.org/10.1109/ICRA40945.2020.9197314.
[48] H. Adeli, S. Kumar, Distributed genetic algorithm for structural optimization, J. Aerospace Eng. 8 (3) (1995) 156–163.
[49] P. Erdős, A. Rényi, et al., On the evolution of random graphs, Publ. Math. Inst. Hung. Acad. Sci. 5 (1) (1960) 17–60.
[50] H. Baniata, A. Anaqreh, A. Kertesz, Full Database for Test Results of the BOSS Protocol, Zenodo, 2023, URL https://doi.org/10.5281/zenodo.8037447.
[51] H. Toutounji, A.C. Aljundi, On randomness and the genetic behavior of cellular automata, in: 2008 3rd International Conference on Information and Communication Technologies: From Theory To Applications, IEEE, 2008, pp. 1–6.
[52] R. Caponetto, L. Fortuna, S. Fazzino, M.G. Xibilia, Chaotic sequences to improve the performance of evolutionary algorithms, IEEE Trans. Evol. Comput. 7 (3) (2003) 289–304.
[53] D. Tennakoon, V. Gramoli, Dynamic blockchain sharding, in: 5th International Symposium on Foundations and Applications of Blockchain 2022, FAB 2022, Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2022, pp. 6:1–6:17.
[54] Y. Shahsavari, K. Zhang, C. Talhi, Toward quantifying decentralization of blockchain networks with relay nodes, Front. Blockchain 5 (2022) 1.