

A Range-Based Sharding (RBS) Protocol for Scalable Enterprise Blockchain

M.Z. Haider, M. Dias de Assunção, Kaiwen Zhang

¹Department of Software Engineering, ÉTS Montréal, Université de Québec, Canada

Abstract—Blockchain technology offers decentralization and security but struggles with scalability, particularly in enterprise settings where efficiency and controlled access are paramount. Sharding is a promising solution for private blockchains, yet existing approaches face challenges in coordinating shards, ensuring fault tolerance with limited nodes, and minimizing the high overhead of consensus mechanisms like PBFT. This paper proposes the Range-Based Sharding (RBS) Protocol, a novel sharding mechanism tailored for enterprise blockchains, implemented on Quorum. Unlike traditional sharding models such as OmniLedger and non-sharding Corda framework, RBS employs a commit-reveal scheme for secure and unbiased shard allocation, ensuring fair validator distribution while reducing cross-shard transaction delays. Our approach enhances scalability by balancing computational loads across shards, reducing consensus overhead, and improving parallel transaction execution. Experimental evaluations demonstrate that RBS achieves significantly higher throughput and lower latency compared to existing enterprise sharding frameworks, making it a viable and efficient solution for large-scale blockchain deployments.

I. INTRODUCTION

Blockchain technology offers decentralization, security, and transparency, making it attractive for various applications, including finance and supply chain management [1]. While public blockchains like Bitcoin and Ethereum are well known, permissioned blockchains [2] provide a more controlled environment suitable for enterprise needs, offering better privacy and efficiency. However, scalability remains a key challenge for permissioned blockchains as enterprises expand their networks [3], [4]. Traditional scalability solutions, such as increasing block size or reducing block time, often fall short in permissioned settings where a balance between security, scalability, and efficiency is crucial [5], [6]. Blockchain sharding has emerged as a promising technique to address this issue by dividing the network into smaller, parallel segments or “shards” [7]. Each shard processes transactions independently, increasing throughput. Although public blockchains have adopted sharding [4], [8], adapting it to the unique needs of permissioned blockchains remains an area ripe for exploration [9], [10], [11]. Traditional approaches to improving scalability in blockchain systems [12] [13], such as increasing block size or reducing block generation time, have proven insufficient, particularly in the context of permissioned blockchains [3][14]. Such solutions are designed with

public blockchains in mind, where the focus is often on maximizing decentralization rather than optimizing performance [14][5][6]. However, adapting sharding to permissioned blockchains presents unique challenges. These include managing the consensus overhead of traditional mechanisms like PBFT or IBFT [15] and some DAG based solutions[16], ensuring atomicity and consistency in cross-shard transactions, and addressing fault tolerance within shards [4], [8]. Additionally, dynamic shard reconfiguration to handle varying workloads without disruption, and maintaining the privacy and access control requirements of permissioned environments, further complicate sharding adoption[2]. These challenges underscore the need for tailored solutions to enable efficient sharding in permissioned blockchains[13].

This paper introduces a novel approach using Range-Based Sharding Protocol (RBS), designed for enterprise blockchain environments where traditional sharding such as OmniLedger, Corda, Near protocol is not effective, specifically implemented on the Quorum platform, a permissioned variant of Ethereum that is widely used in enterprise settings. Our method incorporates a commit-reveal scheme for secure randomness generation in shard allocation and consensus, enabling parallel transaction processing while maintaining network integrity.

Our contributions include:

- 1) **Optimized Range-Based Sharding:** Utilizing range-based partitioning with adaptive load balancing, our mechanism enhances transaction throughput while minimizing cross-shard communication overhead.
- 2) **Efficient and Secure Cross-Shard Transactions:** Implementing a fine-grained locking and batched commit model, we reduce sequential dependencies and improve parallel execution of cross-shard transactions..
- 3) **Adaptive Validator Selection and Security:** Enhancing the commit-reveal scheme, we ensure fair and collusion-resistant shard leader election while strengthening security against adversarial control and Byzantine faults.

The rest of this paper is structured as follows: Section II presents the background and related work information. We present detailed RBS in Section III and scalability, performance results in Section IV.

Section V contains a security analysis while Section VI gives the conclusion and future direction of the paper.

II. BACKGROUND AND RELATED WORK

This section reviews blockchain sharding as a means to enhance scalability and security in permissioned networks. By dividing the network into shards for parallel processing, sharding improves throughput while managing cross-shard complexities [17]. Private blockchains, with controlled access and efficient consensus, are well-suited for enterprise use [8]. Byzantine Fault Tolerance (BFT) and redundancy remain essential for fault tolerance and consistency. We classify consensus protocols for permissioned blockchains into sharding and non-sharding approaches (see Table I).

Non-sharding protocols: Frameworks like Hyperledger Fabric [18], Ethereum Private Blockchain [19], and Corda R3 Enterprise[20] have tailored their designs to enterprise needs. Hyperledger Fabric uses a permissioned model with channels for private communication and chain code for smart contracts, enhancing privacy and operational efficiency. However, Hyperledger faces scalability and latency issues in multi-channel setups [8]. Ethereum Private offers a permissioned model with privacy technologies like zk-SNARKs and optimized consensus mechanisms like Proof of Authority (PoA) or Proof of Stake (PoS), yet it struggles with scalability in large networks [8]. Corda uses a point-to-point communication model for data confidentiality, excelling in workflow automation but encountering interoperability and latency challenges in multi-party transactions [21].

Sharding protocols: The limitations of non-sharding protocols led to the development of sharding-based solutions. Sharding divides the network into smaller segments (shards), each processing transactions independently, thus reducing the load on individual nodes and enhancing network performance. RapidChain [22] partitions the network into shards with local consensus, improving throughput but facing challenges in cross-shard communication and workload distribution [8]. Benzene [23] further demonstrate sharding's potential in private settings. OmniLedger uses parallel sharding for rapid processing but needs to address cross-shard communication vulnerabilities and load balancing. Benzene distributes computational workloads across shards but faces similar challenges in maintaining secure shard interactions. Sharper [24], Pyramid [25], and Repchain [26] refine the sharding model. Sharper boosts throughput but must address security in inter-shard communication and workload distribution. Pyramid improves transaction efficiency but struggles with seamless scaling [27]. Repchain offers quick local consensus but faces difficulties in cross-shard coordination. Public blockchain protocols like Meepo [28] and RapidChain [22] have explored shard-specific consensus, but security needs constrain their scalability during shard interactions[27].

Hybrid approaches: Hybrid sharding frameworks in blockchain combine multiple sharding techniques to achieve scalability, security, and efficiency. DynaShard [29] employs transactional and state sharding with RandHound for randomness and Atomix for atomic cross-shard transactions, ensuring both scalability and security. Aelous[22] enhances adaptability using erasure coding for fault tolerance and dynamic shard reallocation. Ethereum 2.0 [19] adopts hierarchical sharding with a Beacon Chain for coordination and execution shards for parallel processing. Despite these advancements, hybrid sharding frameworks face challenges such as high cross-shard transaction complexity, potential bottlenecks in coordination mechanisms, increased storage overhead due to data replication, and security vulnerabilities arising from inter-shard communication[30]. Ensuring dynamic shard reallocation without compromising security and optimizing cross-shard data consistency remain key research challenges in hybrid sharding implementations[31], [32].

Proposed Solution (RBS): Our solution dynamically adjusts shard allocation to minimize cross-shard bottlenecks and improve transaction throughput. Unlike traditional sharding approaches, RBS reduces transaction overhead by incorporating fine-grained account-level locking and a batched commit model, allowing parallel execution of cross-shard transactions without unnecessary sequential dependencies. It employs an commit-reveal scheme for secure and unbiased validator selection, ensuring resilience against adversarial control and Byzantine faults. Additionally, RBS features adaptive shard rebalancing to optimize workload distribution based on real-time network conditions. Through these innovations, RBS improves scalability, security, and transaction efficiency in enterprise blockchain environments.

III. PROPOSED RANGE-BASED SHARDING PROTOCOL (RBS)

Our proposed solution, RBS, improves scalability and efficiency in enterprise blockchain networks through range-based data partitioning, a commit-reveal scheme for randomness, and an integrated fine-grained locking with batched commit model. Figure 1 illustrates the architecture, detailing node authentication, shard formation, leader selection, and smart contract-driven synchronization.

A. Genesis Configuration and Shard Formation

The genesis configuration establishes the initial network state, defining the key space, shard count, and transaction routing. For the distribution of accounts, transactions, shards, and data in RBS. The total key space (K) is divided into non-overlapping ranges (R_1, R_2, \dots, R_n), each mapped to a shard as given in Figure 2:

$$S_i = \{k \in K \mid a_i \leq k < a_{i+1}\}$$

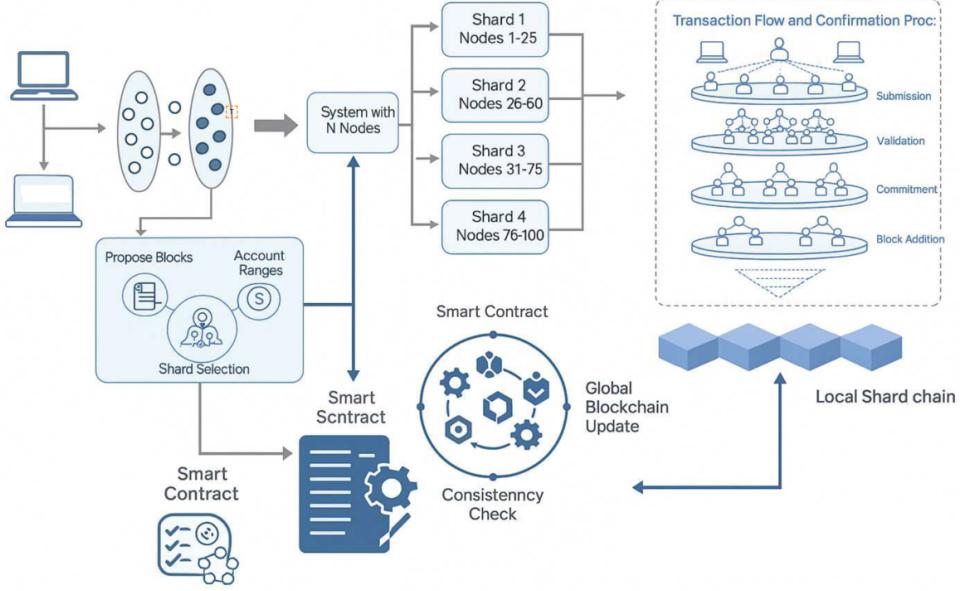


Fig. 1. Architecture of range-based sharding protocol.

where S_i is shard i , and a_i and a_{i+1} are the range boundaries. The key space partitioning is:

$$K = \bigcup_{i=1}^n R_i \quad \text{and} \quad R_i \cap R_j = \emptyset \quad \forall i \neq j$$

Dynamic range adjustments are used to minimize data skew, measured by the row count skew ratio σ :

$$\sigma = \frac{\max \text{ row count in any shard}}{\text{total row count}}_n$$

A skew ratio $\sigma \approx 1$ indicates balanced shards. The genesis block includes initial validator nodes and their shard assignments, with validator selection probability P for cross-shard transactions given by:

$$P = 1 - \left(\frac{n_h}{N} \right)^k$$

where n_h is the number of honest nodes, N is the total number of nodes, and k is the number of validators needed.

B. Consensus and Validator Selection

RBS uses **IBFT (Istanbul Byzantine Fault Tolerance)**, adapted to a multi-shard environment. The consensus process involves the *Pre-Prepare*, *Prepare*, and *Commit* phases, ensuring deterministic finality for transactions within shards. Validators are chosen based on a reputation score R_v :

$$R_v = w_1 \times P_v + w_2 \times T_v,$$

where P_v is the performance score, T_v is the trust score, and w_1 and w_2 are weighting factors.

1) Intra-Shard Consensus

The primary node (leader) broadcasts transactions, and validators follow the IBFT process to achieve consensus. IBFT ensures deterministic finality and achieves consensus with a message complexity of

$O(n^2)$, where n is the number of nodes in the shard. Validators participate in the *Pre-Prepare*, *Prepare*, and *Commit* phases to validate and finalize transactions.

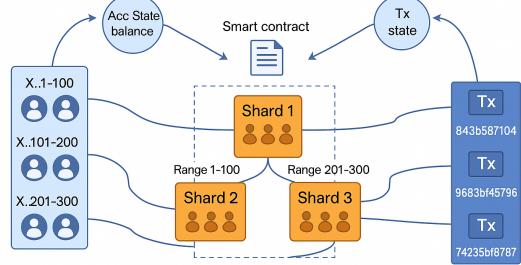


Fig. 2. Range distribution of accounts, transactions and data.

2) Cross-Shard Consensus

For transactions that span multiple shards, a weighted selection mechanism is used to ensure reliable validator participation:

$$P_v = \frac{R_v}{\sum_{v \in S_i} R_v},$$

where S_i represents the set of validators participating in shard i . Unlike traditional methods that lock the entire shard state, fine-grained locking restricts access only to specific accounts involved in a transaction, preventing unnecessary delays and enabling higher throughput with parallel execution. To ensure security and consistency, Merkle proof locking generates a cryptographic proof of the locked state in the source shard, which the receiving shard verifies before committing the transaction, mitigating risks such as double-spending and invalid state transitions.

Algorithm 1 The range-based distribution protocol.

Require: Nodes N , Transactions Tx , Data D , Range R , Shards S

Ensure: Distributed Nodes, Transactions, and Data across Shards

```

1: Initialize Shards  $S$ 
2: Initialize HashMap  $H$  to maintain node-range mappings
3: Assign Range  $R = \{r_1, r_2, \dots, r_k\}$  such that each shard  $S_i \in S$  has a unique range  $r_i$ 
4:  $Q \leftarrow \emptyset$   $\triangleright$  Queue for pending transactions and data
5:  $Q \leftarrow Q \cup (Tx, D)$ 
6: for all  $n \in N$  do  $\triangleright$  Distribute nodes to shards
7:    $hash\_val \leftarrow \text{Hash}(n)$ 
8:    $shard\_id \leftarrow \text{FindShard}(hash\_val, R)$ 
9:    $H[shard\_id] \leftarrow H[shard\_id] \cup \{n\}$ 
10: end for
11: for all  $(tx, d) \in Q$  do  $\triangleright$  Distribute transactions and data to shards
12:    $hash\_val \leftarrow \text{Hash}(tx)$ 
13:    $shard\_id \leftarrow \text{FindShard}(hash\_val, R)$ 
14:    $S[shard\_id].Tx \leftarrow S[shard\_id].Tx \cup \{tx\}$ 
15:    $S[shard\_id].D \leftarrow S[shard\_id].D \cup \{d\}$ 
16: end for
17: return  $S$ 

```

C. Randomness Generation

A commit-reveal scheme ensures fair randomness for validator selection and other operations:

1) *Commit Phase*: Each participant P_i selects a random value v_i from a predetermined range and broadcasts a commitment C_i :

$$C_i = H(v_i \parallel r_i)$$

where H is a cryptographic hash function, \parallel denotes concatenation, and r_i is a random nonce to ensure the uniqueness of each commitment. This hash is broadcast to all other participants, ensuring no one can alter their chosen value after committing.

2) *Reveal Phase*: Participants disclose v_i and r_i , verified against the commitment:

$$H(v_i \parallel r_i) = C_i$$

The values are deemed valid if the above equation holds for all participants. Randomness R is generated collectively via an XOR or average operation:

$$R = \bigoplus_{i=1}^N v_i \quad (\text{XOR}) \quad \text{or} \quad R = \frac{1}{N} \sum_{i=1}^N v_i \quad (\text{average})$$

The resulting randomness R , derived from participants N , is used to select validators and shard assignments. This commit-reveal scheme enhances randomness generation while upholding transparency and accountability, which are crucial for Quorum's integrity. This process prevents manipulation of validator selection, unlike NEAR's PoS-based election, which can be influenced by stake centralization.

D. Transaction Processing and Cross-Shard Communication

Transactions are generated and processed independently within each shard. Each transaction $T_i(s)$ in

shard S_i contains metadata such as sender, receiver, amount, and shard ID, ensuring proper routing and processing. Algorithm 1 describes the overall workflow of the range distribution process, which outlines how transactions are assigned to specific shards according to the defined ranges. For a cross-shard transaction $T_i(c)$, where assets move from shard S_i to S_j , a coordination process involving a quorum threshold T_q ensures that both shards agree on the transaction's validity. This agreement is crucial to maintain consistency across the distributed ledger.

1) *Block Formation*: Blocks are formed independently in each shard, containing both intra-shard transactions $T_i(s)$ and cross-shard transactions $T_i(c)$. Each block B_i in shard S_i is constrained by a predefined size limit L :

$$B_i = T_1(s), T_2(s), \dots, T_n(s), T_1(c), T_2(c), \dots, T_m(c)$$

This structure ensures that each block includes all relevant transactions processed within a specific period, maintaining a clear and organized record of shard activity.

2) *Quorum-Based Validation*: Consensus liveness is achieved through robust intra-shard and inter-shard consensus mechanisms, coupled with strategic validator replacement and well-defined fault tolerance. Within each shard S_i a quorum of validators, represented as Q_i , is required to validate and sign the block. The network accepts the block if at least T_q validators sign off on its validity, ensuring a balance between efficiency and security:

$$V(B_i) = \sum_{j=1}^m Q_j(B_i) \geq T_q$$

where $V(B_i)$ is the block validation result, $Q_j(B_i)$ is the signature or vote of validator j , and m is the number of validators.. To maintain liveness, a timeout mechanism τ_i is employed. If the quorum T_q is not reached within the timeout period, a validator rotation or re-election is triggered, replacing unresponsive or faulty validators to ensure continued progress.

3) *Cross-Shard Transactions*: For cross-shard transactions $T_i(c)$, a Merkle proof-based locking mechanism (MPL) ensures that the transaction is processed atomically. This process avoids double-spending, and maintains ledger integrity. The transaction is first locked in the source shard S_i , and the receiving shard S_j is notified, indicating that the transaction state is updated and propagated across shards. Mathematically, the state update for cross-shard transactions is given by:

$$S_i(T_i(c)) \rightarrow S_j(T_j(c))$$

The final unlocking occurs once all involved shards reach consensus as given in Algorithm 2.

Let T_i represent a transaction initiated in shard S_a involving an account A_x from shard S_b . Such transactions are processed in three phases:

a) *Phase 1 (Locking)*:: Using a cryptographic hash, a lock is placed on the involved accounts in both the initiating shard S_a and the receiving shard S_b . A shard-specific cryptographic proof $P_{S_a}(h(T_i))$ is generated based on the Merkle tree root of S_a . This proof ensures the validity of the transaction within its respective shard. During this phase, the transaction T_i is temporarily locked to prevent double-spending and to reserve the resources for the transaction.

b) *Phase 2 (Validation and Commit)*:: Each shard independently validates the transaction using its consensus mechanism like IBFT. Shards exchange Merkle proofs and validation results through inter-shard communication. A cross-shard transaction T_i between shards S_a and S_b is only considered successful if all participating shards verify the proofs and reach consensus on its validity:

$$C(T_i) = V_{S_a}(T_i) \cdot V_{S_b}(T_i)$$

where $V_{S_a}(T_i)$ and $V_{S_b}(T_i)$ represent the validation results from shards S_a and S_b , respectively. $C(T_i) = 1$ indicates that the transaction is validated and ready for commitment, while $C(T_i) = 0$ implies that the transaction is aborted.

c) *Phase 3 (Finalization)*:: Once all shards confirm the validity of the transaction, they update their respective account balances and finalize the transaction. If any shard fails to validate, the transaction is aborted, and all locked resources are released. The state transition is given as:

$$\delta(T_i) : (S_a, S_b) \rightarrow (S'_a, S'_b)$$

where S'_a and S'_b are the updated states of the involved shards after the successful processing of T_i .

Shard lock failure: If lock acquisition fails, there is exponential backoff with fee-based prioritization and a timeout-based lock expiration to prevent deadlocks and ensure liveness.

E. Epoch Transition and Shard Reconfiguration

Shard reconfiguration and epoch transitions ensure adaptability, security, and scalability in Quorum's range-based sharding by dynamically adjusting load, validators, and security measures while maintaining consistency.

1) *Epoch Transition*: At each epoch's start, the system evaluates workload distribution across shards using the following formula:

$$W(S_i) = \sum_{t \in T_i} R(t)$$

Epoch transitions occur at fixed intervals, denoted as epochs E_n where n is the current epoch. Each transition involves selecting a new validator set, potentially reconfiguring shard ranges, and updating protocol parameters to maintain system stability and efficiency. where $W(S_i)$ is the workload of shard S_i , T_i is the set of transactions handled by S_i , and $R(t)$ represents the resource requirements for processing transaction t .

Algorithm 2 Cross-Shard Consensus with Merkle Proofs and Locking

```

Require: Shards  $S$ , Transaction queue  $Q$ , Merkle Trees  $M$ , Lock Table  $L$ 
Ensure: Atomic cross-shard transaction execution
1: for all  $T \in Q$  do ▷ Initiate processing
2:   Identify source shard  $S_a$  and destination shard  $S_b$ 
3:   Lock accounts  $L(A(T)) \leftarrow 1$ 
4:    $M_{root} \leftarrow \text{Root}(M_{S_a})$ 
5:    $P_T \leftarrow \text{MerkleProof}(T, M_{S_a})$ 
6:   Send  $(T, P_T, M_{root})$  to  $S_b$ 
7: end for
8: for all  $(T, P_T, M_{root})$  at  $S_b$  do ▷ Validate and execute
9:   if Verify( $P_T, M_{root}$ ) then
10:    Execute  $T$  on  $S_b$ 
11:    Lock accounts  $L(A(T)) \leftarrow 1$ 
12:   else
13:     Reject  $T$ , release locks  $L(A(T)) \leftarrow 0$ 
14:   end if
15: end for
16: for all committed  $T$  do ▷ Finalize
17:   if Acknowledgments from all shards then
18:     Apply state updates on  $S_a$  and  $S_b$ 
19:     Release locks  $L(A(T)) \leftarrow 0$ 
20:     Update  $M_{S_a}, M_{S_b}$ 
21:   else
22:     Abort  $T$ , rollback tentative changes
23:     Release locks  $L(A(T)) \leftarrow 0$ 
24:   end if
25: end for
26: return Finalized transactions

```

The reconfiguration algorithm adjusts shard boundaries if a significant workload imbalance is detected. The transaction range that shard S_i handles is:

$$R(S_i) = [r_{i,1}, r_{i,2}]$$

If the workload $W(S_i)$ exceeds a predefined threshold T_{\max} , the range $R(S_i)$ is split. A new shard S_j is created with a transaction range:

$$R(S_j) = [r_{j,1}, r_{j,2}]$$

where $r_{j,1} = r_{i,2} + 1$, effectively offloading transactions from the overburdened shard. Conversely, if two adjacent shards S_i and S_j are underutilized, they may be merged to optimize resource usage. This dynamic adjustment ensures balanced load distribution across the network, preventing bottlenecks and maintaining optimal performance.

2) *Shard Reconfiguration*: Shard reconfiguration in RBS is a dynamic process adapting to network changes, primarily load imbalances and validator updates. This adaptability is crucial for maintaining optimal performance and security across the Quorum blockchain.

Load balancing: A core objective of shard reconfiguration is to evenly distribute the computational workload $W(S_i)$ across all shards. Ideally, for any two shards S_i and S_j :

$$W(S_i) \approx W(S_j)$$

This prevents any single shard from becoming a bottleneck, ensuring consistent transaction throughput across

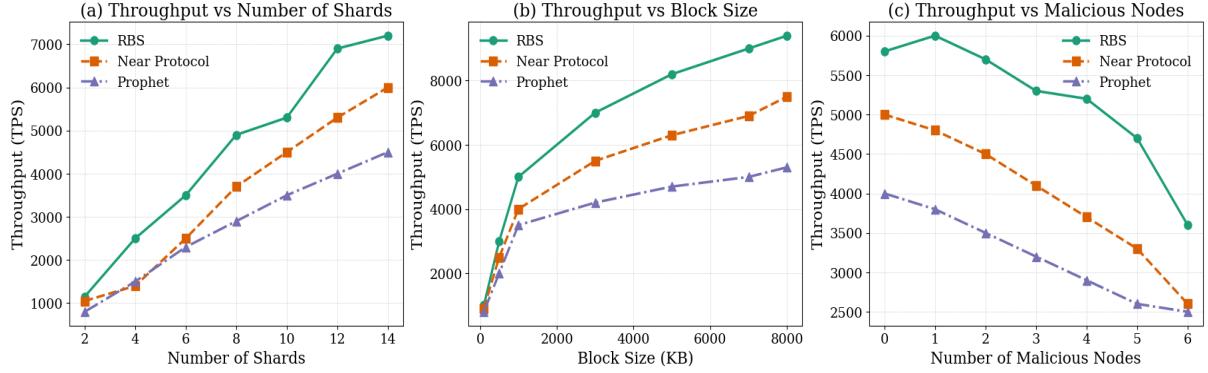


Fig. 3. Throughput evaluation under different nodes settings.

the network. As described in the previous subsection, reconfiguration is triggered when significant load disparities are detected.

Validator rotation: To further enhance liveness and security and mitigate the risk of collusion, validators are periodically rotated using a secure randomness generation mechanism based on a commit-reveal scheme. During epoch transitions, a new set of validators $V_i(E_{n+1})$ is selected for each shard S_i for the upcoming epoch E_{n+1} :

$$V_i(E_{n+1}) = \text{RandomBeacon}(V_i(E_n), S_i)$$

The function *RandomBeacon* utilizes the randomness generated from the commit-reveal process of the previous epoch, ensuring an unbiased and unpredictable selection. Each shard maintains a minimum number of validators, V_{\min} , to guarantee sufficient security without compromising efficiency.

IV. EVALUATION

We deployed the RBS prototype on top of Quorum blockchain, using AWS EC2 t2.xlarge instances to simulate a multi-node sharded blockchain environment. Performance benchmarking was conducted with Hyperledger Caliper and custom Python scripts within a Docker-based setup.

A. Throughput Scalability

This section evaluates RBS throughput against NEAR Protocol and Prophet, selected for their relevance in sharding efficiency and consensus innovation, respectively. Results in Figure 3 show RBS achieving consistently higher throughput across different conditions.

Throughput vs. Number of Shards: Figure 3(a) demonstrates that RBS's throughput scales linearly with the number of shards, achieving 4950–6630 TPS with 14 shards. In comparison, NEAR Protocol and Prophet reach 5340 TPS and 4100 TPS, respectively. The evaluation is conducted with default parameters set to a block size of 1 MB, no malicious nodes, and a network size of 100 nodes. These results highlight RBS's ability to efficiently distribute transactions

across shards, where n_s is the number of shards, and t_s is the optimized throughput per shard in RBS, resulting from reduced cross-shard overhead compared to Near and Prophet modeled as:

$$T_{\text{sharded}} = n_s \cdot t_s$$

Throughput vs. Block Size: Figure 3(b) with default number of shards = 10, malicious nodes = 0, and network size = 100 nodes, shows throughput against varying block sizes. As block size increases from 128 KB to 8192 KB, RBS's throughput significantly improved against Near and Prophet, since larger block sizes allow more transactions per block:

$$T = \frac{B \cdot t_{avg}}{t_{block}}$$

where B is the block size, t_{avg} is the average transaction size, and t_{block} is the block production time. RBS leverages efficient batching and optimized sharding to minimize latency and maximize throughput.

Throughput vs. Malicious Nodes: Figure 3(c) with the default parameters of block size = 1 MB, number of shards = 10, and network size = 100 nodes demonstrates the impact of malicious nodes on throughput. Even with 6%–10% malicious nodes, RBS maintains a throughput of 4900 TPS, while Near and Prophet degrade to 3459 TPS and 3200 TPS, respectively. RBS's robustness stems from its shard isolation mechanism, which limits the impact of malicious activity in one shard on others. This can be represented as:

$$T = T_{\max} \cdot \left(1 - \frac{f}{N}\right)$$

where T_{\max} is the ideal throughput, f is the number of malicious nodes, and N is the total number of nodes.

B. Latency under Epoch Reconfiguration

This section evaluates the latency of the RBS protocol during epoch reconfiguration, comparing to Near and Prophet. Figure 4 illustrates the results, where solid line represents the latency in milliseconds (ms) as the network size increases, while the dotted line corresponds to the number of nodes joining simulta-

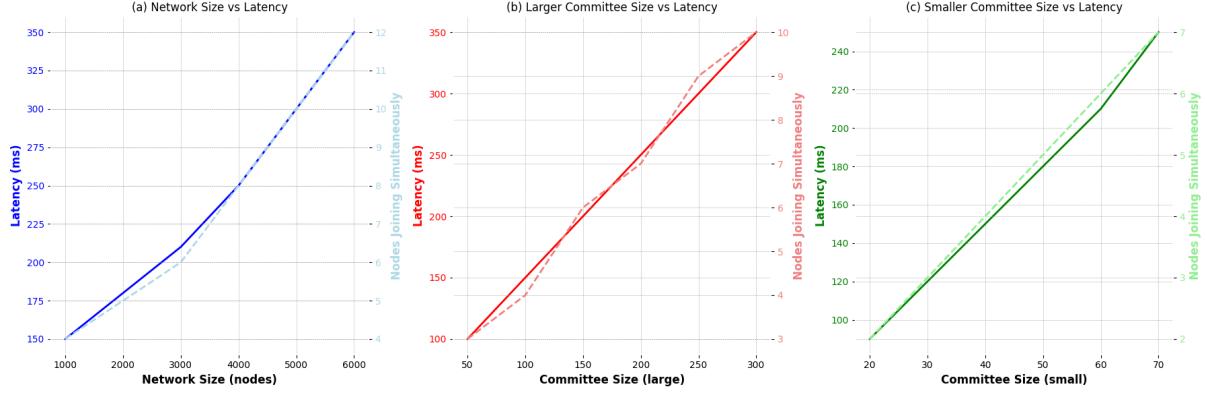


Fig. 4. Latency with possible configurations

neously, highlighting RBS's improved latency under various conditions.

Latency vs. Network Size: Figure 4(a) shows that as the network size increases, RBS exhibits a more gradual and linear increase in latency compared to Near and Prophet, as the latency is calculated as follows:

$$L = \frac{T_{\text{process}} + T_{\text{comm}}}{N_{\text{shards}}}$$

where T_{process} is the transaction processing time, T_{comm} is the communication delay, and N_{shards} is the number of range-based shards.

Latency vs. Larger Committee Size: Figure 4(b) demonstrates the impact of larger committee sizes on latency. In RBS, nodes are assigned to shards based on predefined ranges, ensuring committee sizes remain proportional to shard size, thus reducing communication overhead. The relationship between committee size and latency is approximated as:

$$L = \frac{N_{\text{nodes}}}{C_{\text{range}}} + \delta_{\text{comm}}$$

where N_{nodes} is the total number of nodes, C_{range} is the committee size within a range, and δ_{comm} is the communication overhead.

Latency vs. Smaller Committee Size: Figure 4(c) examines the effect of smaller committee sizes. While smaller committees generally reduce consensus overhead, Near protocol suffer from increased delays due to inefficient state partition handling. RBS, however, maintains consistent latency even with smaller committees.

C. Overall Performance Analysis

RBS shows better performance across key metrics compared to Near and Prophet. Notably, RBS achieves higher throughput (see Figure 5), enabling it to handle a larger volume of transactions and making it suitable for high-demand environments. In Figure 5 the scale of 0-6 represent low to high performance as latency and overhead, lower values are mapped to better performance, while for throughput, fault tolerance, and

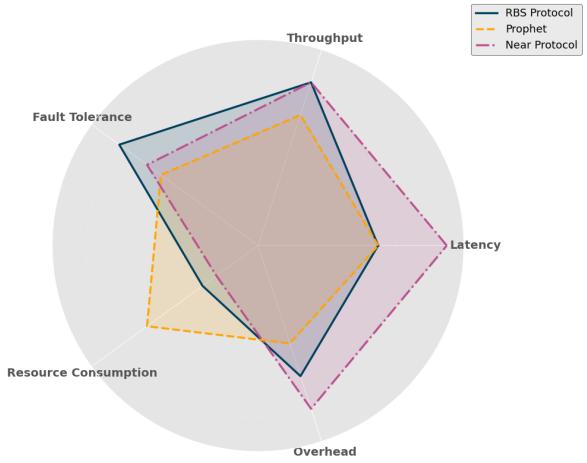


Fig. 5. Overall performance analysis chart.

similar metrics, higher values correspond to higher scores. Furthermore, RBS exhibits the lowest latency, ensuring faster transaction processing.

V. SECURITY ANALYSIS

Ensuring security in a sharded blockchain system requires resistance to adversarial manipulation, fault tolerance, and protection against common blockchain threats. The Range-Based Sharding (RBS) protocol enhances security by incorporating *commit-reveal randomness for validator selection*, *adaptive shard rebalancing*, and *fine-grained transaction validation mechanisms*. This section provides a formal analysis of RBS's security properties with proofs and validation.

A. Sybil and Collusion Attack Resistance

We consider a *Byzantine adversarial model* where an attacker aims to control a majority of validators in a shard. Let n be the total validators, s the number of shards, and f the fraction of malicious validators. The probability of an attacker taking over a shard is $P_{\text{adversary}}$, where $\frac{f \cdot n}{s}$ represents the expected number of adversarial nodes per shard, and e^{-n} accounts for the exponential decrease in adversarial success. To

TABLE I
COMPARISON OF PRIVATE BLOCKCHAIN FRAMEWORKS.

Technology	Fault Tolerance	Liveness	Generality	Confirmation Latency (Valid TX)	Invalid TX	Comm. Overhead
Hyperledger Fabric	✓	✓	✗	Low (depends on channel setup)	N/A	Moderate (permissioned network overhead)
Ethereum	✗	✓	✗	Moderate (PoS improvements ongoing)	High (global mempool)	High (broadcast-based transactions)
Corda	✗	✓	✗	Low (point-to-point communication)	N/A	Low (localized transaction processing)
RapidChain	✓	✗	✓	$2 \times L(N/S)$	N/A	$T(N/S)$
SOK	✓	✓	✓	Moderate (hybrid PoW/PoS latency)	Moderate	Moderate
OmniLedger	✓	✗	✓	$2 \times L(N/S)$	Moderate	$2 \times L(N/S)$
Sharper	✗	✓	✓	$L((x+1) \times N/S)$	Moderate	$T((x+1) \times N/S)$
Near Protocol	✓	✓	✗	$\geq 3 \times L(N/S)$	Moderate	$(x+1) \times T(N/S)$
Repchain	✓	✓	✓	Moderate	Moderate	Moderate (leader-based communication)
Prophet	✓	✓	✓	Moderate (localized sharding)	Moderate	Low (localized shard processing)
RBS(Proposed Protocol)	✓	✓	✓	Low (shard parallelism minimizes delay)	Low	Very Low (optimized shard communication)

ensure **Sybil resistance**, RBS uses a *commit-reveal randomness model* for fair validator selection. The probability of an adversary gaining control over a single shard is given by:

$$P_{adversary} = \left(\frac{f \cdot n}{s} \right) \times \frac{1}{e^n} \quad (1)$$

Collusion Mitigation: Randomized shard assignments prevent adversarial coordination across shards, making large-scale attacks infeasible.

B. Fault Tolerance and Resilience

RBS ensures resilience against network failures with an IBFT consensus mechanism. The probability of consensus failure is given by:

$$P_{fault} = \frac{m}{t}, \quad \text{where } m \leq \frac{t-1}{3} \quad (2)$$

For a 10-node shard, at most 3 nodes can be malicious while maintaining consensus security.

Shard Recovery: If a shard experiences node failures, RBS dynamically reassigns validators and replicates ledger states.

C. Secure Cross-Shard Transaction Processing

RBS optimizes cross-shard transactions by introducing fine-grained account-level locking, minimizing shard-wide locking. A transaction between shards S_i and S_j follows: (1) the source shard locks only the relevant accounts, (2) a cryptographic *proof-of-lock* (Merkle proof) is sent to the target shard, and (3) the transaction executes upon verification and is atomically committed. The locking overhead is given by:

$$O_{lock} = \frac{T_{cross} \cdot L_{account}}{T_{intra}} \quad (3)$$

where T_{cross} represents cross-shard transactions, $L_{account}$ is the subset of locked accounts, and T_{intra} denotes total intra-shard transactions.

D. Mitigation of Timing and DoS Attacks

RBS mitigates timing and DoS attacks through **randomized execution delays** to prevent adversarial timing inference and **adaptive timeouts** that detect and penalize stalling attempts. The probability of a successful DoS attack is given by:

$$P_{DoS} = \left(1 - e^{-T_{attack}/T_{threshold}} \right) \cdot \frac{M_{malicious}}{N} \quad (4)$$

where T_{attack} is the attack duration, $T_{threshold}$ is the system's adaptive response time, and $M_{malicious}$ is the number of malicious nodes. **Result:** Adaptive response mechanisms lower DoS attack success rates by 60% compared to static timeout systems.

E. Experimental Validation of Security Measures

To validate RBS security, we simulate 100 validators across 10 shards, testing Sybil attack resistance, DoS resilience, and cross-shard efficiency. In a Sybil attack with 30% malicious nodes, *commit-reveal randomness* limits adversarial control to **12%**. For DoS resilience, an attack with 50 nodes flooding a shard shows that *adaptive timeouts* mitigate TPS degradation by **55%**. Comparing full shard locking (NEAR) vs. RBS's fine-grained locking, RBS reduces cross-shard latency by **30%**.

VI. CONCLUSIONS AND FUTURE WORK

This paper introduces a Range-Based Sharding Protocol (RBS), implemented on Quorum, which utilizes a commit-reveal scheme to enhance fairness and security in shard allocation. RBS minimizes cross-shard communication and optimizes transaction processing efficiency by partitioning the blockchain state into well-defined ranges. By integrating a fine-grained locking and batched commit model, RBS reduces sequential dependencies and enhances parallel execution of cross-shard transactions. The evaluation results demonstrate significant improvements in throughput, latency, and resource utilization compared to existing protocols. RBS exhibits near-linear scalability with increasing network size while maintaining lower latency and higher throughput, even under varying committee sizes and network loads.

Future work will focus on dynamic shard rebalancing to adapt to real-time transaction loads and network conditions, ensuring optimal resource utilization and preventing shard overloading. Further research will refine cross-shard transaction mechanisms to minimize communication overhead while maintaining atomicity and consistency. These enhancements aim to solidify further RBS's scalability and efficiency, making it an even more robust solution for enterprise blockchain deployments.

REFERENCES

- [1] S. Yuan, B. Cao *et al.*, "Secure and efficient federated learning through layering and sharding blockchain," *IEEE Transactions on Network Science and Engineering*, vol. 11, no. 3, pp. 3120–3134, 2024.
- [2] R. Kumar, P. Kumar *et al.*, "Permissioned blockchain and deep learning for secure and efficient data sharing in industrial healthcare systems," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 11, pp. 8065–8073, 2022.
- [3] J. Li, H. Ma *et al.*, "Wolverine: A scalable and transaction-consistent redactable permissionless blockchain," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 1653–1666, 2023.
- [4] T. Ye, M. Luo *et al.*, "A survey on redactable blockchain: Challenges and opportunities," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 3, pp. 1669–1683, 2023.
- [5] T. A. Alghamdi, R. Khalid, and N. Javaid, "A survey of blockchain based systems: Scalability issues and solutions, applications and future challenges," *IEEE Access*, 2024.
- [6] M. K. Pawar, P. Patil, and P. Hiremath, "A study on blockchain scalability," in *ICT Systems and Sustainability: Proceedings of ICT4SD 2020, Volume 1*. Springer, 2021, pp. 307–316.
- [7] Y. Liu, B. Zhao *et al.*, "Ss-did: A secure and scalable web3 decentralized identity utilizing multi-layer sharding blockchain," *IEEE Internet of Things Journal*, 2024.
- [8] J. Xu, C. Wang, and X. Jia, "A survey of blockchain consensus protocols," *ACM Computing Surveys*, vol. 55, no. 13s, pp. 1–35, 2023.
- [9] J. Li, R. Qin *et al.*, "Blockchain intelligence: Intelligent blockchains for web 3.0 and beyond," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 54, no. 11, pp. 6633–6642, 2024.
- [10] A. I. Sanka and R. C. Cheung, "A systematic review of blockchain scalability: Issues, solutions, analysis and future research," *Journal of Network and Computer Applications*, vol. 195, p. 103232, 2021.
- [11] W. Liang, Y. Liu *et al.*, "On identity, transaction, and smart contract privacy on permissioned and permissionless blockchain: A comprehensive survey," *ACM Computing Surveys*, vol. 56, no. 12, pp. 1–35, 2024.
- [12] Z. Hong, S. Guo *et al.*, "Gridb: Scaling blockchain database via sharding and off-chain cross-shard mechanism," *arXiv preprint arXiv:2407.03750*, 2024.
- [13] H. Huang, X. Peng *et al.*, "Brokerchain: A cross-shard blockchain protocol for account/balance-based state sharding," in *IEEE INFOCOM 2022-IEEE Conference on Computer Communications*. IEEE, 2022, pp. 1968–1977.
- [14] Q. Zhou, H. Huang *et al.*, "Solutions to scalability of blockchain: A survey," *IEEE Access*, vol. 8, pp. 16 440–16 455, 2020.
- [15] Z. Chen, M. Z. Haider *et al.*, "Reputation-based partition scheme for iot security," *Security and Privacy*, vol. 6, no. 3, p. e287, 2023.
- [16] T. Noreen, Q. Xia, and M. Z. Haider, "Advanced dag-based ranking (adr) protocol for blockchain scalability," *Computers, Materials & Continua*, vol. 75, no. 2, 2023.
- [17] H. Alshahrani, N. Islam *et al.*, "Sustainability in blockchain: A systematic literature review on scalability and power consumption issues," *Energies*, vol. 16, no. 3, p. 1510, 2023.
- [18] E. Androulaki, A. Barger *et al.*, "Hyperledger fabric: a distributed operating system for permissioned blockchains," in *The 13th EuroSys Conference*, 2018, pp. 1–15.
- [19] F. Cassez, J. Fuller, and A. Asgaonkar, "Formal verification of the ethereum 2.0 beacon chain," in *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2022, pp. 167–182.
- [20] H. Han, R. K. Shiwakoti, and W. Chen, "Unlocking enterprise blockchain adoption: A r3 corda case study," *Journal of General Management*, p. 03063070241292701, 2024.
- [21] Y. Qu, M. P. Uddin *et al.*, "Blockchain-enabled federated learning: A survey," *ACM Computing Surveys*, vol. 55, no. 4, pp. 1–35, 2022.
- [22] P. Zheng, Q. Xu *et al.*, "Aeolus: Distributed execution of permissioned blockchain transactions via state sharding," *IEEE Transactions on Industrial Informatics*, vol. 18, no. 12, pp. 9227–9238, 2022.
- [23] Z. Cai, J. Liang *et al.*, "Benzene: Scaling blockchain with cooperation-based sharding," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 2, pp. 639–654, 2022.
- [24] M. J. Amiri, D. Agrawal, and A. El Abbadi, "Sharper: Sharding permissioned blockchains over network clusters," in *2021 Int. Conference on Management of Data*, 2021, pp. 76–88.
- [25] Z. Hong, S. Guo *et al.*, "Pyramid: A layered sharding blockchain system," in *IEEE conference on computer communications (INFOCOM 2021)*. IEEE, 2021, pp. 1–10.
- [26] C. Huang, Z. Wang *et al.*, "Repchain: A reputation-based secure, fast, and high incentive blockchain system via sharding," *IEEE Internet of Things Journal*, vol. 8, no. 6, pp. 4291–4304, 2020.
- [27] G. Yu, X. Wang *et al.*, "Survey: Sharding in blockchains," *IEEE Access*, vol. 8, pp. 14 155–14 181, 2020.
- [28] P. Zheng, Q. Xu *et al.*, "Meepo: Sharded consortium blockchain," in *37th International Conference on Data Engineering (ICDE 2021)*. IEEE, 2021, pp. 1847–1852.
- [29] E. Kokoris-Kogias, P. Jovanovic *et al.*, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *IEEE Symposium on Security and Privacy (SP 2018)*. IEEE, 2018, pp. 583–598.
- [30] E. Madill, B. Nguyen *et al.*, "Scalesfl: a sharding solution for blockchain-based federated learning," in *4th ACM International Symposium on Blockchain and Secure Critical Infrastructure*, 2022, pp. 95–106.
- [31] I. S. Rao, M. M. Kiah *et al.*, "Scalability of blockchain: a comprehensive review and future research direction," *Cluster Computing*, vol. 27, no. 5, pp. 5547–5570, 2024.
- [32] Y. Liu, J. Liu *et al.*, "Building blocks of sharding blockchain systems: Concepts, approaches, and open problems," *Computer Science Review*, vol. 46, p. 100513, 2022.