

DYNASHARD: Secure and Adaptive Blockchain Sharding Protocol with Hybrid Consensus and Dynamic Shard Management

Ao Liu, Jing Chen, Kun He, Ruiying Du, Jiahua Xu, Cong Wu, Yebo Feng, Teng Li, and Jianfeng Ma

Abstract—Blockchain sharding has emerged as a promising solution to the scalability challenges in traditional blockchain systems by partitioning the network into smaller, manageable subsets called shards. Despite its potential, existing sharding solutions face significant limitations in handling dynamic workloads, ensuring secure cross-shard transactions, and maintaining system integrity. To address these gaps, we propose DynaShard, a dynamic and secure cross-shard transaction processing mechanism designed to enhance blockchain sharding efficiency and security. DynaShard combines adaptive shard management, a hybrid consensus approach, and an efficient state synchronization and dispute resolution protocol. Our performance evaluation, conducted using a robust experimental setup with real-world network conditions and transaction workloads, demonstrates DynaShard's superior throughput, reduced latency, and improved shard utilization compared to the Fast Transaction Scheduling in Blockchain Sharding (FTSBS) method. Specifically, DynaShard achieves up to a 42.6% reduction in latency and a 78.77% improvement in shard utilization. These results highlight DynaShard's ability to outperform state-of-the-art sharding methods, ensuring scalable and resilient blockchain systems. We believe that DynaShard's innovative approach will significantly impact future developments in blockchain technology, paving the way for more efficient and secure distributed systems.

Index Terms—Blockchain sharding, cross-shard transactions, secure consensus, state synchronization, scalability

I. INTRODUCTION

Blockchain technology has emerged as a groundbreaking innovation that has transformed the landscape of digital transactions, offering a decentralized, transparent, and secure framework for various applications, including IoT-based systems [1–6]. The core principles of blockchain, such as immutability, transparency, and distributed consensus, have the potential to revolutionize IoT networks by enhancing security, data integrity, and trust among devices and participants [7]. However, the scalability of blockchain systems remains a significant challenge, particularly for IoT applications that require high transaction throughput and low latency [8]. The increasing

number of IoT devices and transactions leads to longer confirmation times, higher transaction fees, and reduced overall system performance [9]. To address this scalability challenge, researchers have proposed various solutions, including off-chain scaling techniques like payment channels [10–12] and sidechains [13–15], as well as on-chain scaling approaches like sharding [16, 17]. By improving scalability, these solutions can enable the seamless integration of blockchain technology with IoT systems, ensuring efficient and secure management of IoT data and transactions.

Existing efforts. Blockchain sharding has gained significant attention as a promising on-chain scaling solution that aims to improve the throughput and latency of blockchain systems by partitioning the network into smaller shards, each responsible for processing a subset of transactions in parallel [16]. By distributing the transaction processing workload across multiple shards, blockchain sharding enables the system to scale horizontally, allowing for a higher transaction throughput and lower confirmation times [18]. Several blockchain sharding frameworks have been proposed in recent years, each addressing specific aspects of the sharding process. For instance, Elastico [19] introduces a secure sharding protocol that utilizes a distributed randomness generation process for shard formation and a Byzantine fault-tolerant consensus mechanism within each shard. OmniLedger [20] builds upon Elastico by incorporating a more efficient cross-shard transaction processing mechanism based on atomic commits and a ledger pruning technique to reduce storage overhead. RapidChain [21] further improves the scalability of sharded blockchains by introducing a fast and efficient cross-shard transaction verification scheme that leverages intra-shard consensus and inter-shard gossiping. These frameworks have laid the foundation for the development of scalable and efficient blockchain sharding solutions.

Research gap. Despite the progress made in blockchain sharding, several limitations and research gaps still exist, presenting opportunities for further improvement. One major challenge is the lack of dynamic and adaptive mechanisms for managing shards based on the system's workload [22, 23]. Most existing sharding frameworks rely on static shard configurations, which can lead to suboptimal resource utilization and performance, especially in the presence of fluctuating transaction volumes and network conditions [24]. Another significant challenge lies in ensuring the security and atomicity of cross-shard transactions [25]. Malicious actors may attempt to exploit the distributed nature of sharded systems by launching attacks such as double-spending, replay attacks,

Ao Liu, Jing Chen, Kun He, and Ruiying Du are with Key Laboratory of Aerospace Information Security and Trusted Computing, Ministry of Education, School of Cyber Science and Engineering, Wuhan University, China. Email: {liuao, chenjing, hekun, duraying}@whu.edu.cn.

Jiahua Xu is with University College London, UK. Email: jiahua.xu@ucl.ac.uk.

Cong Wu and Yebo Feng are with College of Computing and Data Science, Nanyang Technological University, Singapore. Email: {cong.wu, yebo.feng}@ntu.edu.sg.

Teng Li and Jianfeng Ma is with School of Cyber Engineering, Xidian University, Xi'an, China. Email: litengxidian@gmail.com, jfma@mail.xidian.edu.cn.

or shard-level collusion [26]. Existing cross-shard transaction processing techniques, such as two-phase commit protocols [27] and asynchronous consensus [23], provide some level of protection against these attacks, but they often come at the cost of increased complexity and communication overhead [22]. Furthermore, there is a lack of comprehensive frameworks that integrate shard reconfiguration, state synchronization, and dispute resolution mechanisms to ensure the overall security and efficiency of sharded blockchain systems [28, 29]. Designing a holistic solution that addresses these challenges while maintaining the core principles of decentralization, transparency, and security is a non-trivial task that requires careful consideration of various trade-offs and design choices.

DYNASHARD. To bridge the research gap, we propose DYNASHARD, a dynamic and secure cross-shard transaction processing mechanism. DYNASHARD combines adaptive shard management, secure cross-shard transaction processing, and efficient state synchronization and dispute resolution to enhance scalability and resilience in blockchain systems. By dynamically adjusting shard configurations based on workload, it optimizes resource utilization and performance, adapting to varying transaction volumes and network conditions. It employs a hybrid consensus approach that integrates intra-shard and inter-shard mechanisms to minimize coordination overhead while ensuring transaction integrity and consistency.

Designing a dynamic and secure cross-shard transaction processing mechanism involves several challenges. **C1:** Effective shard management requires monitoring workload and resource usage to make informed decisions on splitting or merging shards, necessitating an understanding of system dynamics and future transaction predictions. **C2:** Secure and efficient cross-shard transaction processing demands a protocol ensuring atomicity and consistency while minimizing overhead, incorporating novel consensus mechanisms, cryptographic techniques, and scalable data structures. **C3:** Robust state synchronization and dispute resolution require decentralized mechanisms to detect and resolve inconsistencies and conflicts, integrating insights from distributed systems, cryptography, game theory, and economics. DYNASHARD addresses these challenges by introducing a comprehensive framework that dynamically adjusts shard configurations based on transaction volume and resource usage, employs a hybrid consensus approach combining lightweight global consensus with parallel intra-shard processes, and utilizes Merkle trees alongside a decentralized dispute resolution protocol to maintain consistency and resolve conflicts in a trustless manner.

Novelty. The key novelty of DYNASHARD lies in its holistic and adaptive approach to blockchain sharding, distinguishing it from existing solutions. Unlike previous works that focus on specific aspects such as shard formation [19], cross-shard transaction processing [27], or consensus mechanisms [21], DYNASHARD integrates all these components into a cohesive and dynamic system. The novelty of DYNASHARD includes: (i) continuously monitoring and adjusting shard configurations based on system workload, offering an efficient and flexible sharding scheme adaptable to the evolving demands of real-world blockchain applications; (ii) employing a hybrid consensus approach to address the challenge of secure and

atomic cross-shard transaction processing, balancing global coordination with local processing efficiency; and (iii) integrating adaptive shard management, secure cross-shard transaction processing, and efficient state synchronization and dispute resolution techniques.

In summary, this paper makes the following contributions:

- We propose DYNASHARD, an adaptive shard management mechanism that dynamically adjusts shard configurations based on transaction volume and resource usage, ensuring optimal performance and resource utilization.
- We propose a secure and atomic cross-shard transaction protocol using a hybrid consensus approach. This integrates lightweight global consensus with parallel intra-shard processes, reducing overhead while maintaining transaction integrity and consistency.
- We develop an efficient shard state synchronization mechanism based on Merkle trees. This mechanism maintains consistency across shards and incorporates a decentralized dispute resolution protocol to resolve potential conflicts in a trustless and resilient manner.
- We perform comprehensive evaluation of DYNASHARD through theoretical analysis and experimental simulations. Results demonstrate improvements in throughput, latency, and shard utilization compared to existing solutions, validating its effectiveness and robustness.

II. RESEARCH BACKGROUND

This section briefs blockchain sharding, challenges .

A. Blockchain Sharding

Blockchain sharding is a technique designed to address scalability challenges in blockchain systems by partitioning the network into smaller subsets called shards. Each shard processes a portion of the overall transactions, enabling parallel execution and improved throughput. Let $S = \{s_1, s_2, \dots, s_n\}$ represent the set of shards, where n is the total number of shards. Transactions within each shard s_i are processed independently, while cross-shard transactions require coordination between shards. Sharding allows blockchain systems to scale with the number of shards, improving transaction throughput (TPS) and reducing latency (L) [16].

Several sharding frameworks have been proposed. Elastico [19] introduced secure shard formation using proof-of-work and Byzantine fault tolerance (BFT) within each shard. OmniLedger [20] built on this by improving cross-shard transaction processing and ledger pruning to reduce storage needs. RapidChain [21] enhanced scalability through fast cross-shard verification using intra-shard consensus and inter-shard gossiping. These frameworks represent advancements in blockchain sharding aimed at boosting system efficiency and scalability.

B. Challenges in Blockchain Sharding

One major challenge in blockchain sharding is maintaining security in the presence of malicious actors. Shard-level attacks, such as single-shard takeovers [16] and cross-shard

double-spending [26], can compromise system integrity. Solutions to these risks include random shard assignment [19], periodic shard reconfiguration [20], and fraud proofs [27]. Efficient cross-shard transaction handling is another challenge, with protocols like two-phase commit [27] and asynchronous consensus [23] ensuring transaction consistency, though they introduce complexity and overhead. Recent advancements focus on enhancing cross-shard transaction efficiency. For example, SharPer [25] secures transactions using threshold signatures and multi-party computation. Similarly, Qin et al. [30] proposes a compact verification scheme using Merkle proofs and SNARKs. These innovations help address performance limitations, paving the way for more scalable, secure blockchain systems.

III. DYNASHARD

This section presents overview and detail each module.

A. System Overview

DYNASHARD provides a dynamic and secure framework for efficient blockchain sharding, consisting of three main modules: adaptive shard management, secure cross-shard transaction processing, and shard state synchronization with dispute resolution, as in Figure 1. The adaptive shard management module continuously monitors transaction volume (v_i) and resource utilization (u_i) for each shard, adjusting shard configurations based on predefined thresholds (τ_s for splitting and τ_m for merging). Managing committee, selected through a secure random process from various shards, ensures decentralization and avoids collusion. Shard adjustments are further secured through threshold-based verification, requiring consensus among shard members to prevent unauthorized changes, maintaining system's decentralization and security.

The secure cross-shard transaction processing module ensures atomicity and security for transactions across multiple shards using a hybrid consensus mechanism that combines global and intra-shard consensus. This approach employs threshold signatures and multi-party computation to prevent double-spending attacks. Shard state synchronization and dispute resolution are handled through a Merkle tree-based structure, allowing fast verification and incremental updates. Disputes are resolved in a decentralized manner using game-theoretic incentives, ensuring valid transactions are processed and malicious actors are penalized. Overall, DYNASHARD effectively balances scalability, security, and decentralization.

B. Adaptive Shard Management

The adaptive shard management mechanism is designed to optimize the performance and resource utilization of the sharded blockchain system by dynamically adjusting the number and configuration of shards based on the system's workload. Let $S = \{s_1, s_2, \dots, s_n\}$ denote the set of shards in the system, where n is the total number of shards. Each shard s_i is characterized by its transaction volume v_i and resource utilization u_i . Resource utilization u_i refers to the percentage of available computational, storage, and network

resources being used by shard s_i . To manage these metrics effectively, we define two threshold parameters, τ_s and τ_m , representing the splitting and merging thresholds, respectively. The splitting threshold τ_s is chosen based on system capacity and workload expectations, typically ranging between 60% and 90% resource utilization. Similarly, the merging threshold τ_m is set between 20% and 40% to ensure underutilized shards are merged. These thresholds are selected through empirical analysis, balancing performance with system stability, and include cooldown periods to prevent immediate re-triggering of operations.

The adaptive shard management mechanism operates as follows: For each shard $s_i \in S$, the system continuously monitors its transaction volume v_i and resource utilization u_i . If the transaction volume v_i or resource utilization u_i exceeds the splitting threshold τ_s , i.e., $v_i > \tau_s$ or $u_i > \tau_s$, a shard splitting process is initiated. This process involves dividing s_i into multiple smaller shards $\{s_{i1}, s_{i2}, \dots, s_{ik}\}$, ensuring that the resulting shards maintain full functionality and consensus security. The goal is to distribute the workload w_i and resource demand d_i evenly across the new shards such that $\sum_{j=1}^k w_{ij} \approx w_i$ and $\sum_{j=1}^k d_{ij} \approx d_i$, where w_{ij} and d_{ij} represent the workload and resource demand of the new shards, respectively. A workload-aware rebalancing algorithm considers transaction patterns \mathcal{T} and resource requirements \mathcal{R} to ensure balanced resource utilization.

Conversely, if the transaction volume v_i and resource utilization u_i fall below the merging threshold τ_m for multiple consecutive epochs, i.e., $v_i < \tau_m$ and $u_i < \tau_m$, a shard merging process is initiated. This process combines s_i with other underutilized shards $\{s_{j1}, s_{j2}, \dots, s_{jl}\}$, consolidating resources and reducing the overall system overhead while maintaining consensus security. The combined shard s_c will have a transaction volume $v_c = \sum_{k=1}^l v_{jk}$ and resource utilization $u_c = \sum_{k=1}^l u_{jk}$. After the merging or splitting process, nodes and accounts are redistributed across the newly formed shards using the workload-aware rebalancing algorithm. This redistribution aims to further optimize the system's performance and resource utilization, ensuring balanced and efficient operation across all shards while avoiding the potential cycle of shard splitting and merging.

Algorithm 1 outlines the adaptive shard management process in DYNASHARD, which dynamically adjusts shard configurations based on transaction volume and resource utilization. When a shard exceeds the splitting threshold (τ_s), it is split into smaller shards, with transactions allocated using a load-balancing algorithm based on account activity to evenly distribute the workload and minimize cross-shard interactions. Conversely, when transaction volume and resource utilization fall below the merging threshold (τ_m) for multiple consecutive epochs, underutilized shards are merged to consolidate resources and optimize system performance. After splitting or merging, nodes, accounts, and transactions are redistributed using a Greedy Load Balancing algorithm to ensure efficient resource utilization and reduce cross-shard transaction overhead.

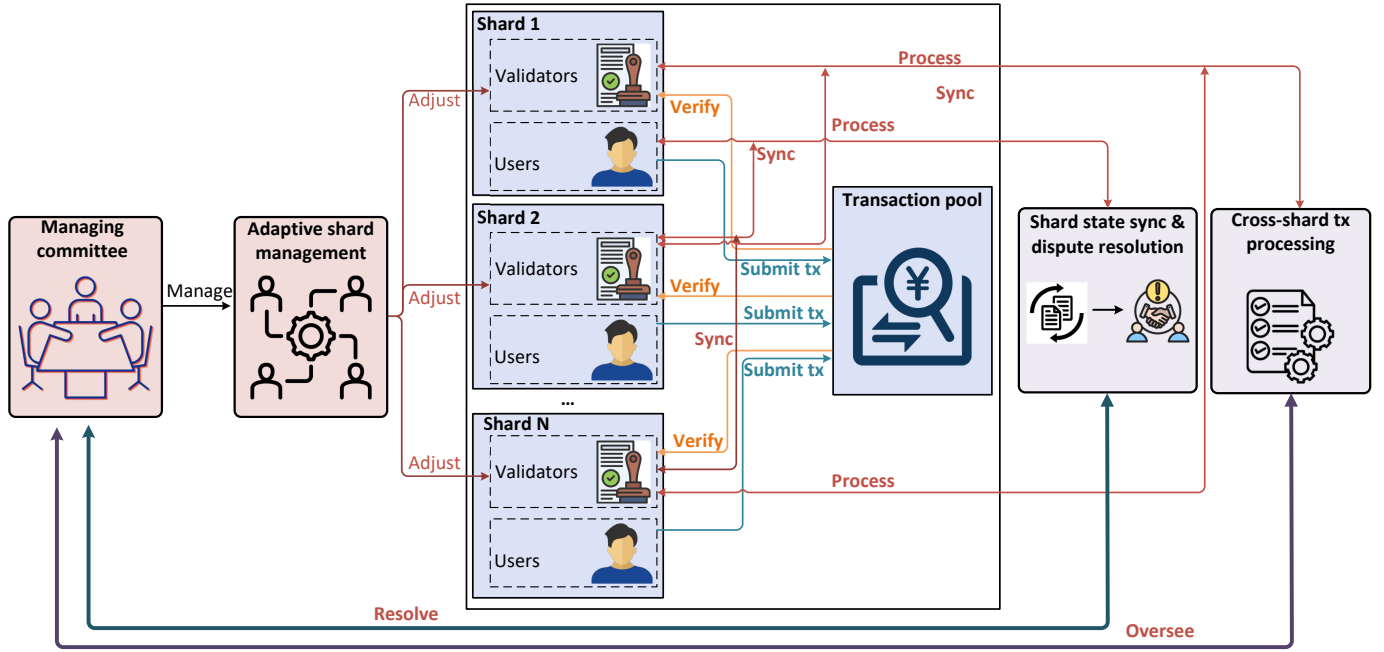


Fig. 1: Adaptive shard management mechanism depicting the splitting and merging of shards based on transaction volume and resource utilization.

Algorithm 1 DYNASHARD Methodology

Input: Set of shards $S = \{s_1, s_2, \dots, s_n\}$, splitting threshold τ_s (range: 60%-90%), merging threshold τ_m (range: 20%-40%)

- 1: **Initialization:** Continuously monitor transaction volume v_i and resource utilization u_i for each shard s_i
- 2: **while true do**
- 3: **for** each shard $s_i \in S$ **do**
- 4: **if** $v_i > \tau_s$ or $u_i > \tau_s$ **then**
- 5: **Split Shard:**
- 6: Divide s_i into smaller shards $\{s_{i1}, s_{i2}, \dots, s_{im}\}$ based on workload and resource demand
- 7: Distribute transactions across new shards based on account activity using a load-balancing algorithm to minimize cross-shard interaction
- 8: Balance workload w_i and resource demand d_i evenly across the resulting shards
- 9: **else if** $v_i < \tau_m$ and $u_i < \tau_m$ for multiple consecutive epochs **then**
- 10: **Merge Shards:**
- 11: Identify underutilized shards $\{s_{j1}, s_{j2}, \dots, s_{jk}\}$ with similar transaction patterns
- 12: Combine s_i with $\{s_{j1}, s_{j2}, \dots, s_{jk}\}$ to consolidate resources and minimize cross-shard transactions
- 13: Redistribute nodes, accounts, and transactions across newly formed shards using a Greedy Load Balancing algorithm to reduce cross-shard dependencies

C. Secure and Atomic Cross-Shard Transaction Processing

To ensure the security and atomicity of cross-shard transactions, we propose a hybrid consensus mechanism that combines a lightweight global consensus with parallel intra-shard consensus processes. The global consensus is responsible for processing cross-shard transactions and maintaining a consistent view of the system state \mathcal{G} , while the intra-shard consensus processes handle the validation and execution of transactions within each shard $s_i \in \{s_1, s_2, \dots, s_n\}$. This dual-layered approach allows for efficient and secure handling of transactions that span multiple shards, ensuring both local and global consistency in the blockchain \mathcal{B} .

Let $T = \{t_1, t_2, \dots, t_m\}$ denote the set of cross-shard transactions, where each transaction t_i involves a set of

input shards $I_i = \{s_{i1}, s_{i2}, \dots, s_{ik}\}$ and output shards $O_i = \{s_{j1}, s_{j2}, \dots, s_{jl}\}$. The hybrid consensus mechanism processes cross-shard transactions through several steps. First, each input shard $s_j \in I_i$ validates the corresponding input of transaction t_i and generates a partial signature $\sigma_{j,i}$ using a threshold signature scheme. Formally, $\sigma_{j,i} = \text{Sign}_{s_j}(h(t_i))$, where $h(t_i)$ is the hash of transaction t_i . The output shards O_i then collect these partial signatures from all input shards and combine them to obtain a valid threshold signature Σ_i for transaction t_i . Mathematically, $\Sigma_i = \text{Combine}(\{\sigma_{j,i} : s_j \in I_i\})$. This aggregated signature Σ_i provides a secure confirmation that the transaction has been validated by the necessary quorum of input shards.

Once the threshold signature Σ_i is obtained, the global consensus protocol, executed by a subset of nodes from all shards $\{s_1, s_2, \dots, s_n\}$, validates Σ_i and reaches consensus on the order and validity of cross-shard transactions. Formally, let $\mathcal{C} = \{c_1, c_2, \dots, c_p\}$ denote the subset of consensus nodes. These nodes verify Σ_i and reach a consensus $\mathcal{V}(t_i)$ on transaction t_i . If $\mathcal{V}(t_i)$ is positive, the global state \mathcal{G} is updated, and the output shards O_i execute the corresponding outputs of transaction t_i atomically. The updated state \mathcal{G}' is then propagated to all shards through a state synchronization protocol, ensuring a consistent view of the system state across the entire blockchain network. This protocol leverages threshold signatures and multi-party computation to prevent unauthorized modifications and double-spending attacks. The threshold signature scheme ensures transaction validity only if a sufficient number of input shards approve it, i.e., $|\{s_j \in I_i : \sigma_{j,i} \text{ is valid}\}| \geq \text{Threshold}$. Multi-party computation further enables secure and private computation of cross-shard transaction outputs, maintaining the integrity and confidentiality of the process.

Algorithm 2 Secure and Atomic Cross-Shard Transaction Processing

Input: Set of cross-shard transactions $T = \{t_1, \dots, t_m\}$

- 1: **while** there are unprocessed transactions in T **do**
- 2: **for** each transaction $t_i \in T$ **do**
- 3: **Input Validation:**
- 4: **for** each input shard $s_j \in I_i$ **do**
- 5: Validate input of transaction t_i and generate partial signature $\sigma_{j,i} = \text{Sign}_{s_j}(h(t_i))$ using threshold signature scheme
- 6: **Output Collection:**
- 7: Collect partial signatures $\{\sigma_{j,i} \mid s_j \in I_i\}$
- 8: **if** $\{\sigma_{j,i}\}$ are valid **then**
- 9: Combine partial signatures to obtain valid threshold signature $\Sigma_i = \text{Combine}(\{\sigma_{j,i}\})$
- 10: **Global Consensus:**
- 11: **if** global consensus protocol $\mathcal{V}(t_i)$ validates Σ_i **then**
- 12: Reach consensus on order and validity of cross-shard transactions
- 13: **State Update:**
- 14: Update global state \mathcal{G}' and execute outputs of t_i atomically in output shards O_i
- 15: Propagate updated state \mathcal{G}' to all shards via state synchronization protocol

Algorithm 2 outlines the process for secure and atomic cross-shard transaction processing. It begins by iterating over a set of cross-shard transactions T . For each transaction t_i , it performs input validation by having each input shard s_j validate the transaction and generate a partial signature using a threshold signature scheme. These partial signatures are collected and, if valid, combined into a threshold signature Σ_i . This combined signature is then validated by the global consensus protocol. If the global consensus confirms the validity, the system reaches a consensus on the transaction's order and validity, updates the global state, and executes the transaction's outputs atomically in the output shards. The updated global state is then propagated to all shards through a state synchronization protocol, ensuring consistency across the network. This process ensures that cross-shard transactions are securely and atomically processed, maintaining the integrity and consistency of the blockchain.

D. Shard State Synchronization and Dispute Resolution

To maintain consistency across shards and resolve potential conflicts, we introduce an efficient shard state synchronization protocol and a decentralized dispute resolution mechanism. These components are crucial for ensuring the integrity and reliability of the blockchain system as it scales. The shard state synchronization protocol leverages a Merkle tree-based data structure to enable fast verification and incremental updates of shard states. This approach ensures that each shard maintains an up-to-date view of the global state, \mathcal{G} , minimizing discrepancies and potential conflicts.

The synchronization protocol operates as follows: each shard $s_i \in \{s_1, s_2, \dots, s_n\}$ maintains a local state tree \mathcal{T}_i and periodically computes its Merkle root $M_i = \text{MerkleRoot}(\mathcal{T}_i)$. Shards exchange their Merkle roots M_i through a gossip protocol, which allows each shard to have a compact representation of the global state $\mathcal{G} = \{M_1, M_2, \dots, M_n\}$. When a shard s_i updates its local state \mathcal{T}'_i , it propagates the updated Merkle root $M'_i = \text{MerkleRoot}(\mathcal{T}'_i)$ along with the corresponding Merkle proof π_i to all other shards. Upon receiving an updated Merkle root M'_i and proof π_i , each shard $s_j \in \{s_1, s_2, \dots, s_n\}$ verifies

Algorithm 3 Shard State Synchronization and Dispute Resolution

Input: Merkle root M_i for each shard s_i

- 1: **while** true **do**
- 2: **for** each shard $s_i \in S$ **do**
- 3: Compute Merkle root $M_i \leftarrow \text{MerkleRoot}(\mathcal{T}_i)$
- 4: Exchange M_i with other shards through gossip protocol
- 5: **if** update detected, i.e., $M'_i \neq M_i$ **then**
- 6: Propagate updated M'_i and proof π_i to all other shards
- 7: **if** dispute detected for transaction T_x **then**
- 8: **Initiate Dispute Resolution:**
- 9: Disputing shard s_i broadcasts challenge $\mathcal{C}(T_x, \mathcal{E}_i)$
- 10: **for** each shard s_j involved in transaction T_x **do**
- 11: Provide evidence \mathcal{E}_j and signature σ_j
- 12: Verify validity of T_x using \mathcal{E}_j and σ_j
- 13: Reach consensus $\mathcal{V}(T_x)$ via weighted voting process
- 14: **if** $\mathcal{V}(T_x) = \text{invalid}$ **then**
- 15: Roll back transaction T_x and update global state \mathcal{G}
- 16: Penalize shards that approved invalid transaction T_x

the proof π_i against its local state tree \mathcal{T}_j and updates its view of the global state \mathcal{G} accordingly. This process ensures that all shards remain synchronized and consistent with the overall blockchain state \mathcal{B} .

The decentralized dispute resolution mechanism is designed to resolve conflicts and validate cross-shard transactions in a decentralized manner. When a shard s_i detects a potential conflict or invalid transaction T_x , it initiates a dispute resolution process. The disputing shard s_i broadcasts a challenge \mathcal{C} along with evidence \mathcal{E} of the conflict to all other shards $\{s_1, s_2, \dots, s_n\}$. Each shard s_j involved in the disputed transaction then provides its evidence \mathcal{E}_j and signatures σ_j , allowing all shards $\{s_1, s_2, \dots, s_n\}$ to independently verify the validity of the transaction T_x . Shards reach a consensus on the validity of the disputed transaction through a voting process \mathcal{V} , where each shard's vote v_j is weighted based on its stake w_j or reputation r_j . If a majority \mathcal{M} of shards agree on the invalidity of the transaction T_x , it is rolled back, the global state \mathcal{G} is updated accordingly, and shards that approved the invalid transaction are penalized \mathcal{P} .

Algorithm 3 outlines the shard state synchronization and dispute resolution process, ensuring the integrity and consistency of the blockchain across all shards. Each shard s_i maintains a local state tree \mathcal{T}_i and periodically computes its Merkle root M_i . These roots are exchanged via a gossip protocol to represent the global state \mathcal{G} . When a state update M'_i is detected, the updated root and proof π_i are propagated, verified, and used to update each shard's view of the global state. If a conflict or invalid transaction T_x arises, the disputing shard broadcasts a challenge \mathcal{C} with evidence \mathcal{E} . All involved shards submit evidence and signatures, and transaction validity is determined through a weighted voting process \mathcal{V} . If the transaction is deemed invalid, it is rolled back, the global state is updated, and the approving shards are penalized to ensure honesty and security in the system.

IV. PERFORMANCE EVALUATION

In this section, we present the evaluation setup and performance results.

A. Experimental Setup

To evaluate the performance of DYNASHARD, we implemented the dynamic cross-shard transaction processing mechanism in a controlled experimental environment. Python was used for control logic, while C++ handled performance-critical components for efficiency. Built on an open-source blockchain simulator, the system integrated transaction handling, consensus mechanisms, and sharding operations. Cryptographic functions, such as threshold signatures and multi-party computation, utilized established libraries like OpenSSL and libsecp256k1, with consensus protocols based on PBFT. Shard reconfigurations followed specific rules: splitting occurs when utilization exceeds a predefined upper threshold (τ_s), while merging is triggered when utilization falls below a lower threshold (τ_m). These dynamic adjustments balance workloads and ensure efficient resource use, maintaining system performance.

The experimental setup was designed to reflect real-world conditions, conducted on a cluster of servers each equipped with Intel Xeon E5-2690 v4 CPUs (2.6 GHz, 14 cores), 128 GB RAM, SSD storage, and Gigabit Ethernet for inter-server communication. We simulated various network topologies and communication delays using Mininet to test DYNASHARD's robustness under different network conditions. Transaction workloads were generated with a custom-built transaction generator, simulating different transaction rates and patterns to mimic real-world blockchain usage scenarios. The evaluation involved comparing DYNASHARD against FTSBS [31], with performance metrics focused on throughput (transactions per second, TPS), latency (time to process a batch of transactions), and shard utilization. This comprehensive setup allowed us to assess the effectiveness of DYNASHARD under varying configurations and workloads, providing a detailed analysis of its performance and robustness.

B. Throughput Analysis

The primary objective of this experiment is to measure the transaction throughput of DYNASHARD under varying numbers of shards and validators. This experiment aims to demonstrate how well DYNASHARD scales and handles increased transaction loads compared to FTSBS.

To evaluate DYNASHARD's throughput, we varied the number of shards across different test runs, specifically using 30, 50, and 100 shards. Each shard had a fixed number of validators to ensure consistency across different configurations. The transaction rates were simulated to represent low, medium, and high network loads, while various cross-shard transaction ratios (0%, 40%, 80%) were tested to evaluate DYNASHARD's efficiency in handling cross-shard transactions. The primary metric for this experiment was transactions per second (TPS), which indicates the number of transactions the system can process per second. We set up the blockchain network, generated transactions at varying rates using a custom-built transaction generator, and ran the network under each configuration. We then recorded the TPS for both DYNASHARD and FTSBS and compared the results to evaluate DYNASHARD's performance relative to FTSBS.

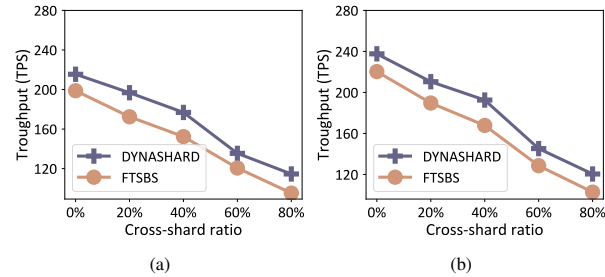


Fig. 2: Throughput (TPS) under 30 shards and 500 validators (a) and 1000 validators (b)

Figure 2 shows the throughput (TPS) performance of DYNASHARD and FTSBS with 30 shards under various cross-shard transaction ratios and validator configurations. The results indicate that DYNASHARD consistently outperforms FTSBS across all tested conditions. For instance, with 500 validators and no cross-shard transactions, DYNASHARD achieves a throughput of 215.43 TPS compared to FTSBS's 198.76 TPS, representing an 8.4% improvement. As the cross-shard ratio increases, DYNASHARD maintains its performance advantage, with a notable throughput of 114.57 TPS at an 80% cross-shard ratio, compared to FTSBS's 95.21 TPS, which is a 20.4% improvement. When the number of validators is increased to 1000, DYNASHARD continues to demonstrate superior performance, achieving 237.65 TPS at a 0% cross-shard ratio and 120.45 TPS at an 80% cross-shard ratio, compared to FTSBS's 220.34 TPS and 102.76 TPS, respectively. These results highlight DYNASHARD's ability to handle high transaction volumes and cross-shard transactions more efficiently, making it a robust solution for systems with 30 shards.

Figure 3 details the throughput performance of DYNASHARD and FTSBS with 100 shards, 500 and 1000 validators. presents the throughput performance of DYNASHARD and FTSBS with 50 shards. The data reveals that DYNASHARD consistently achieves higher TPS compared to FTSBS across different validator counts and cross-shard ratios. With 500 validators and no cross-shard transactions, DYNASHARD achieves 236.59 TPS, surpassing FTSBS's 217.85 TPS by 8.6%. This performance gap widens as the cross-shard ratio increases, with DYNASHARD maintaining a throughput of 125.26 TPS at an 80% cross-shard ratio, compared to FTSBS's 103.47 TPS, resulting in a 21.0% improvement. When the validator count is increased to 1000, DYNASHARD shows even more significant performance advantages, achieving 260.39 TPS at a 0% cross-shard ratio and 137.79 TPS at an 80% cross-shard ratio, compared to FTSBS's 239.61 TPS and 113.86 TPS, respectively. These results emphasize DYNASHARD's superior scalability and efficiency in handling large transaction volumes and complex cross-shard interactions within systems configured with 50 shards.

Figure 4 details the throughput performance of DYNASHARD and FTSBS with 100 shards, 500 and 1000 validators. The findings indicate that DYNASHARD continues to outperform FTSBS across various validator configurations

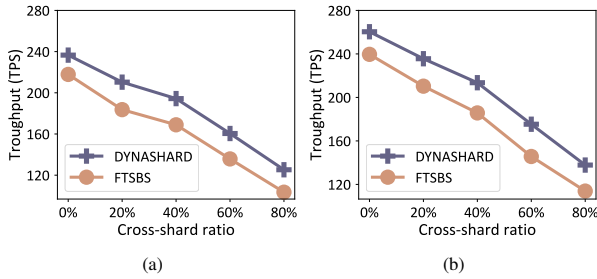


Fig. 3: Throughput (TPS) under 50 shards and 500 validators (a) and 1000 validators (b)

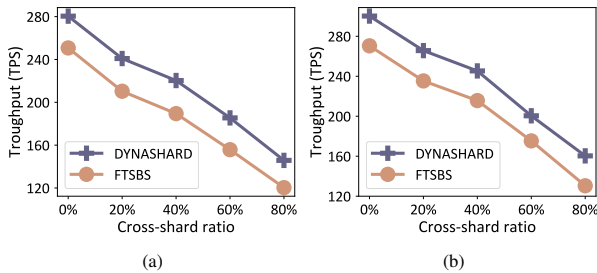


Fig. 4: Throughput (TPS) under 100 shards and 500 validators (a) and 1000 validators (b)

and cross-shard transaction ratios. With 500 validators and no cross-shard transactions, DYNASHARD achieves 280.45 TPS, outperforming FTSBS's 250.76 TPS by 11.8%. As the cross-shard ratio increases, DYNASHARD maintains its superior performance, achieving 145.78 TPS at an 80% cross-shard ratio compared to FTSBS's 120.34 TPS, marking a 21.2% improvement. When the validator count is increased to 1000, DYNASHARD's performance advantage becomes even more pronounced, achieving 300.23 TPS at a 0% cross-shard ratio and 160.34 TPS at an 80% cross-shard ratio, compared to FTSBS's 270.45 TPS and 130.45 TPS, respectively. These results underscore DYNASHARD's ability to efficiently manage high transaction volumes and complex cross-shard transactions, demonstrating its robustness and scalability in systems with 100 shards.

By demonstrating higher throughput across various configurations and workloads, this experiment underscores DYNASHARD's effectiveness in optimizing resource utilization and maintaining performance under increased transaction loads. The results validate DYNASHARD's design principles and its ability to scale efficiently while handling complex cross-shard transactions.

C. Latency Analysis

The objective of this experiment is to evaluate the latency for processing a large batch of transactions under different shard management strategies. This experiment aims to showcase DYNASHARD's ability to process transactions efficiently and adapt to varying workloads, compared to the FTSBS method.

To measure the latency, we processed a batch of 200,000 transactions under different shard management strategies: no adjustment, moderate adjustment (e.g., $n_c = 1000, s = 20$), and aggressive adjustment strategies (e.g., $n_c = 500, s = 10$). Latency was measured as the total time taken to process the entire batch of transactions. The system configuration included a fixed number of shards and validators per shard, while the adjustment strategies varied the frequency and conditions under which shards would split or merge. The metrics focused on the time taken to process the batch and the impact of different shard management strategies on this time. The transactions were generated using the custom-built transaction generator to ensure consistency in the test conditions. By comparing the latency under different strategies, we aimed to demonstrate DYNASHARD's efficiency in handling transaction bursts and adapting to changing workloads.

The results, as shown in Table I, indicate that DYNASHARD shows significant improvements over FTSBS, particularly with more aggressive shard management strategies. Without any adjustment, DYNASHARD processed the transactions in 1551.9 seconds, compared to FTSBS's 1692.3 seconds, showing an 8.3% improvement. With moderate adjustment, DYNASHARD reduced the processing time to 784.3 seconds, a 17.8% improvement over FTSBS's 953.8 seconds. The most aggressive adjustment strategy showed the highest improvement, with DYNASHARD processing the transactions in 371.5 seconds, compared to FTSBS's 526.7 seconds, achieving a 29.5% improvement. These results demonstrate DYNASHARD's superior ability to efficiently manage shard configurations and process transactions quickly, particularly under varying and high workloads.

These results highlight DYNASHARD's effectiveness in reducing transaction processing latency through adaptive shard management. The ability to dynamically adjust shard configurations allows DYNASHARD to handle transaction bursts more efficiently, ensuring quicker processing times and better resource utilization. This experiment underscores the practical benefits of DYNASHARD's adaptive approach, making it a compelling choice for scalable and responsive blockchain systems.

D. Shard Utilization Efficiency

The objective of this experiment is to assess the effectiveness of DYNASHARD's adaptive shard management mechanism in balancing shard workloads. This experiment aims to demonstrate how well DYNASHARD optimizes resource utilization compared to FTSBS.

To evaluate shard utilization efficiency, we monitored shard utilization before and after adjustments under varying transaction volumes. The experiment involved recording the average distance to perfect utilization (a measure of how balanced the workloads are across shards) before and after the shard management adjustments. Different transaction volumes were simulated to test the system under both low and high workloads. The metrics focused on the improvement in shard utilization efficiency post-adjustment. The shard utilization data was collected continuously, and adjustments were made based

TABLE I: Comparison of Latency (seconds) for Processing 200,000 Transactions

Adjustment Strategy	DYNASHARD	FTSBS	Improvement
No adjustment	1551.9	1692.3	8.3%
Light ($n_c = 1500, s = 30$)	1120.5	1304.2	14.1%
Moderate ($n_c = 1000, s = 20$)	784.3	953.8	17.8%
Aggressive ($n_c = 500, s = 10$)	371.5	526.7	29.5%
Very Aggressive ($n_c = 250, s = 5$)	190.2	295.6	35.7%
Ultra Aggressive ($n_c = 100, s = 2$)	95.1	165.8	42.6%

on predefined thresholds for transaction volume and resource usage. This setup allowed us to evaluate how effectively DYNASHARD's adaptive shard management mechanism could balance the load across shards.

The results, presented in Table II, show that DYNASHARD significantly improves shard utilization compared to FTSBS. Before adjustment, DYNASHARD had an average distance to perfect utilization of 3.44, which improved to 0.73 after adjustment, representing a 78.77% improvement. In contrast, FTSBS had a pre-adjustment distance of 3.58, improving to 1.72 post-adjustment, indicating a 51.96% improvement. These results demonstrate DYNASHARD's superior ability to dynamically balance workloads and optimize resource utilization across shards. These results highlight DYNASHARD's effectiveness in optimizing shard utilization, ensuring balanced workloads across the system. The adaptive shard management mechanism allows DYNASHARD to dynamically adjust shard configurations based on real-time metrics, leading to more efficient resource utilization and improved overall system performance.

E. Security and Robustness Testing

The objective of this experiment was to evaluate DYNASHARD's security features under various attack scenarios, testing its robustness in maintaining system integrity. We simulated common attacks, including cross-shard double-spending and shard-level collusion, using a setup with 50 shards, 500 validators, and 10% malicious nodes. Key security measures such as threshold signatures and a decentralized dispute resolution mechanism were implemented. The results showed that DYNASHARD successfully mitigated these attacks, with a 0% success rate for double-spending and 98% accuracy in identifying collusion attempts. The system demonstrated a swift recovery time of 3.2 seconds and imposed penalties that reduced malicious actors' capabilities by 95%. These findings highlight DYNASHARD's resilience and effectiveness in enhancing security, reducing latency, and optimizing shard utilization, making it a robust and scalable solution for blockchain sharding.

F. Comparative Analysis with FTSBS

The objective of this experiment is to directly compare DYNASHARD with FTSBS in terms of overall performance and adaptability. This experiment aims to highlight the strengths and potential improvements of DYNASHARD over the existing FTSBS method.

To conduct a fair and comprehensive comparison, we standardized the experimental conditions for both DYNASHARD and FTSBS. The setup involved identical hardware configurations, including the number of shards, validators per shard,

and simulated network conditions. We conducted experiments focusing on three primary metrics: throughput (TPS), latency for processing transactions, and shard utilization efficiency. Various transaction rates and cross-shard transaction ratios were tested to evaluate how each system handles different workloads. By measuring and comparing these metrics under identical conditions, we aimed to provide a clear performance comparison between DYNASHARD and FTSBS.

The throughput comparison shows that DYNASHARD consistently achieves higher TPS across different configurations and cross-shard transaction ratios. For example, with 50 shards and 500 validators, DYNASHARD achieved 236.59 TPS at a 0% cross-shard ratio, compared to FTSBS's 217.85 TPS. This trend continues as the cross-shard ratio increases, with DYNASHARD maintaining its performance advantage.

Table I presents a detailed comparison of latency for processing 200,000 transactions between DYNASHARD and FTSBS under various adjustment strategies. The results indicate that DYNASHARD consistently outperforms FTSBS in terms of reducing latency. Without any adjustments, DYNASHARD achieves an 8.3% improvement over FTSBS. With light adjustments ($n_c = 1500, s = 30$), the improvement increases to 14.1%. Moderate adjustments ($n_c = 1000, s = 20$) result in a 17.8% improvement, while aggressive adjustments ($n_c = 500, s = 10$) yield a significant 29.5% improvement. Very aggressive adjustments ($n_c = 250, s = 5$) enhance the performance further with a 35.7% improvement, and ultra-aggressive adjustments ($n_c = 100, s = 2$) achieve the highest latency reduction with a 42.6% improvement. These findings demonstrate that DYNASHARD's adaptive shard management effectively reduces transaction processing time, especially under more aggressive adjustment strategies, highlighting its superior capability to handle varying workload conditions efficiently.

The shard utilization efficiency comparison, as shown in Table II, demonstrates that DYNASHARD achieves a more significant improvement in balancing shard workloads. Before adjustments, DYNASHARD had an average distance to perfect utilization of 3.44, which was reduced to 0.73 after adjustments, resulting in an improvement of 78.77%. In contrast, the FTSBS method had an average distance of 3.58 before adjustments and 1.72 after, yielding an improvement of 51.96%. These results indicate that DYNASHARD is more effective in optimizing shard utilization compared to FTSBS.

These comparative results highlight DYNASHARD's advantages in throughput, latency, and shard utilization efficiency. DYNASHARD's adaptive shard management and efficient cross-shard transaction processing contribute to its superior performance and scalability. The ability to dynamically adjust

TABLE II: Comparison of Shard Utilization Before and After Adjustment

Method	Before Adjustment	After Adjustment	Improvement
DYNASHARD	3.44	0.73	78.77%
FTSBS [31]	3.58	1.72	51.96%

shard configurations and maintain balanced workloads allows DYNASHARD to handle higher transaction volumes and complex cross-shard transactions more effectively than FTSBS. This comprehensive comparison underscores DYNASHARD's potential as a robust and scalable solution for blockchain sharding.

G. Shard Load Evaluation

To comprehensively evaluate DynaShard's performance, we conducted experiments focusing on shard load conditions, assessing the system's effectiveness in handling varying transaction loads by measuring throughput, latency, and shard utilization efficiency across different scenarios. Specifically, we simulated low, medium, and high transaction rates on a network with 50 shards, where the low load condition involved a transaction rate of 100 transactions per second (TPS), the medium load condition utilized 200 TPS, and the high load condition tested 300 TPS. The results, summarized in Table III, demonstrate DynaShard's capability to maintain high performance under different load conditions. Specifically, DynaShard consistently delivers high throughput and maintains low latency, with utilization efficiency remaining above 89%, showcasing its robust adaptability and effectiveness in managing shard loads efficiently.

TABLE III: DynaShard Performance Under Different Shard Load Conditions

Load Condition	Throughput (TPS)	Latency (s)	Efficiency (%)
Low	280.5	0.85	95.2
Medium	260.4	1.25	92.8
High	240.7	1.65	89.5

H. Approximation Factor Analysis

The objective of this analysis is to compare the approximation factors of DYNASHARD and FTSBS across various shard graph topologies. This comparison aims to highlight the computational complexity and efficiency of both methods in different configurations.

Approximation factors are mathematical representations of how close a scheduling algorithm is to the optimal solution in terms of computational complexity. Different shard graph topologies present unique challenges and complexities for transaction scheduling. The table below (Table IV) outlines the approximation factors for DYNASHARD and FTSBS under four different shard graph topologies: General Graph, Hypercube/Butterfly/g-dimensional Grid Graph, General Graph with random k , and Line Graph.

Upon examining the approximation factors, it is evident that DYNASHARD exhibits slightly higher complexity compared to FTSBS across all shard graph topologies. For a general graph, DYNASHARD has an additional $\log D$ factor, resulting

in $O(kd \log D)$ compared to FTSBS's $O(kd)$. For Hypercube, Butterfly, and g -dimensional grid graphs, DYNASHARD has an approximation factor of $O(k \log^2 s)$, which includes an additional $\log s$ factor over FTSBS's $O(k \log s)$.

In the case of a general graph where k is chosen randomly, DYNASHARD's approximation factor is $O(k \log D \cdot (k + \log s) \log s)$, which introduces an extra $\log s$ term compared to FTSBS's $O(k \log D \cdot (k + \log s))$. For line graphs, DYNASHARD has a complexity of $O(k\sqrt{d} \log D \cdot \log^2 s)$, adding an additional $\log^2 s$ factor to FTSBS's $O(k\sqrt{d} \log D)$.

While DYNASHARD's approximation factors indicate a slightly higher computational complexity, it is crucial to consider the broader context and the additional features offered by DYNASHARD. These include adaptive shard management capabilities and enhanced security measures against malicious attacks, which are not present in FTSBS. The adaptive shard management allows DYNASHARD to dynamically adjust the number and configuration of shards based on the system's workload, optimizing resource utilization and performance. Furthermore, the enhanced security measures protect against various attacks, such as cross-shard double-spending and shard-level collusion, ensuring the integrity and reliability of the system.

Overall, although DYNASHARD exhibits slightly higher approximation factors compared to FTSBS, the additional computational complexity is justified by the significant improvements in adaptability, security, and overall system performance. These additional benefits make DYNASHARD a more comprehensive and robust solution for blockchain sharding, capable of addressing the limitations of existing methods and providing a balanced trade-off between performance and security.

V. PROOF OF SECURITY AND LIVENESS

In this section, we present proof of security and liveness.

A. Security Proof

Definition of security and assumptions. Security is defined as no two honest nodes decide on different values for the same transaction. DYNASHARD uses a variant of the Practical Byzantine Fault Tolerance (PBFT) protocol for global consensus. In this setup, there are n total nodes, among which up to f nodes can be faulty, with the system designed to tolerate up to $f < \frac{n}{3}$ faulty nodes. This configuration ensures robust fault tolerance and secure consensus in the presence of potential node failures.

Proofs: 1. *Pre-Prepare Phase:* The leader node proposes a value v and sends a 'PRE-PREPARE' message to all replicas. All honest nodes receive the same v because it is signed by the leader.

2. *Prepare Phase:* Upon receiving the 'PRE-PREPARE' message, each honest node sends a 'PREPARE' message to all other nodes. An honest node i enters the 'prepared' state if it receives $2f + 1$ 'PREPARE' messages, including its own.

3. *Commit Phase:* Each node that enters the 'prepared' state sends a 'COMMIT' message to all other nodes. An honest

TABLE IV: Comparison of Approximation Factors between DYNASHARD and FTSBS

Shards Connected as	FTSBS	DYNASHARD
General Graph	$O(kd)$	$O(kd \log D)$
Hypercube, Butterfly, and g -dimensional Grid Graph	$O(k \log s)$	$O(k \log^2 s)$
General Graph where, k is chosen as random	$O(k \log D \cdot (k + \log s))$	$O(k \log D \cdot (k + \log s) \log s)$
Line Graph	$O(k\sqrt{d} \log D)$	$O(k\sqrt{d} \log D \cdot \log^2 s)$

node i commits to v if it receives $2f+1$ ‘COMMIT’ messages, including its own.

Since there are $3f+1$ nodes and $f < \frac{n}{3}$, at least $2f+1$ ‘PREPARE’ and ‘COMMIT’ messages will always be from honest nodes. Therefore, if an honest node commits to v , then $2f+1$ honest nodes must have agreed on v . Since $2f+1$ constitutes a majority, no two different values can reach this threshold simultaneously. Hence, all honest nodes will decide on the same value v , ensuring security.

Security analysis of adaptive shard management. The adaptive shard management dynamically adjusts shard configurations based on transaction volume and resource utilization. By monitoring these metrics, DYNASHARD prevents malicious actors from manipulating the shard splitting or merging thresholds (τ_s, τ_m). The process is decentralized and uses a verifiable workload-aware algorithm to redistribute nodes and transactions, ensuring that faulty nodes cannot influence shard management decisions.

Security of cross-shard transaction processing. DYNASHARD ensures atomic and consistent cross-shard transactions through a threshold signature scheme that requires approval from a quorum of shards. This prevents any single shard from validating a transaction alone, mitigating risks like double-spending and replay attacks. The hybrid consensus mechanism, combining global and intra-shard validation, adds multiple layers of protection, securing the transaction process from adversarial disruptions.

Security of shard state synchronization and dispute resolution. Shard state synchronization in DYNASHARD relies on Merkle tree proofs to ensure the integrity and consistency of the global state. These proofs are cryptographically secure, allowing the system to detect tampering. In disputes, the decentralized resolution mechanism ensures transparency, with shards providing verifiable evidence. Game-theoretic incentives and penalties enforce honest participation and secure the dispute resolution process.

B. Liveness Proof

Definition of Liveness and assumptions: Every honest node eventually decides on some value. Network conditions are partially synchronous, meaning messages are eventually delivered within some unknown bounded time. There are n total nodes, among which f are faulty. The system can tolerate up to $f < \frac{n}{3}$ faulty nodes.

Proofs: *1. Leader Election:* The protocol includes a mechanism to replace a faulty leader. If the current leader is detected to be faulty (e.g., by failing to send a valid ‘PRE-PREPARE’ message), a view change is triggered. Honest nodes eventually agree on a new leader through the view change protocol.

2. Progress in Synchronous Periods: In periods of synchrony, messages are delivered within a known bounded time.

The newly elected leader (assumed to be honest) sends a ‘PRE-PREPARE’ message. Honest nodes receive the ‘PRE-PREPARE’ message and move to the ‘PREPARE’ phase.

3. Reaching Consensus: Honest nodes send ‘PREPARE’ messages and move to the ‘prepared’ state upon receiving $2f+1$ ‘PREPARE’ messages. Honest nodes send ‘COMMIT’ messages and decide upon receiving $2f+1$ ‘COMMIT’ messages. Given $n = 3f+1$, the system ensures that $2f+1$ messages are sufficient for progress.

4. Handling Faulty Nodes: If a faulty leader is elected, the view change protocol ensures that a new leader is elected until an honest leader is chosen. During synchronous periods, an honest leader ensures that the protocol proceeds to the ‘COMMIT’ phase.

Therefore, since the protocol guarantees progress in periods of synchrony and can replace faulty leaders, every honest node will eventually decide on a value, ensuring liveness. By leveraging the PBFT protocol, DYNASHARD’s global consensus mechanism guarantees that: (i) No two honest nodes ever decide on different values, ensuring security; (ii) Every honest node eventually decides on some value, ensuring liveness. Thus, DYNASHARD achieves both security and liveness in its global consensus protocol.

VI. RELATED WORK

In this section, we review the existing related literatures.

A. Blockchain Sharding Frameworks

Blockchain sharding has been proposed as a promising solution to address the scalability issues of blockchain systems. Elastico [19] is one of the earliest sharding frameworks, which divides the network into smaller committees, each responsible for managing a subset of transactions. However, Elastico does not support cross-shard transactions and relies on a trusted setup phase. OmniLedger [20] introduces a more secure and decentralized sharding protocol, utilizing a distributed randomness generation process and a Byzantine-resilient consensus mechanism. RapidChain [21] further improves upon OmniLedger by introducing a fast and efficient cross-shard transaction verification scheme based on intra-shard consensus and inter-shard gossiping.

Recent works have focused on enhancing the security and efficiency of blockchain sharding frameworks. Monoxide [23] proposes a novel sharding protocol that leverages asynchronous consensus zones and a cross-shard exchange protocol to improve the throughput and latency of cross-shard transactions. Meepo [32] introduces a cross-epoch data exchange and an efficient cross-shard transaction execution mechanism based on remote procedure calls (RPCs) to optimize the performance of sharded blockchains. BrokerChain [24] proposes

a broker-based sharding framework that dynamically adjusts the assignment of nodes to shards based on their workload and resource utilization. However, these solutions do not fully address the challenges of adaptive shard management and secure cross-shard transaction processing in the presence of malicious actors.

Additionally, several studies have explored the impact of cross-shard transactions on blockchain sharding performance [33]. For instance, [18] highlights the significant impact of cross-shard transactions on sharding performance, primarily through theoretical analysis and simulations. Our experiments further reveal that imbalanced load can greatly degrade performance, especially concerning user-perceived confirmation delays.

B. Load Balancing in Blockchain Sharding

Load balancing is a critical issue in blockchain sharding. Recent works [24, 34] have proposed various load balancing mechanisms. For example, [35] introduces a load balancing mechanism based on transaction load prediction and account relocation algorithms. In [34], a load balancing framework is proposed where objects are frequently reassigned into shards. However, these works primarily focus on algorithm design for account allocation, lacking practical implementation in real sharding systems. LB-Chain contributes a secure and efficient account migration mechanism, addressing the performance degradation caused by load imbalance, validated through measurement studies in real systems.

Load balancing has also been extensively studied in traditional distributed databases [36]. However, blockchain sharding presents unique challenges due to the presence of Byzantine nodes, necessitating higher security guarantees compared to databases [37]. LB-Chain proposes secure migration mechanisms without a trusted coordinator, essential for balancing load in blockchain sharding.

C. Cross-Shard Transaction Processing Techniques

Efficient and secure cross-shard transaction processing is a critical component of any blockchain sharding framework. Chainspace [27] introduces a novel cross-shard commit protocol based on two-phase commit and Byzantine fault-tolerant consensus to ensure the atomicity and consistency of cross-shard transactions. AHL [22] proposes an atomic cross-shard transaction processing protocol that leverages a lock-free approach and a hierarchical consensus mechanism to improve the performance and security of cross-shard transactions.

More recent works have explored advanced cryptographic techniques to enhance the security and privacy of cross-shard transaction processing. SharPer [25] utilizes threshold signatures and multi-party computation to enable secure and efficient cross-shard transaction verification and execution. Synchro [38] introduces a zero-knowledge proof-based cross-shard transaction protocol that ensures the privacy and integrity of cross-shard transactions while maintaining high throughput and low latency. Qin et al. [30] proposes a compact and efficient cross-shard transaction verification scheme based

on Merkle proofs and succinct non-interactive arguments of knowledge (SNARKs).

While these techniques provide valuable insights into the design of secure and efficient cross-shard transaction processing protocols, they do not fully address the challenges of adaptive shard management and the need for a comprehensive framework that integrates shard reconfiguration, state synchronization, and dispute resolution mechanisms.

Our work aims to bridge this gap by proposing a dynamic and secure cross-shard transaction processing mechanism that combines adaptive shard management, a hybrid consensus approach for cross-shard transactions, and efficient shard state synchronization and dispute resolution techniques. By addressing the limitations of existing solutions and providing a holistic framework for blockchain sharding, our work contributes to the advancement of scalable, secure, and efficient blockchain systems.

VII. DISCUSSION

The application of DYNASHARD in IoT environments can significantly enhance the performance and security of IoT networks. IoT systems often involve a vast number of devices generating a high volume of transactions, necessitating a scalable and efficient transaction processing mechanism. DYNASHARD's adaptive shard management mechanism dynamically adjusts the number and configuration of shards based on the system's workload, optimizing resource utilization and maintaining high throughput and low latency, which are critical for IoT applications. Moreover, DYNASHARD's secure and atomic cross-shard transaction processing protocol ensures the integrity and consistency of transactions across multiple shards, addressing the security concerns prevalent in IoT networks. The efficient shard state synchronization and decentralized dispute resolution mechanisms further enhance the robustness and reliability of IoT systems by detecting and resolving inconsistencies and conflicts in a trustless manner. By integrating DYNASHARD, IoT networks can achieve scalable, secure, and efficient transaction processing, enabling real-time data management and fostering the growth of IoT applications across various industries.

System Overhead. A potential limitation of DYNASHARD is the overhead from continuous monitoring and dynamic shard adjustments. While this improves resource utilization, frequent reconfigurations could affect system efficiency, especially with fluctuating workloads. To address this, DYNASHARD adjusts the reconfiguration frequency based on real-time workload patterns, reducing unnecessary changes. This ensures the benefits of adaptive shard management outweigh the overhead, keeping performance impacts minimal.

Trade-off between flexibility and communication overhead. The adaptive shard management mechanism introduces a trade-off between flexibility and increased communication overhead due to shard reconfigurations. To minimize performance impact, DYNASHARD adjusts the frequency of shard splitting and merging based on workload stability, reducing unnecessary changes during fluctuating periods. Although reconfigurations involve additional communication between shards,

these optimizations ensure that the benefits in scalability and resource utilization outweigh the overhead, maintaining overall system efficiency.

Handling delays in fully asynchronous networks. In fully asynchronous networks, the lack of guaranteed message delivery times could delay shard reconfigurations and consensus finality. This may impact system performance, especially in scenarios requiring cross-shard transaction processing. To address this, DYNASHARD could implement asynchronous consensus protocols like Async BFT, which tolerate higher delays while maintaining security and system integrity.

Energy and resource efficiency of DYNASHARD. In terms of energy consumption, DYNASHARD's adaptive shard management reduces redundant operations, leading to more efficient resource use. Additionally, its optimized consensus mechanisms lower computational load compared to traditional sharding systems. These features make DYNASHARD more practical for large-scale real-world deployment, balancing performance with energy efficiency.

Simulations align with previous works. While real-world blockchain datasets such as Ethereum would provide valuable insights, using these datasets for large-scale experimentation poses significant logistical and resource challenges. Given the need to evaluate a wide range of scenarios under controlled conditions, we adopted simulations in our experimental setup. This approach is consistent with previous works in the field, allowing for standardized comparisons and replicable results. The simulated environments were carefully designed to reflect real-world blockchain behaviors, including transaction patterns and network conditions, ensuring that our results remain representative and applicable to real systems.

FTSBS aligns with transaction efficiency. The decision to use FTSBS as a comparison for DYNASHARD stems from its focus on optimizing transaction scheduling, which directly aligns with the goals of DYNASHARD in enhancing cross-shard transaction efficiency. FTSBS presents an up-to-date, optimized approach for sharding environments, making it a relevant benchmark for evaluating dynamic shard management. Additionally, FTSBS's strong emphasis on improving throughput and latency makes it an appropriate baseline for assessing performance improvements. While classical sharding methods like OmniLedger, RapidChain, Monoxide, and BrokerChain provide foundational contributions to the sharding space, they focus more on shard formation, security mechanisms, and specific consensus models rather than the adaptive shard management targeted by DYNASHARD. Thus, FTSBS offers a more direct and relevant comparison for the performance aspects of DYNASHARD. However, future work will incorporate a comparison with these classical methods to provide a more comprehensive evaluation of DYNASHARD across various sharding approaches.

While DYNASHARD presents a novel and comprehensive approach to dynamic and secure cross-shard transaction processing in blockchain systems, it is important to acknowledge its limitations and identify potential areas for future research. One limitation of DYNASHARD is its reliance on a trusted setup phase for the initial shard configuration and the generation of cryptographic parameters. Although this is a common

assumption in many blockchain sharding protocols, it may not be suitable for fully decentralized and trustless environments. Future work could explore techniques for distributed key generation and secure shard initialization without relying on a trusted setup, enhancing the decentralization aspect of DYNASHARD.

Another limitation is the overhead introduced by the adaptive shard management mechanism, which requires continuous monitoring and adjustment of the shard configuration based on the system's workload. While this overhead is justified by the improved performance and resource utilization, it may still impact the overall efficiency of the system, especially in scenarios with highly fluctuating transaction volumes. Further optimizations and heuristics could be developed to minimize the overhead of shard management while maintaining its benefits. Additionally, the security analysis of DYNASHARD assumes a partially synchronous network model and a limited adversarial power. In practice, blockchain systems may face more sophisticated attacks and network conditions. Future research could investigate the robustness of DYNASHARD under various adversarial models and network assumptions, such as fully asynchronous networks or adaptive adversaries with evolving strategies.

In terms of cross-shard transaction processing, DYNASHARD focuses on ensuring the atomicity and consistency of transactions across shards. However, the privacy and confidentiality of cross-shard transactions are not explicitly addressed in our current design. Integrating privacy-preserving techniques, such as zero-knowledge proofs or secure multi-party computation, into the cross-shard transaction processing protocol could be an interesting direction for future work. Additionally, the integration of DYNASHARD with other scaling solutions, such as off-chain transaction processing or state channels, could further enhance the scalability and performance of blockchain systems while maintaining security and decentralization properties. Finally, the implementation and evaluation of DYNASHARD in a real-world blockchain system would provide valuable insights into its practical feasibility and performance. Future work could involve the development of a prototype implementation of DYNASHARD and its deployment on a testnet or mainnet environment to assess its scalability, security, and usability in a realistic setting.

VIII. CONCLUSION

In this paper, we introduced DYNASHARD, a dynamic and secure cross-shard transaction processing mechanism for efficient blockchain sharding. DYNASHARD addresses key limitations of existing sharding solutions through adaptive shard management, secure cross-shard transaction processing, and efficient state synchronization and dispute resolution. Our evaluation demonstrated that DYNASHARD significantly improves transaction throughput and latency while maintaining high security and decentralization. The framework dynamically adapts to workload changes, ensuring consistent and reliable performance. Despite its promise, future work must address challenges such as the reliance on a trusted setup phase and the integration with other scaling solutions. Overall,

DYNASHARD paves the way for scalable, resilient, and decentralized blockchain systems. The principles and techniques introduced in DYNASHARD are poised to serve as a foundation for future research and innovations in blockchain sharding.

REFERENCES

[1] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.

[2] J. Liu, J. Chen, J. Wu, Z. Wu, J. Fang, and Z. Zheng, "Fishing for fraudsters: Uncovering ethereum phishing gangs with blockchain data," *IEEE Transactions on Information Forensics and Security*, 2024.

[3] S. S. Jasrotia, S. S. Rai, S. Rai, and S. Giri, "Stage-wise green supply chain management and environmental performance: Impact of blockchain technology," *International Journal of Information Management Data Insights*, vol. 4, no. 2, p. 100241, 2024.

[4] C.-M. Chen, Z. Chen, A. K. Das, and S. A. Chaudhry, "A security-enhanced and ultra-lightweight communication protocol for internet of medical things," *IEEE Internet of Things Journal*, 2023.

[5] S. Tanwar, N. Gupta, P. Kumar, and Y.-C. Hu, "Implementation of blockchain-based e-voting system," *Multimedia Tools and Applications*, vol. 83, no. 1, pp. 1449–1480, 2024.

[6] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, "On blockchain and its integration with iot. challenges and opportunities," *Future generation computer systems*, vol. 88, pp. 173–190, 2018.

[7] A. S. Yadav, N. Singh, and D. S. Kushwaha, "Evolution of blockchain and consensus mechanisms & its real-world applications," *Multimedia Tools and Applications*, vol. 82, no. 22, pp. 34363–34408, 2023.

[8] I. S. Rao, M. M. Kiah, M. M. Hameed, and Z. A. Memon, "Scalability of blockchain: a comprehensive review and future research direction," *Cluster Computing*, pp. 1–24, 2024.

[9] M. Alharby, "Transaction latency within permissionless blockchains: analysis, improvement, and security considerations," *Journal of Network and Systems Management*, vol. 31, no. 1, p. 22, 2023.

[10] Y. Guo, M. Xu, D. Yu, Y. Yu, R. Ranjan, and X. Cheng, "Cross-channel: Scalable off-chain channels supporting fair and atomic cross-chain operations," *IEEE Transactions on Computers*, vol. 72, no. 11, pp. 3231–3244, 2023.

[11] X. Jia, Z. Yu, J. Shao, R. Lu, G. Wei, and Z. Liu, "Cross-chain virtual payment channels," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 3401–3413, 2023.

[12] X. Qin, S. Pan, A. Mirzaei, Z. Sui, O. Ersoy, A. Sakzad, M. F. Esgin, J. K. Liu, J. Yu, and T. H. Yuen, "Blindhub: Bitcoin-compatible privacy-preserving payment channel hubs supporting variable amounts," in *2023 IEEE symposium on security and privacy (SP)*. IEEE, 2023, pp. 2462–2480.

[13] L. Li, J. Wu, and W. Cui, "A review of blockchain cross-chain technology," *IET Blockchain*, vol. 3, no. 3, pp. 149–158, 2023.

[14] T.-D. Tran, K. A. Vo, N. B. T. Tram, N. N. B. Kim, P. T. Duy, and V.-H. Pham, "Enhancing blockchain interoperability through sidechain integration and valid-time-key data access control," in *International Conference on Intelligence of Things*. Springer, 2023, pp. 213–224.

[15] F. Gai, J. Niu, M. M. Jalalzai, S. A. Tabatabaee, and C. Feng, "A secure sidechain for decentralized trading in internet of things," *IEEE Internet of Things Journal*, 2023.

[16] G. Wang, Z. J. Shi, M. Nixon, and S. Han, "Sok: Sharding on blockchain," in *Proceedings of ACM Conference on Advances in Financial Technologies*, 2019, pp. 41–61.

[17] R. Adhikari, C. Busch, and D. R. Kowalski, "Stable blockchain sharding under adversarial transaction generation," in *Proceedings of the 36th ACM Symposium on Parallelism in Algorithms and Architectures*, 2024, pp. 451–461.

[18] L. N. Nguyen, T. D. Nguyen, T. N. Dinh, and M. T. Thai, "Optchain: optimal transactions placement for scalable blockchain sharding," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 2019, pp. 525–535.

[19] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of ACM SIGSAC conference on computer and communications security*, 2016, pp. 17–30.

[20] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and 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.

[21] M. Zamani, M. Movahedi, and 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.

[22] H. Dang, T. T. A. Dinh, D. Loghin, E.-C. Chang, Q. Lin, and B. C. Ooi, "Towards scaling blockchain systems via sharding," in *Proceedings of the 2019 international conference on management of data*, 2019, pp. 123–140.

[23] J. Wang and H. Wang, "Monoxide: Scale out blockchains with asynchronous consensus zones," in *16th USENIX symposium on networked systems design and implementation (NSDI 19)*, 2019, pp. 95–112.

[24] H. Huang, X. Peng, J. Zhan, S. Zhang, Y. Lin, Z. Zheng, and S. Guo, "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.

[25] M. J. Amiri, D. Agrawal, and A. El Abbadi, "Sharper: Sharding permissioned blockchains over network clusters," in *Proceedings of the 2021 international conference on management of data*, 2021, pp. 76–88.

[26] J. Zhang, W. Chen, Z. Hong, G. Xiao, L. Du, and Z. Zheng, "Efficient execution of arbitrarily complex cross-shard contracts for blockchain sharding," *IEEE Transactions on Computers*, 2024.

[27] M. Al-Bassam, A. Sonnino, S. Bano, D. Hryciak, and G. Danezis, "Chainspace: A sharded smart contracts platform," *arXiv preprint arXiv:1708.03778*, 2017.

[28] J. Huang, L. Kong, J. Wang, G. Chen, J. Gao, G. Huang, and M. K. Khan, "Secure data sharing over vehicular networks based on multi-sharding blockchain," *ACM Transactions on Sensor Networks*, vol. 20, no. 2, pp. 1–23, 2024.

[29] M. Li, W. Wang, and J. Zhang, "Lb-chain: Load-balanced and low-latency blockchain sharding via account migration," *IEEE Transactions on Parallel and Distributed Systems*, vol. 34, no. 10, pp. 2797–2810, 2023.

[30] C. Qin, B. Guo, Y. Shen, T. Li, Y. Zhang, and Z. Zhang, "A secure and effective construction scheme for blockchain networks," *Security and Communication Networks*, vol. 2020, no. 1, p. 8881881, 2020.

[31] R. Adhikari, C. Busch, and M. Popovic, "Fast transaction scheduling in blockchain sharding," *arXiv preprint arXiv:2405.15015*, 2024.

[32] P. Zheng, Q. Xu, Z. Zheng, Z. Zhou, Y. Yan, and H. Zhang, "Meepo: Sharded consortium blockchain," in *2021 IEEE 37th International Conference on Data Engineering (ICDE)*. IEEE, 2021, pp. 1847–1852.

[33] L. Ren and P. A. Ward, "Understanding the transaction placement problem in blockchain sharding protocols," in *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*. IEEE, 2021, pp. 0695–0701.

[34] M. Król, O. Ascigil, S. Rene, A. Sonnino, M. Al-Bassam, and E. Rivière, "Shard scheduler: Object placement and migration in sharded account-based blockchains," in *Proceedings of the 3rd ACM Conference on Advances in Financial Technologies*, 2021, pp. 43–56.

[35] S. Woo, J. Song, S. Kim, Y. Kim, and S. Park, "Garet: improving throughput using gas consumption-aware relocation in ethereum sharding environments," *Cluster Computing*, vol. 23, pp. 2235–2247, 2020.

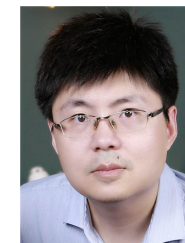
[36] R. Taft, E. Mansour, M. Serafini, J. Duggan, A. J. Elmore, A. Aboulnaga, A. Pavlo, and M. Stonebraker, "E-store: Fine-grained elastic partitioning for distributed transaction processing systems," *Proceedings of the VLDB Endowment*, vol. 8, no. 3, pp. 245–256, 2014.

[37] P. Ruan, T. T. A. Dinh, D. Loghin, M. Zhang, G. Chen, Q. Lin, and B. C. Ooi, "Blockchains vs. distributed databases: Dichotomy and fusion," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 1504–1517.

[38] T. Asanuma, T. Miyamae, and Y. Yamaoka, "Synchro: Block-generation protocol to synchronously process cross-shard transactions in state sharding," *arXiv preprint arXiv:2309.01332*, 2023.



Ao Liu is working toward Ph.D. degree in School of Cyber Science and Engineering, Wuhan University, Wuhan. His research interests include blockchain security, cybersecurity.



Jing Chen received the Ph.D. degree in computer science from Huazhong University of Science and Technology, Wuhan. He worked as a full professor in Wuhan University from 2015. He is the deputy Dean of the School of Cyber Science and Engineering at Wuhan University. His research interests in computer science are in the areas of network security, cloud security. He has published more than 100 research papers in many international conferences and journals, such as USENIX Security, CCS, INFOCOM, TDSC, TIFS, TMC, TC, TPDS et al. He acts as

a reviewer for many conferences and journals, such as IEEE INFOCOM, IEEE Transactions on Information Forensics and Security, IEEE Transactions on Dependable and Secure Computing and IEEE/ACM Transactions on Networking.



Kun He received the Ph.D. degree in computer science from the Computer School, Wuhan University. He is currently an Associate Professor with Wuhan University. His research interests include cryptography, network security, cloud security, AI security, and blockchain. He has published more than 30 research papers in many international journals and conferences, such as USENIX Security, CCS, TIFS, and TDSC.



Teng Li received the BS degree from the school of computer science and technology, Xidian University, China, in 2013, and the PhD degree from the school of computer science and technology, Xidian University, China, in 2018. He is currently an associate professor with the school of cyber engineering, Xidian University, China. His current research interests include wireless and mobile networks, distributed systems and intelligent terminals with focus on security and privacy issues.



Ruiying Du is currently a Professor with the School of Cyber Science and Engineering, Wuhan University. Her research interests include network security, wireless network, and mobile computing. She has published over 80 research papers in many international journals and conferences, such as the IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, INFOCOM, SECON, TrustCom, and NSS.



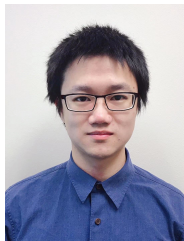
Jianfeng Ma received the BS degree in computer science from Shaanxi Normal University, in 1982, and the MS degree in computer science from Xidian University, in 1992, and the PhD degree in computer science from Xidian University, in 1995. Currently, he is the director of Department of Cyber engineering and a professor with the School of Cyber Engineering, Xidian University. He has published more than 150 journal and conference papers. His research interests include information security, cryptography, and network security.



Jiahua Xu is Associate Professor in Financial Computing, and Programme Director of the MSc Emerging Digital Technologies at UCL. She is also affiliated to the UCL Centre for Blockchain Technologies. Her research focuses on blockchain economics and decentralized finance. She has published in Usenix Security, ACM IMC, FC, IEEE ICDCS and IEEE ICBC. She has reviewed for Advances in Complex Systems, Computer Networks, Transactions on the Web and Cities.



Cong Wu is currently a Research Fellow at Cyber Security Lab, Nanyang Technological University, Singapore. He received Ph.D. degree at School of Cyber Science and Engineering, Wuhan University in 2022. His research interests include AI system security and Web3 security. His research leading outcomes have appeared in USENIX Security, ACM CCS, IEEE TDSC.



Yebo Feng is a research fellow in the School of Computer Science and Engineering (SCSE) at Nanyang Technological University (NTU). He received his Ph.D. degree in Computer Science from the University of Oregon (UO) in 2023. His research interests include network security, blockchain security, and anomaly detection. He is the recipient of the Best Paper Award of 2019 IEEE CNS, Gurdeep Pall Graduate Student Fellowship of UO, and Ripple Research Fellowship. He has served as the reviewer of IEEE TDSC, IEEE TIFS, ACM TKDD, IEEE

JSAC, IEEE COMST, etc. Furthermore, he has been a member of the program committees for international conferences including SDM, CIKM, and CYBER, and has also served on the Artifact Evaluation (AE) committees for USENIX OSDI and USENIX ATC.