A Scalable Blockchain Framework for Secure Data and Computational Resource Management in SDN

Debashis Das*, La Chiara Landrum*, Pushpita Chatterjee*, and Uttam Ghosh*

*Department of CS and DS, Meharry Medical College, Nashville, TN, USA
debashis.das@ieee.org, llandrum24@mmc.edu, pushpita.c@ieee.org, ghosh.uttam@ieee.org

2024 IEEE Globecom Workshops (GC Wkshps) | 979-8-3315-0567-7/24/$31.00 ©2024 IEEE | DOI: 10.1109/GCWkshp64532.2024.11100329

***Abstract*—The rapid evolution of Software-Defined Networking (SDN) has transformed network management by decoupling the control and data planes. It provides centralized control, enhanced flexibility, and programmability of network management services. However, this centralized control introduces security vulnerabilities and challenges related to data integrity, unauthorized access, and resource management. In addition, it brings forth significant challenges in secure and scalable data storage and computational resource management. These challenges are further increased by the need for real-time processing and the ever-increasing volume of data. To address these challenges, this paper presents a scalable blockchain-based framework for security and computational resource management in SDN architectures. The proposed framework ensures decentralized and tamper-resistant data handling and utilizes smart contracts for automated resource allocation. Due to the need for advanced security and scalability in SDN networks, this work incorporates sharding to improve parallel processing capabilities. The performance of sharded versus non-sharded blockchain systems under various network conditions is evaluated. Our findings demonstrate that the sharded blockchain model enhances scalability and throughput with robust security and fault tolerance. The framework is also assessed for its performance, scalability, and security to enhance SDN resilience against data breaches, malicious activities, and inefficient resource distribution.**

***Index Terms*—Decentralized computing, Network resilience, Secure data transmission, Data storage security, Fault tolerance.**

I. Introduction

Software-defined networking (SDN) has changed how we manage networks by separating the control functions from the actual data flow to control network resources from a central point [1]. This allows network administrators to manage and control the entire network from a centralized controller. It also provides the development and management of smart communities and wireless networks by providing flexibility and centralized control. It can dynamically allocate bandwidth and optimize performance based on user demand and network conditions to support various services like smart lighting, traffic management, and energy distribution [2]. However, this centralization nature of SDN creates several issues, such as data integrity & tampering, scalability & performance, and transparency & trust [3]. These issues underscore the need for reliable ways to protect both data storage and network resources. If an attacker gains access to the central controller, they could potentially disrupt the entire network. In addition, if the central controller is compromised, sensitive data flowing through the network could be exposed to unauthorized access or be altered without detection [4]. These challenges can have severe consequences, including network downtime, loss of sensitive data, and decreased trust among users and stakeholders. Therefore, there are required innovative solutions that can secure SDN environments without sacrificing their efficiency and ensure robust and trustworthy network management.

Meanwhile, Blockchain technology provides a decentralized framework and distributed storage that inherently resists tampering and makes secure SDN environments [1]. The control plane in SDN can be decentralized across multiple nodes by integrating blockchain technology, where each maintains a copy of the network state within an immutable ledger [5]. This decentralized approach ensures that even if one node is compromised, the network as a whole remains secure and operational. Blockchain's consensus mechanisms can be used to validate network changes and transactions to prevent unauthorized modifications and ensure data integrity [6]. The transparency provided by blockchain allows all network participants to verify operations and reduce the risk of malicious activities. Thus, this integration not only addresses the security weaknesses of centralized SDN but also enhances scalability, trustworthiness, resilience, and robust network infrastructure.

Therefore, this paper proposes a blockchain-based SDN controller model that enhances network security, scalability, and fault tolerance by developing decentralized control, secure data storage, and distributed computing. The proposed model employs sharding to divide the blockchain network into smaller, manageable segments (shards) to process transactions in parallel. The proposed cross-shard communication and coordination mechanisms ensure that transactions involving multiple shards are handled consistently and securely. The smart contracts are implemented to enforce access control policies to ensure only authorized entities can access or modify network configurations and data transactions. Furthermore, the proposed model ensures a robust security framework capable of protecting the network from various attacks, including tampering, unauthorized access, and data breaches. Furthermore, the key contributions of this work are:

- This paper proposes a decentralized SDN architecture where multiple blockchain nodes act as distributed controllers to resolve fault tolerance and improve network security.
- This paper provides a tamper-proof and transparent data

storage mechanism for SDN configurations and transactions by employing hash-based cryptography.

- In addition, the Sharding technique is utilized to divide the blockchain network into smaller ones and to allow parallel transaction processing to significantly improve throughput and overall network efficiency.
- The paper undergoes rigorous security analysis for using sharding to evaluate its effectiveness including data consistency, scalability, overhead cost, and throughput.

The remainder of the paper is organized as follows. In section II, a few recent existing Blockchain-enabled SDN frameworks are discussed with their limitations. Section III presents the proposed blockchain-based SDN system and discusses the overall implementation process. Then, section IV performs a security analysis of the proposed system model. Section V presents the experimental results and performance evaluations of the proposed system. Finally, section VI summarizes our work and discusses possible future research.

II. Related Works

In recent research, the integration of SDN and Blockchain has emerged as a promising approach for enhancing security, scalability, and privacy in cloud storage and IoT networks. Several studies have demonstrated the effectiveness of using SDN's centralized control for improved reliability and flexibility. Blockchain's decentralized nature ensures robust trust and security in transaction processes. In literature [7], the authors proposed the Block-SDoTCloud architecture, which uses SDN and Blockchain to enhance security in cloud storage networks. The architecture utilizes a centralized SDN controller for improved reliability, flexibility, and load balancing. However, the system's performance under varying network conditions is the key challenge.

The authors in [8] proposed a network infrastructure that integrates SDN and Blockchain to create a secure and adaptable framework for next-generation smart cities. However, the scalability of the framework as the number of IoT devices continues to grow and the potential latency introduced by Blockchain transactions remains challenging. In literature [9], this work introduces a blockchain-based forensic architecture within SDN for Internet of Things (IoT) environments to address challenges in digital forensics such as data integrity, evidence deletion, and alteration. This architecture establishes a Chain of Custody (CoC) using blockchain, integrates flow table rules on SDN switches for different types of traffic (VoIP, FTP, HTTP), and utilizes a Linear Homomorphic Signature (LHS) algorithm for user validation. However, the remaining challenges are the scalability and efficiency of the forensic architecture as the network grows, particularly in handling large volumes of IoT traffic and complex security demands.

In [10], this work presents "DistB-SDCloud," a blockchain-based SDN architecture for cloud computing platforms in Smart Industrial Internet of Things (IIoT) applications. The remaining challenges are the architecture's scalability as the number of IoT devices and data volumes increase. The study [11] highlights the importance of access control as a critical defense mechanism for organizations to ensure cybersecurity and comply with data privacy in SDN environments. They discussed the scalability and adaptability of access control and data management mechanisms across diverse and dynamic environments are remaining challenges. Further research is needed to address these challenges. Thus, our work focuses on addressing these concerns by proposing a solution that enhances security, scalability, and efficiency within SDN-Blockchain integrated networks.

TABLE I: **Useful notations and their definitions.**

Notation	Description
$\mathbb{N}$	Total number of controllers in the blockchain network (nodes).
$\mathbb{S}_i$	Local state maintained by node i (configuration and status of the network managed by the SDN controller).
$\mathcal{M}$	Threshold number of nodes required to reach consensus
$\mathbb{S}'$	New proposed network state after an update.
$\mathbb{C}$	Number of compromised nodes in the network.
$\mathbb{U}'$	Proposed update to the network state.
$\mathcal{K}$	Shard in the network.
$\mathbb{N}_j$	Number of nodes within a shard.
$\delta(t)$	Additional nodes required for consensus during high-threat periods.
$\mathcal{T}_j$	A data transaction in the blockchain network.
$\mathcal{D}_j$	Data associated with transaction $\mathcal{T}_j$.
$\mathcal{S}_j$	Leaf node representing the hash of a transaction in the Merkle tree.
$\mathbb{N}_i$	Non-leaf node in the Merkle tree (hash of its child nodes.)
$\mathcal{R}$	Merkle Root, representing the hash of all transactions.
$\mathcal{K}'$	Adversary attempting to bypass access control.
$\mathcal{F}$	A distributed application (Dapp) within the SDN network.
θ_0	Initial state of the system before executing smart contracts.
θ_i	Output state after executing smart contract η_i.
Φ	Set of validators in the network.
ϑ	Validator in the network.
$\mathbb{H}(\theta_i)$	Cryptographic hash of state θ_i.
$\mathcal{T}$	Total number of transactions processed by the network.
$\oplus$	the concatenation of two hashes

III. Proposed System Model

The proposed system model integrates blockchain technology with SDN to create a decentralized and secure network architecture. The key components of this architecture include a decentralized controller network, secure data storage mechanisms, a distributed computing framework, and scalability solutions. Table I presents the useful notations and their definitions.

A. Decentralized SDN Controller Network

Instead of relying on a single centralized controller, the proposed system utilizes multiple blockchain nodes acting as distributed controllers. Each controller maintains a copy of the SDN state, and blockchain consensus mechanisms ensure that all nodes agree on the network state. This decentralization enhances fault tolerance and reduces the risk of a single point of failure. Each node i maintains a local state $\mathbb{S}_i$, which is the current configuration and status of the network managed by the SDN controller. In a Byzantine Fault Tolerance (BFT) model [12], consensus is reached when at least $\mathbb{M}$ out of $\mathbb{N}$ nodes agree on the network state. The threshold $\mathbb{M}$ is typically defined as:

$$\mathbb{M} = \left\lceil \frac{2\mathbb{N}}{3} \right\rceil \tag{1}$$

This condition ensures that the network can tolerate up to $\mathbb{N} - \mathbb{M}$ compromised nodes to maintain consensus. For a decision or network update to be validated, the states $\mathbb{S}_i$ across all nodes must reach consensus, i.e., $\mathbb{S}_1 = \mathbb{S}_2 = \cdots = \mathbb{S}_{\mathbb{N}}$. When a network change occurs, it is proposed by a node i and propagated across the network. Other nodes validate the change by comparing it with their current state $\mathbb{S}_j$ and the proposed state. Therefore, consensus is achieved when $\mathbb{M}$ nodes validate and adopt the new state. The new state $\mathbb{S}'$ is accepted if the following condition is met:

$$\sum_{i=1}^{\mathbb{N}} \text{agree}(\mathbb{S}'_i) \geq \mathbb{M}, \tag{2}$$

$$\text{Where, agree}(\mathbb{S}'_i) = \begin{cases} 1 & \text{if node } i \text{ agrees new state;} \\ 0 & \text{otherwise;} \end{cases} \tag{3}$$

If $\mathbb{C}$ is the number of compromised nodes, the system is secure as long as $\mathbb{C} < \mathbb{N} - \mathbb{M}$. Beyond this threshold, the network may suffer from inconsistencies, as malicious nodes could overpower the honest ones. Therefore, network updates are proposed by node i and require validation by at least $\mathbb{M}$ nodes. If an inconsistency is detected after an update, the network can roll back to the last agreed state $\mathbb{S}_{prev}$. Here, the rollback is initiated if the number of nodes supporting the rollback exceeds the threshold $\mathbb{M}$. The network reverts to $\mathbb{S}_{prev}$ if the following rollback condition is met.

$$\sum_{i=1}^{\mathbb{N}} rollback(\mathbb{U}'_i) \geq \mathbb{M} \tag{4}$$

Instead of requiring each node to individually sign a transaction or state update, a threshold signature scheme allows $\mathbb{M}$ nodes to collectively generate a single valid signature [13]. If $\mathbb{N}$ nodes attempt to sign a transaction, a valid signature is produced only if at least $\mathbb{M}$ nodes participate. Then no single compromised node can produce a valid signature on its own.

Therefore, the network is divided into $\mathcal{K}$ shards, with each shard managing a subset of nodes and transactions. Each shard reaches consensus independently to reduce the overall load and increase scalability. Consensus within a shard is achieved if $\mathbb{M}_j = \left\lceil \frac{2\mathbb{N}_j}{3} \right\rceil$ nodes agree on the shard's state. The global network state is then an aggregation of the consensus states from each shard. The system dynamically adjusts $\mathbb{M}$ based on real-time network conditions and potential threats. For instance, during a detected attack, $\mathbb{M}$ could be increased to enhance security under normal conditions, and a lower $\mathbb{M}$ could be used to improve performance. Therefore, $\mathbb{M}$ becomes a function of time and threat level as follows:

$$\mathbb{M}(t) = \left\lceil \frac{2\mathbb{N}(t)}{3} \right\rceil + \delta(t) \tag{5}$$

Thus, the proposed model ensures that the proposed architecture remains secure, fault-tolerant, scalable, and capable of handling large-scale and complex network environments.

B. Secure Data Storage Management

This section demonstrates how the blockchain mechanism ensures the integrity, transparency, and security of data transactions, and how smart contracts enforce access control. The proposed system utilizes cryptographic techniques and smart contracts to safeguard network configurations and data transactions and create a tamper-proof and decentralized storage model [14]. It covers both the security of the data structure (using cryptographic hashes and Merkle trees) and the correctness of access control (using smart contracts). Data transactions in the SDN are recorded on the blockchain, where each transaction is cryptographically linked to the previous one. Each transaction $\mathcal{T}_j$ in the blockchain is represented as:

$$\mathcal{T}_j = (\hbar(\mathcal{T}_{j-1}), \mathcal{D}_j) \tag{6}$$

A cryptographic hash function $\hbar$ has the following properties: **1)** $\hbar(x)$ always produces the same hash for the same input x. The hash function is collision-resistant which means it is computationally infeasible to find two different inputs that produce the same hash. **2)** A hash $\hbar(x)$ is computationally infeasible to find an input x' such that $\hbar(x') = \hbar(x)$. **3)** It is computationally infeasible to find two distinct inputs x and y such that $\hbar(x) = \hbar(y)$. **4)** A small change in the input x significantly changes the output $\hbar(x)$. Suppose an adversary tries to modify transaction $\mathcal{T}_j$ to $\mathcal{T}'_j$, where

$$\mathcal{T}'_j = \left(\hbar(\mathcal{T}_{j-1}), \mathcal{D}'_j\right) \tag{7}$$

The new transaction $\mathcal{T}'_j$ would produce a different hash $\hbar(\mathcal{T}'_j)$. Consequently, the hash in transaction $\mathcal{T}_{j+1}$ would no longer match $\hbar(\mathcal{T}'_j)$ and breaks the chain. Thus, to verify the integrity of large numbers of transactions, they are organized into a Merkle tree. A Merkle tree is a binary tree where each leaf node represents the hash of a transaction, and each non-leaf node is the hash of its child nodes. Each leaf node $\mathcal{L}_j$ in the Merkle tree represents the hash of a transaction: $\mathcal{T}_j = \hbar(\mathcal{T}_j)$. The root of the Merkle tree (known as the Merkle Root [15]) $\Re$ is computed as follows:

$$\Re = \hbar\left(\hbar(\mathcal{T}_1) \oplus \hbar(\mathcal{T}_2)\right) \oplus \hbar\left(h(\mathcal{T}_3) \oplus \hbar(\mathcal{T}_4)\right) \ldots \tag{8}$$

The Merkle proof consists of the hashes of the sibling nodes from the leaf node corresponding to $\mathcal{T}_j$ to the Merkle Root. If the computed root from the proof matches the stored Merkle Root, the transaction is valid and has not been tampered with. Therefore, access control in the proposed model is enforced through smart contracts, which are self-executing contracts with the terms of the agreement directly written into code. The smart contract defines an access control policy $\mathcal{A}_{\mathcal{K}}(\mathcal{D}_j)$ for each data transaction $\mathcal{D}_j$. $\mathcal{A}_{\mathcal{K}}(\mathcal{D}_j) = True$ if and only if the entity $\mathcal{K}$ satisfies the conditions defined in the smart contract for accessing or modifying $\mathcal{D}_j$. It checks whether entity $\mathcal{K}$ holds a specific role (e.g., Admin controller) and whether the current time is within a predefined access window. Now, consider an adversary $\mathcal{K}'$ who tries to bypass the smart contract by directly modifying $\mathcal{D}_j$. Since $\mathcal{D}_j$ is stored on the blockchain, any unauthorized modification attempt would fail the consensus check (as $\mathcal{A}_{\mathcal{K}'}(\mathcal{D}_j) = False$). The network nodes would reject the transaction and, in this way maintain the integrity and security of the data.

C. Distributed Computational Resource Management

The distributed computational resource management in the proposed system uses smart contracts and a consensus mechanism to ensure secure, consistent, and coordinated operations across the network. This section illustrates how state transitions are managed, consensus is reached, and data consistency is maintained. It uses smart contracts to deploy and execute $\mathcal{F}$ [16]. It performs decentralized computing tasks across the network and ensures that the operations are secure, consistent, and resilient to tampering. $\mathcal{F}$ within the SDN operates as a sequence of smart contracts $\eta_1, \eta_2, \ldots, \eta_m$. Each smart contract is a piece of code that performs a specific computation or operation within the $\mathcal{F}$. Each smart contract η_i takes the input state θ_{i-1} from the previous contract and produces a new state θ_i. The output state θ_i of one contract may serve as the input state θ_i for another contract η_{i+1} and it creates dependencies between contracts. The entire execution sequence $\theta_0 \rightarrow \theta_1 \rightarrow \cdots \rightarrow \theta_m$ must be consistent across all network nodes. The consensus mechanism ensures that all nodes in the network agree on the state of the system after each smart contract execution. This is essential for preventing unauthorized changes and maintaining consistency across the SDN network. Each validator $\vartheta \in \Phi$ proposes a state θ_i after executing the contract η_i. Validators vote on the proposed state θ_i. A consensus is reached if a sufficient majority (see equation 1) for BFT of validators agree on the state θ_i. The state transition is accepted if: $AgreedState(\theta_i) = MajorityVote(\theta_i)$. If the majority of validators agree on θ_i, then θ_i is accepted as the new state of the system. All nodes in the network independently execute the smart contracts and propose the resulting state θ_i. The use of cryptographic hashes and the consensus mechanism prevents any unauthorized changes to the state of $\mathcal{F}$. Nodes verify the proposed state by comparing the hash of the executed state with the proposed state hash. If $\hbar(\theta_i)$ matches the proposed state, it is accepted; otherwise, it is rejected. Once consensus is reached on θ_m, the final state of the distributed application, θ_m is committed to the blockchain. This ensures that the results of the distributed computation are secure, consistent, and verifiable by all participants.

D. Scalability management

To address scalability challenges in blockchain-enhanced SDN systems, the sharding technique is designed to improve transaction throughput and overall network efficiency without compromising security. Sharding involves dividing the blockchain network into smaller and more manageable segments called shards. Each shard operates independently and processes a subset of transactions. Without sharding, each of the $\mathbb{N}$ nodes processes all $\mathcal{T}$ transactions. With sharding, the network is divided into $\mathcal{K}$ shards, and each shard processes $\frac{\mathcal{T}}{\mathcal{K}}$ transactions. If the total number of nodes $\mathbb{N}$ is uniformly distributed across the shards, then each shard will have approximately $\frac{\mathcal{N}}{\mathcal{K}}$ nodes. The processing capacity of each shard is $\frac{\mathcal{T}}{\mathcal{K}}$, and the total network processing capacity becomes $\mathcal{K} \times \frac{\mathcal{T}}{\mathcal{K}} = \mathcal{T}$, which is the same as without sharding but distributed among shards. The load on each node is reduced to $\frac{\mathcal{T}}{\mathcal{K}}$, compared to $\mathcal{T}$ without sharding. Each node only processes a fraction of the total transactions, reduces congestion, and improves throughput. Formally: Load per node = $\frac{\mathcal{T}}{\mathcal{K}}$ and total throughput = $\mathbb{N} \times \frac{\mathcal{T}}{\mathcal{K}}$.

However, sharding introduces complexities in coordinating transactions across multiple shards and maintaining a consistent global state. So, cross-shard transactions and atomic commits are used to manage this consistency. A cross-shard protocol ensures that all shards involved in a multi-shard transaction reach a consensus on the transaction outcome. Let $\mathcal{T}_{ij}$ represent a transaction that affects both shard i and shard j. This is modeled as: $\mathcal{T}_{ij}$ = Coordination between $\mathcal{K}_i$ and $\mathcal{K}_j$ where $\mathcal{K}_i \in$ Shard$i, \mathcal{K}_j \in$ Shard$j$. For atomicity, a coordinator ensures that either all shards involved in the transaction commit the transaction or none do. This can be modeled as:

$$\text{Commit}(\mathcal{T}_{ij}) = \begin{cases} 1 & \text{if all } \mathcal{K}_i \text{ involved agree} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

The global state $\mathcal{K}_{\text{global}}$ of the network is the sum of the states of all shards, represented as:

$$\mathcal{K}_{\text{global}} = \sum_{i=1}^{\mathcal{K}} \mathcal{K}_i \tag{10}$$

State changes within a shard must be propagated to other relevant shards to maintain consistency. The propagation of state $\mathcal{K}_i$ from shard i to shard j is essential when $\mathcal{T}_{ij}$ affects multiple shards.

$$\mathcal{K}_j \leftarrow \text{Propagate}(\mathcal{K}_i), \forall i, j \in \{1, 2, \ldots, \mathcal{K}\} \tag{11}$$

An atomic commit protocol can be used to ensure that the global state is consistent across all shards: each shard prepares for the transaction and shares its readiness with others and all shards commit the transaction only if all are prepared, which is shown in equation 12. The consensus mechanism ensures that all nodes within a shard agree on the shard's state. That means: Consensus$(\mathcal{K}_i)$ = Agreement among nodes in $\mathcal{K}_i$

$$\text{Commit}(\mathcal{T}) = \begin{cases} 1 & \text{if Prepared}(\mathcal{K}_i)\ \forall \mathcal{K}_i \in \text{Involved Shards} \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

To maintain fault tolerance, the system must ensure that the global state is correct even if some nodes fail or act maliciously. This is often achieved using BFT mechanisms, which require fault tolerance condition: $\mathbb{N} \geq 3f + 1$. Therefore, the scalability of the proposed model can be improved by ensuring consistency and fault tolerance using shards.

IV. Security Analysis

The proposed model integrates blockchain technology with sharding to enhance security and reliability. The security analysis focuses on two key aspects:

Tamper-proof Data storage: Tamper-proof data storage ensures that once data is recorded, it cannot be altered or deleted without detection. Suppose an attacker wants to alter a transaction in block B_k. The attacker would need to change B_k and then recalculate hashes for all subsequent blocks $B_{k+1}, B_{k+2}, \ldots, B_n$. The difficulty of this task depends on the computational effort required to recalculate the hashes: Effort $=$ Difficulty $(H(B_k), H(B_{k+1}), \ldots, H(B_n))$. The difficulty of altering n blocks is exponential for n which makes it infeasible to tamper if the network's consensus mechanism is robust. The probability of a single node successfully mining a block (validating a transaction) is $\frac{1}{2^{\text{Difficulty}}}$. To successfully alter a transaction, an attacker would need to control a fraction f of the total network computational power, where $f > \frac{1}{2}$. Thus, this is computationally infeasible for a well-distributed network to achieve.

Risk of Single Point of Failure: To analyze the security of the proposed model we examine two key aspects: the risk of a single point of failure and the probability of successful attacks. In the proposed system, each node operates independently, and the probability that any individual node fails is denoted as ρ. To understand the system's resilience, we need to compute the probability that at least one node fails. First, calculate the probability that a single node does not fail, which is $1-\rho$. If all nodes are independent, the probability that none of the n nodes fail is Probability of no failures $= (1-\rho)^n$. Therefore, the probability that at least one node fails is:

$$\mathcal{P}_{\text{failure}} = 1 - (1-\rho)^n \tag{13}$$

As n increases, $(1-\rho)^n$ decreases because even if each node has a relatively high failure probability, the combined probability of no failures decreases exponentially. Consequently, $\mathcal{P}_{\text{failure}}$ approaches 1, which indicates the likelihood of at least one node failing is high when n is large. For large n, $\mathcal{P}_{\text{failure}}$ becomes more stable and less sensitive to individual node failures. This means the system becomes increasingly resilient to single points of failure.

Attack Probability: In a decentralized system, an attacker needs to compromise a fraction f of the total number of nodes to disrupt the system. The probability $\mathcal{P}_{\text{attack}}$ indicates that an attacker can successfully disrupt the system is calculated by considering the fraction of compromised nodes as stated below:

$$\mathcal{P}_{\text{attack}} = \frac{\text{Number of compromised nodes}}{\text{Total nodes}} \tag{14}$$

If f is the fraction of compromised nodes required to cause disruption, then: $\mathcal{P}_{\text{attack}} = f$. The effectiveness of an attack depends on how many nodes are compromised relative to the total number of nodes. If f is small (i.e., the fraction of compromised nodes needed is low), the system is more vulnerable. Conversely, if f is high (i.e., a large fraction of nodes needs to be compromised), the system is more secure.

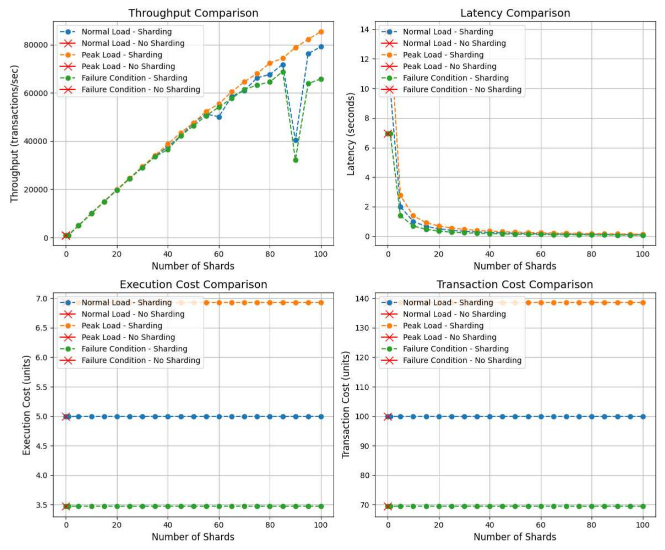

Fig. 1: Performance metrics analysis with and without sharding.

In a well-distributed network with a high number of nodes, f becomes relatively large and makes it difficult for an attacker to compromise a sufficient fraction of nodes. The security analysis of the proposed sharded model demonstrates that decentralized control through sharding significantly improves both failure resilience and attack resistance. Sharding enhances security by distributing data and processing load across multiple nodes (shards). However, it reduces the risk of single points of failure and makes it harder for attackers to compromise a significant fraction of the system. The proposed model's decentralized nature significantly reduces the probability of system failure due to individual node failures, as shown in Fig. 2. With more nodes n, the system's resilience increases because the probability that at least one node fails decreases, and the overall system stability improves.

V. Results and Discussion

The primary goal of this experiment is to evaluate the impact of sharding on several key aspects of the proposed system. The experiment is conducted in a simulated environment where sharding is applied to a transactional system. Mininet (to simulate the SDN part of the setup for handling data flow and traffic management), Ethereum (to create a decentralized application layer in the test setup), and GNS3 (to complement Mininet by adding more advanced routing and network configurations) tools are used in the test setup. The system is modeled to evaluate various metrics related to fault tolerance, overhead costs, throughput, and scalability. The experiment evaluates the performance of a sharded system, where the number of shards is 100. The system's performance and risks associated with sharding are simulated based on a fixed number of transactions and transactions per second rate. The average latency of transactions and latency for transactions within each shard are measured from initiation to completion. Fig 1 provides how sharding impacts various performance and risk metrics and a clear comparison between sharded and non-sharded systems. As

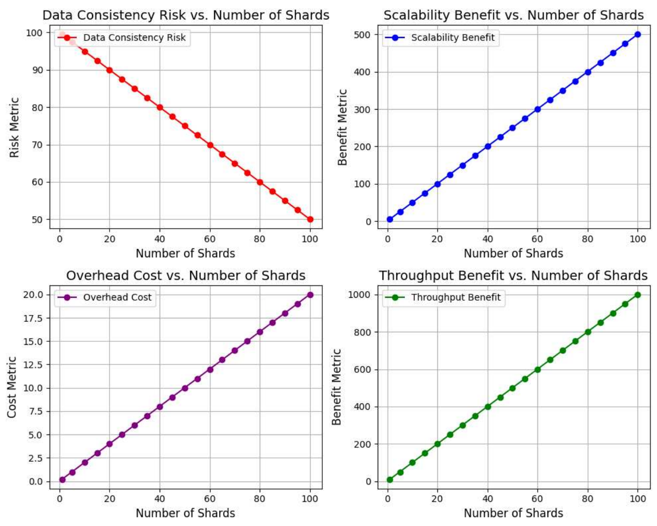

Fig. 2: Analysis of risk metrics associated with sharding.

the number of shards increases, throughput typically improves because the transaction load is distributed across multiple nodes, which reduces the burden on any single node. on the other hand, latency may initially decrease with more shards due to parallel processing but can increase if overhead costs and communication delays between shards become significant. The added complexity of managing inter-shard communication and synchronization can also introduce coordination bottlenecks. Moreover, the risk of data inconsistency increases as the number of shards grows, due to the complexity of maintaining consistency across multiple nodes. Data integrity and coherence across distributed shards can require additional mechanisms, which may further impact system performance. Generally, fault tolerance risk decreases with more shards since the failure of a single shard affects only a portion of the overall system. This balance between performance, latency, and fault tolerance with sharding is depicted in Fig. 2.

VI. Conclusion

The proposed blockchain-based approach addresses the critical challenges of scalability and security in data storage and computing within SDN environments. The integration of blockchain with SDN mitigates the inherent vulnerabilities of centralized control by providing tamper-proof data storage, verifiable transactions, and automated access control through smart contracts. The proposed model creates a secure, decentralized, and scalable SDN network architecture. This work not only contributes to the ongoing development of secure SDN architectures but also finds the way for more resilient and adaptable networking frameworks in the era of advanced data and resource management. The performance improvements demonstrated in our evaluation underscore the potential of this approach to serve as a foundational model for future secure and scalable SDN infrastructures. Future work will focus on further optimizing the integration of blockchain with SDN to reduce latency and improve transaction throughput. Further research will also investigate the use of advanced consensus algorithms and cross-chain interoperability to enhance the scalability and efficiency of the system.

Acknowledgement

This work was also supported by the National Science Foundation under award numbers 2219741 and 2401928.

References

[1] D. Das, S. Banerjee, K. Dasgupta, P. Chatterjee, U. Ghosh, and U. Biswas, "Blockchain enabled sdn framework for security management in 5g applications," in *Proceedings of the 24th International Conference on Distributed Computing and Networking*, ser. ICDCN '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 414–419. [Online]. Available: https://doi.org/10.1145/3571306.3571445

[2] A. Muthanna, A. A. Ateya, A. Khakimov, I. Gudkova, A. Abuarqoub, K. Samouylov, and A. Koucheryavy, "Secure and reliable iot networks using fog computing with software-defined networking and blockchain," *Journal of Sensor and Actuator Networks*, vol. 8, no. 1, p. 15, 2019.

[3] J. C. C. Chica, J. C. Imbachi, and J. F. B. Vega, "Security in sdn: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 159, p. 102595, 2020.

[4] M. Rahouti, K. Xiong, Y. Xin, S. K. Jagatheesaperumal, M. Ayyash, and M. Shaheed, "Sdn security review: Threat taxonomy, implications, and open challenges," *IEEE Access*, vol. 10, pp. 45 820–45 854, 2022.

[5] U. Ghosh, D. Das, P. Chatterjee, and S. Shetty, "Quantum-enabled blockchain for data processing and management in smart cities," in *2023 IEEE 24th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, 2023, pp. 425–430.

[6] D. Das, S. Banerjee, P. Chatterjee, U. Ghosh, W. Mansoor, and U. Biswas, "Design of an automated blockchain-enabled vehicle data management system," in *2022 5th International Conference on Signal Processing and Information Security (ICSPIS)*, 2022, pp. 22–25.

[7] A. Rahman, M. J. Islam, M. Saikat Islam Khan, S. Kabir, A. I. Pritom, and M. Razaul Karim, "Block-sdotcloud: Enhancing security of cloud storage through blockchain-based sdn in iot network," in *2020 2nd International Conference on Sustainable Technologies for Industry 4.0 (STI)*, 2020, pp. 1–6.

[8] S. Rani, H. Babbar, G. Srivastava, T. R. Gadekallu, and G. Dhiman, "Security framework for internet-of-things-based software-defined networks using blockchain," *IEEE Internet of Things Journal*, vol. 10, no. 7, pp. 6074–6081, 2023.

[9] M. Pourvahab and G. Ekbatanifard, "An efficient forensics architecture in software-defined networking-iot using blockchain technology," *IEEE Access*, vol. 7, pp. 99 573–99 588, 2019.

[10] A. Rahman, M. J. Islam, S. S. Band, G. Muhammad, K. Hasan, and P. Tiwari, "Towards a blockchain-sdn-based secure architecture for cloud computing in smart industrial iot," *Digital Communications and Networks*, vol. 9, no. 2, pp. 411–421, 2023.

[11] L. Golightly, P. Modesti, R. Garcia, and V. Chang, "Securing distributed systems: A survey on access control techniques for cloud, blockchain, iot and sdn," *Cyber Security and Applications*, vol. 1, p. 100015, 2023.

[12] X. Dai, L. Huang, J. Xiao, Z. Zhang, X. Xie, and H. Jin, "Trebiz: Byzantine fault tolerance with byzantine merchants," in *Proceedings of the 38th Annual Computer Security Applications Conference*, 2022, pp. 923–935.

[13] D. Das, K. Dasgupta, and U. Biswas, "A secure blockchain-enabled vehicle identity management framework for intelligent transportation systems," *Computers and Electrical Engineering*, vol. 105, p. 108535, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0045790622007509

[14] D. Das, U. Ghosh, P. Chatterjee, and S. Shetty, "Advanced federated learning-empowered edge-cloud framework for school safety prediction and emergency alert system," in *2023 IEEE 12th International Conference on Cloud Networking (CloudNet)*, 2023, pp. 507–512.

[15] S. T. Alvi, M. N. Uddin, and L. Islam, "Digital voting: A blockchain-based e-voting system using biohash and smart contract," in *2020 third international conference on smart systems and inventive technology (ICSSIT)*. IEEE, 2020, pp. 228–233.

[16] L. Besancon, C. F. Da Silva, P. Ghodous, and J.-P. Gelas, "A blockchain ontology for dapps development," *IEEE Access*, vol. 10, pp. 49 905–49 933, 2022.