

Blockchain-Driven Secure Data Sharing Framework for Edge Computing Networks

Fuad A. M. Al-Yarimi*, Ramzi Salah, and Khaled Mohamoud

Abstract: This study examines secure and effective data sharing methods for edge computing networks. Traditional methods of sharing data at the edge have issues with security, speed, and consensus. The goal is to develop a Blockchain-based Secure Data Sharing Framework (BSDSF) capable of improving data integrity, latency, and overall network efficiency for edge-cloud computing applications. BSDSF proposes using blockchain technology with Byzantine Fault Tolerance (BFT) and smart contract-based validation as a new method of secure data sharing. It has a two-tiered consensus protocol to meet the needs of edge computing, which requires instantaneous responses. BSDSF employs Byzantine fault tolerance to deal with errors and protect against attacks. Smart contracts automate validation and consensus operations, while edge computing processes data at the attack site. Node validation and failure detection methods monitor network quality and dependability, while system security ensures secure communication between nodes. BSDSF is an important step toward digital freedom and trust by protecting security and improving transaction reliability. The framework demonstrates a reduction in transaction latency by up to 30% and an increase in throughput by 25% compared to traditional edge computing models, positioning BSDSF as a pivotal solution for fostering digital freedom and trust in edge computing environments.

Key words: Byzantine Fault Tolerance (BFT); blockchain-driven; edge computing networks; consensus contract; data validation contract; node validation contract

1 Introduction

Edge computing emerges as a pivotal technology, addressing the growing demand for localized data processing to reduce latency and enhance efficiency in

our increasingly interconnected world. However, this advancement brings forth significant challenges, including ensuring robust security, maintaining data integrity, and achieving low-latency communication within decentralized networks. These hurdles underscore the critical need for innovative solutions that can seamlessly navigate the complexities of edge computing environments.

In this developing area of edge computing, the distribution and sharing of data among decentralized networks pose big challenges^[1]. The key issues are the security of network services against malice, low latency and high efficiency in data exchange, as well as integrity-preservation of information content^[2]. All of these challenges are complicated by the fact that edge network topologies are highly dynamic, simultaneously

- Fuad A. M. Al-Yarimi is with Department of Computer Science, King Khalid University, Muhayel Aseer 62529, Saudi Arabia. E-mail: fuadalyarimi@gmail.com.
- Ramzi Salah is with Faculty of Information Science and Technology, University Kebangsaan Malaysia, Bangi 43600, Malaysia. E-mail: dr.ramzi.salah@gmail.com.
- Khaled Mohamoud is with Department of Information System, King Khalid University, Muhayel Aseer 62529, Saudi Arabia. E-mail: kasgr@kku.edu.sa.

* To whom correspondence should be addressed.

Manuscript received: 2023-12-24; revised: 2024-02-14; accepted: 2024-03-04

making it more difficult to achieve a consensus and creating single points of failure^[3].

A typical threat which can occur in these networks are Byzantine faults whereby a node's erratic behavior exposes the system to dependence on error or malicious intent and affects, therefore, consensus building through reliable means^[4]. Given the decentralized nature of environments, current methods fail to balance performance with reliability via true distributed data sharing. However, a major bottleneck is the problem of verifying whether participating nodes and their shared data are actually authentic and eligible; this must be done at minimal computational cost with low latency. In order to overcome these difficulties, a strong foundation is required for edge computing networks that allows secure, fast and reliable transfer of information while protecting against Byzantine faults or other types unique to the location. Since the number of edge devices and data is growing by exponential factors, scalability must be considered to make such a system. The ultimate objective is to develop a completely new, decentralized consensus mechanism together with an edge computing system that are mutually compatible and help ensure secure sharing of data in the stakeholders' common network^[5].

The block chain, which underlies many of the same systems that handle bitcoin and other cryptocurrencies (such as Ethereum), is a misused technology used anywhere transactions need to be processed without requiring any central authority. Mechanisms such as consensus algorithms, proof of-work or stake and hashing ensure immutability, transparency and security. In addition to its digital currency roots, blockchain has also been applied in industries such as supply chain management, identity verification and secure cross-sectoral exchanges^[6].

Changing distributed computing paradigms model of edge computing intended to bring latency and bandwidth usage down by aggregating data close to the user center^[7]. The impetus for this approach stems from a search for real-time processing and the ubiquity of Internet of Things (IoT) devices. Services like autonomous vehicles, smart cities and real time-analytics need low latency and local processing, so edge computing is perfectly suited to these applications.

Edge networks, with their many devices and

environments, can be tough to secure. Traditional security methods often fail to provide sufficient data safety. So, for edge computing, we need flexible and adaptable security strategies.

Smart contracts in blockchain networks ensure smooth transactions with fewer disputes. In this framework, these contracts establish consensus rules, handle data sharing, and confirm sensor output.

Edge computing benefits from blockchain. With regard to the proposed model Blockchain-based Secure Data Sharing Framework (BSDSF), a renewed Byzantine Fault Tolerance (BFT) mechanism and smart contracts govern data rules. These features aid robustness, transparency, and efficiency. They're also great for dealing with trust and scalability issues. The framework can securely process and share data, even when faults or attacks occur. Its use of blockchain and smart contracts is an excellent addition to edge computing^[8].

This framework can be reliable for edge computing networks. Tasks involve setting up a stable protocol based on Practical Byzantine Fault Tolerance (PBFT)^[9], creating smart contracts for node checks, and automating data rules. Also, they aim to demonstrate performance benefits and to design secure data-sharing methods.

BSDSF combines blockchain technology's decentralization, advanced BFT mechanisms, and smart contract efficiency. This integration significantly improves edge computing network security, trust, and latency, resulting in better network performance. By addressing the issues listed above, BSDSF prepares edge computing for secure and effective data sharing. BSDSF, which ensures data integrity, confidentiality, and secure sharing in dynamic edge environments, represents a significant step toward overcoming these limitations. This article uses BSDSF as a lens to investigate how combining blockchain technology with edge computing technology could result in a strong, efficient, and trustworthy framework. This would allow for new apps that could transform industries and improve people's digital lives.

The framework integrates a renewed BFT mechanism alongside smart contracts, enhancing the network's resilience against erratic or malicious nodes. This approach ensures secure and reliable consensus building, crucial for decentralized networks where traditional verification methods fall short. By

leveraging blockchain technology's inherent security features, such as consensus algorithms and hashing, BSDSF provides a robust solution for authenticating participating nodes and their data with minimal computational overhead. Additionally, the integration of smart contracts automates consensus rules, data sharing, and sensor output verification, further fortifying the system against Byzantine faults. This comprehensive strategy allows BSDSF to ensure secure, fast, and reliable data transfer in edge computing environments, scaling effectively to accommodate the exponential growth of edge devices and data. The article provides a thorough understanding of BSDSF and its impact on edge computing. Section 2 discusses related research, which demonstrates contemporary contributions in the respective domain. Section 3 discusses the BSDSF's technical foundation, consensus protocol, and smart contracts. In Section 4, we use simulation models and security analyses to show how effective BSDSF is. Section 5 summarizes key findings and discusses the BSDSF's future impact on edge computing.

2 Related Research

In the world of secure data sharing and blockchain, experts have offered several methods. Liu et al.^[10] created a system for Mobile-Edge Computing (MEC). It combines blockchain with a learning method named Asynchronous Advantage Actor-Critic (A3C). The system guarantees safe data sharing with privacy-protecting algorithms and blockchain actions. The A3C method boosts its effectiveness and use of resources. It leads to better security, privacy, less energy use, and more capacity. Firdaus and Rhee^[11] focused on Vehicular Edge Computing Networks (VECNets). They handled challenges around data privacy, dependability, and security. They used cryptographic methods, smart contracts, and distributed ledger technologies for secure data access and sharing. They came up with a reward system to inspire data contribution. It improved data sharing and network functioning in a traffic situation. A safe way to share data was suggested by Karati and Das^[12]. They introduced a system that uses edge computing and codes to protect data. This suggests it achieves security and allows easy sharing of files. Tests show it works well. Meanwhile, Kang et al.^[13] created a blockchain solution. It makes sharing information safe in vehicular

edge computing and networks. It ensures privacy and security, thanks to coding and a reputation-incentive method. Numbers prove it spots harmful nodes and stops data attacks. Gawas et al.^[14] designed a system to safely share data in blockchain-enabled vehicle networks. Their system uses a two-tier authentication process in Vehicular Ad-hoc Networks (VANETs) that lets cars confirm each other's identity and share data based on trust and past actions. The results suggest the system improves both efficiency and security. Onik et al.^[15] presented a hierarchical blockchain method for securely sharing medical data while protecting patient privacy in edge computing. This method keeps personally identifiable information in Electronic Health Records (EHR) anonymous and ensures security through multi-level blockchain applications. Pan et al.^[16] proposed plan protects data privacy in smart transportation networks. Their plan uses a Cloud-assisted Attribute-Based Encryption (CP-ABE) method, a Trusted Privacy Domain (TPD) server, and a secure Message Authentication Code (MAC) mechanism. Karthikeyan and Madhavan^[17] proposed a Trust Model for Secure Data Sharing (TM-SDS) in edge computing networks. This model includes set-up phases, data scrambling with a type of encryption, and secure data handling with MAC. The focus here is on safe data transit and handling. Zhang et al.^[18] presented a blockchain-based data sharing model for vehicular networks. This model mixes Proof of Coin (PoC) and Proof of Work (PoW) consensus mechanics for data agreement and reputation management. They use a specific method for quantifying vehicle reputation based on subjective logic. Zhang et al.^[19] put forward a plan for protected data storage and sharing in blockchain's mobile edge computing, or MEC. This plan takes care of anonymous sign-ins, privacy safety, smart storage space distribution, and verifiable key handling. It proves to be a good stand against various risks and is comparably effective to existing methods. The differences between the Blockchain-Driven Secure Data Sharing Scheme (BDSSS)^[10] and the Secure Data Storage and Sharing Scheme (DSSS)^[19] underlines crucial points in edge computing. Here, better encryption methods are vital. Both BDSSS and DSSS use blockchain tech but adapt it for different uses: BDSSS aims at mobile-edge computing, while DSSS is centered on vehicle networks. BDSSS uses a type of learning method called asynchronous advantage actor

critic. This method increases energy efficiency for mobile and edge servers and boosts blockchain networks' work efficiency. This means they're creatively combining modern learning methods with blockchain technology to fine tune performance in specific computing settings. Differently, DSSS puts more focus on strengthening data security and computational efficiency. One of its main goals is to lighten the computational load, which is vital for the system to manage large amounts of data effectively. Plus, DSSS offers better privacy protections against semitrusted edge nodes, an essential aspect of data confidentiality in wide networks. In this review of related research, a variety of innovative solutions for safe data sharing and incorporating blockchain into edge computing are demonstrated. For example, Lei Liu et al.^[10] demonstrate how blockchain can improve privacy and security when combined with the A3C learning method. For example, Firdaus and Rhee^[11] used cryptographic techniques and smart contracts to improve data privacy and network dependability in their approach to vehicular edge computing networks. The solutions provided by Karati and Das^[12], Kang et al.^[13], and others up to Zhang et al.^[19] are all different approaches to improving data security, privacy, and computational efficiency in various edge computing scenarios. BSDSF is a comprehensive solution that incorporates the advances and lessons from these approaches. BSDSF takes a holistic approach to energy efficiency, privacy, and data integrity, rather than providing individual solutions. It tackles the complex issues of secure data sharing in edge computing with advanced blockchain, smart contract, and learning techniques. BSDSF advances previous research by incorporating a dual-layered approach to security and efficiency, resulting in a more secure, efficient, and scalable framework. BSDSF is an advancement to BDSSS and DSSS. It's designed to overcome issues in edge computing data sharing. These include scalability, security, and efficiency. With a robust emphasis on security, the BSDSF aims to solve key problems. These are necessary for safe, effective data management in big, complex edge computing areas.

3 Method and Material

The proposed consensus protocol, a variant of BFT, uses a two-tier method. Edge nodes rapidly reach a pre-

consensus for immediate data processing. Full nodes achieve network-wide consensus using a robust PBFT protocol, ensuring resilience. The proposed Smart contracts automate key functions. The validation contract sets node validation criteria like identity and trust scores. The data validation contract checks the transferred data. The consensus contract implements the BFT consensus protocol in the network. In data transfer protocol, a sender initiates the process. Edge nodes quickly handle pre-processing and validation. After pre-consensus on data validity, the transaction moves to the core network for full node consensus. The receiver accesses the data after this consensus. BSDSF uses cryptographic model for secure node communication and data validation. Regular smart contract audits identify vulnerabilities, and smart contracts also limit network access. To handle failures and attacks, it detects node performance issues and isolate malicious nodes. It also monitors network transactions and patterns to eliminate threats.

3.1 BSDSF architecture

Figure 1 outlines a design for BSDSF, striving to present steady, safe, and effective data exchange in a decentralized world for edge computing networks. It works through smart contracts and BFT. Here's how it works:

Blockchain infrastructure: The BSDSF is built on a structure which is blockchain-based containing various elements, each with its own unique role within the network. This structure is important for forming a protected data sharing system, forming the foundation of a spread-out ledger.

The system encompasses sender and receiver elements, playing a crucial role in sharing data. Edge nodes can be found near data sources at the edge of the network, made for quick data handling and verification, serving real-time application needs. The main network consists of elements that possess a full, latest copy of the blockchain, doing complex calculations and keeping network integrity intact.

Consensus agreement (BFT version): The BSDSF employs a two-tier agreement system. At the edge level, a pre-agreement is formed rapidly by edge nodes to reduce time delay and meet immediate data processing needs. The general agreement among full nodes is achieved using a BFT version, securing a safe and extensive network agreement.

Autonomous validators: In BSDSF, smart contracts

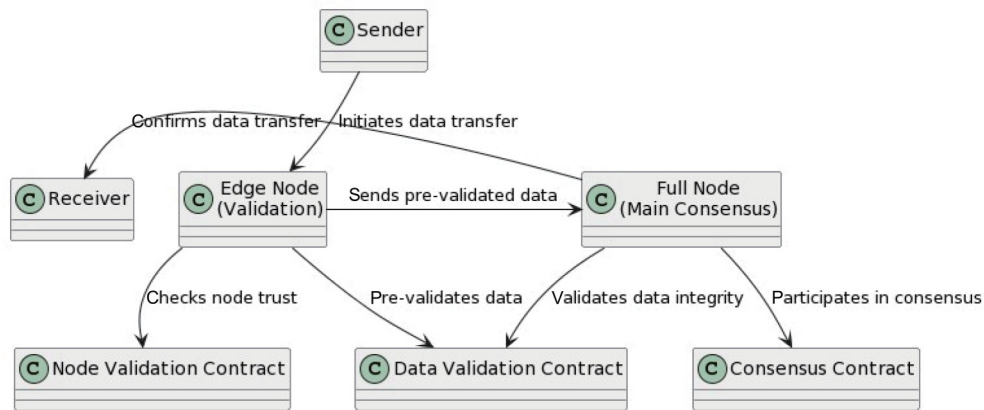


Fig. 1 Components of BSDSF architecture.

work on their own, following set rules for various duties. These involve a validation contract for node validation standards, a data validation contract for safe and genuine data transfer, and a consensus contract for achieving network agreement in a trustless environment.

Transferring data: BSDSF's method for moving data starts simple. The sender node asks a recipient node to accept data. Next, edge processing plays a part to speed things up and lighten network duties. This goes hand-in-hand with a preliminary approval of data correctness by edge nodes, by the rules of the data validation contract. Once it passes, the transaction heads to the full nodes for last review and acceptance. The job is completed when the full nodes agree and the blockchain logs the data, making it available to the recipient.

Safeguarding Info: BSDSF protects data in several ways. Communication lines and user identities are safe due to cryptography. They check smart contracts often to find and fix weak spots. Access is controlled to limit user rights and use of the network.

Failure and attack handling: BSDSF is ready for glitches like node troubles and harmful acts. They watch node activity closely and take--action at signs of an issue. Smart contracts play a role to find and cut off threats to ne-twork safety.

3.2 Pre-consensus at edge level: Detailed design

When these two are brought together at an edge-level pre consensus mechanism, they create a paradigm shift in architecture (see Fig. 2). Since latency cannot be tolerated and the authenticity of data acquired near real time must be verified to maintain network integrity, it's ideal for edge computing. The goal of this pre-

consensus edge-level mechanism is for it to be quick and robust, in particular making use of the location, proximity to data sources, and storage available to trust on the local level. It is designed to eliminate bogus transactions right at the start, and produce valid data for the more rigorous consensus procedure used by all of the network's full nodes. The mathematical model in Algorithm 1 must clarify the steps and procedures in a particularly clear way. The edge-level blockchain pre-consensus mechanism to achieve fast and safe data verification requires careful thought and design. The authentication of senders and recipients and confirmation of data integrity met the need for edge computing to process realtime information. Pick edge nodes as the validators was a previously fixed set of rules. Reliable nodes were identified using trust scores based on past performance and behaviors, and data source proximity was given priority to reduce latency. Nodes with more powerful computing capabilities also preferred the validation process. The system was able to adjust to shifting node performance and network conditions through periodic validator set reevaluation. BFT protocol was altered by this application to employ a round-robin governance scheme for validator nodes. Fair distribution of block proposals led to a decrease in centralization. Voting on a proposed block was conducted by validator nodes in accordance with the rules of the smart contract. These contracts specify business logic, cryptographic standards, and transaction validation. Effectiveness of the system and fault tolerance were balanced by a two-thirds majority consensus threshold. The threshold could be changed to suit the size and security requirements of the network. Validator nodes signed the block to attest to its integrity after it crossed this threshold.

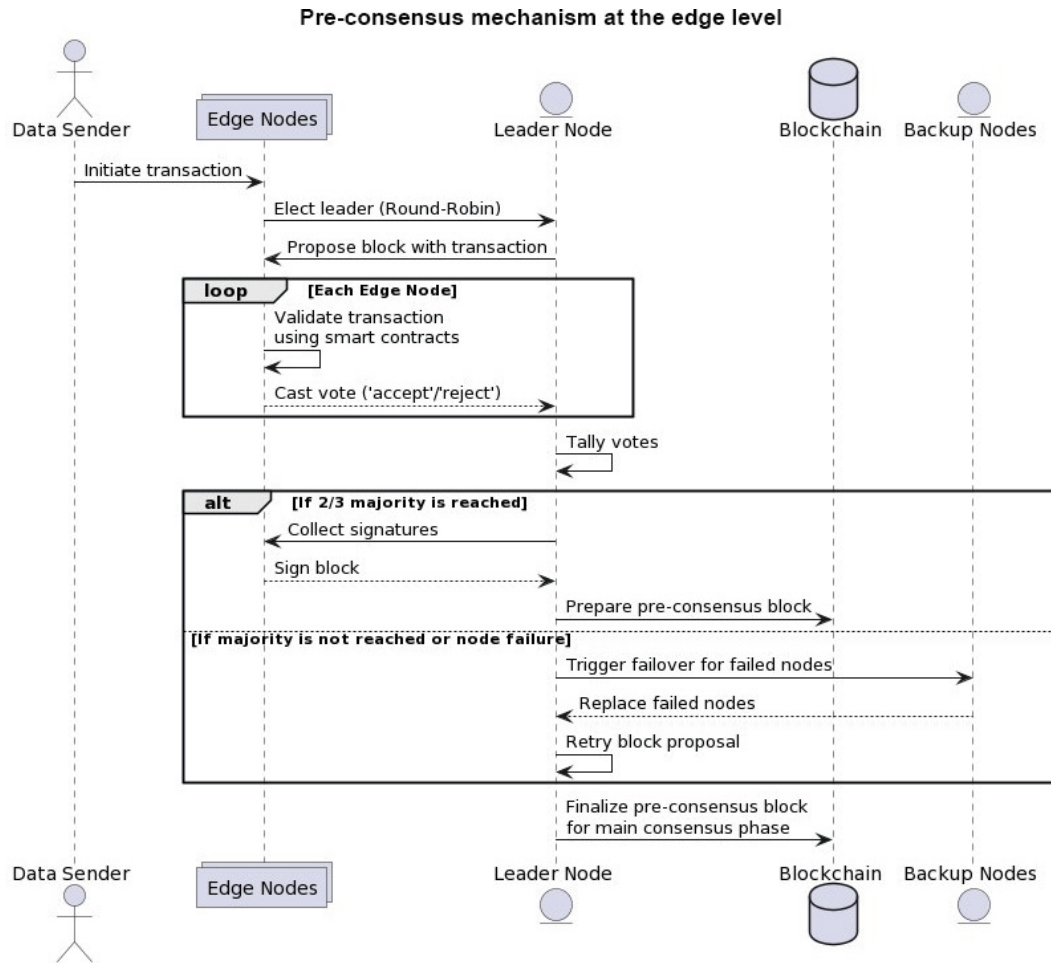


Fig. 2 Sequence flow of the pre-consensus mechanism.

The fault handling of the system was dependable. Nodes sent out regular messages to verify that they were operational. Messages that were missed triggered a failover process in which backup nodes from the selection stage took over without any problems. Anomalous behavior detection systems were put in place to identify and stop it. As validators, malicious nodes were swiftly replaced. The data block marked the end of the pre-consensus process. Validated data transactions in this block were accompanied by metadata and validator signatures. This block was not yet on the blockchain, but it was prepared for the consensus process involving all nodes.

This pre-consensus process was dependent on a number of technical elements. Because smart contracts were designed with edge computing in mind, they were quick and light. Protocols for secure and effective communication were needed for vote broadcasting and maintaining node health checks. The system also includes cryptographic safeguards to improve nodes'

authentication, secure transaction validation, and communication channel security. The pre-consensus mechanism demonstrated a quick and robust system based on proximity to data sources and localized trust, which are critical components of edge computing. In order to prepare data for the rigorous consensus process of the blockchain network, the system filters invalid transactions in advance. Abstracted version of Algorithms 2 and 3 steps follows.

3.3 Main consensus among full nodes: A detailed design

The primary consensus amongst full nodes in a blockchain architecture was meticulously designed and implemented using the PBFT^[9] protocol. Even in the event that a portion of the nodes in the network fail, the multi-phase commit process guarantees data block finality and integrity. Every full node held a complete ledger and was in sync with the network state when the consensus process started. A primary node, selected

Algorithm 1 Pre-consensus at edge level**Input:** Set of transactions T_x **Output:** Pre-consensus block B signed by the majority of validator nodes.**Initialization:**Select N validator nodes based on proximity, trust scores, and computational resources.Assign L from V in a round-robin fashion.**Algorithm steps:**

Step 1: Block Proposal

while L is active **do** L collects transactions T_x and proposes a block B . L broadcasts B to all nodes in V .**end while**

Step 2: Vote Casting

for each validator node $v \in V$ not equal to L **do**Validate B based on predefined smart contract rules.Cast V_vote as 'accept' if B is valid, else 'reject'.Broadcast V_vote to all nodes in V .**end for**

Step 3: Threshold Check

Set $S = \phi$.**for** each received V_vote from V **do**If V_vote is 'accept', add the signature of v to S .If $|S| \geq C$, move to Step 4, else continue collecting votes.**end for**

Step 4: Signature Gathering

if $|S| \geq C$ **then**Aggregate all signatures in S to endorse B .Finalize B with S as the pre-consensus block.**end if**

Step 5: Fault Handling

for $v \in V$ **do**If no H is received from v within a predefined timeout, remove v from V .If v consistently casts 'reject' votes without justification, suspect v as byzantine and remove from V .**end for**

Step 6: Pre-Consensus Block Creation

Create pre-consensus block B with all T_x and S , marking it ready for main consensus.**Termination:** The algorithm terminates when a valid B has been signed and the threshold C has been met, or the set of transactions T_x is empty, $=0$

from among the other nodes, initiated the consensus process. In order to initiate the consensus process, the replicas, or non-primary nodes, awaited signals from the primary nodes. For the purpose of maintaining data consistency and facilitating system recovery, the

network also took snapshots of the system state at regular intervals. Pre-consensus block from the edge level was received by the primary node, which then packaged it into a pre-prepare message and distributed it to all full nodes. To guarantee accurate ordering and fault recovery in the message, unique sequence and view numbers were employed architecture that shown in Fig. 3.

The pre-consensus block was compared to the smart contract rules in prepare by each smart node. Following validation, the network was broadcast with prepared messages to verify the block hash and sequence number. Prepare nodes for consensus with fully stored messages. When every node had a supermajority of matching prepare messages from other nodes, the commit phase started. The node was now prepared to add the block to its ledger and had broadcast a commit message. A block could be added to the ledger when nodes gathered a supermajority of matching commit messages.

After committing the block, nodes moved into the reply phase and confirmed the transaction had been executed by sending confirmation messages to the client. When the block was finalized, the majority of full nodes added it to the blockchain. The secure data transfer to the recipient was completed after the block was permanently added to the ledger and the transactions were executed. In order to maintain and recover the system, checkpoint was necessary. After committing a predetermined number of blocks, the network generated checkpoint edges containing the state of the ledger. A supermajority of nodes were able to expedite recovery, save storage, and discard older log entries by accepting and agreeing to these checkpoint messages. Other nodes could start a view change protocol to choose a new primary node in the event that the current one failed. Additionally, the system could quickly and reliably recover from failures by using supermajority consensus to checkpoint states. Finally, all valid edge data blocks were rigorously committed to the blockchain by the PBFT-based main consensus process. By maintaining system finality and data block integrity even in the presence of byzantine faults, this secured data transfer across the network.

3.4 Node validation contract

The node validation contract in the blockchain framework is designed to automate the process of

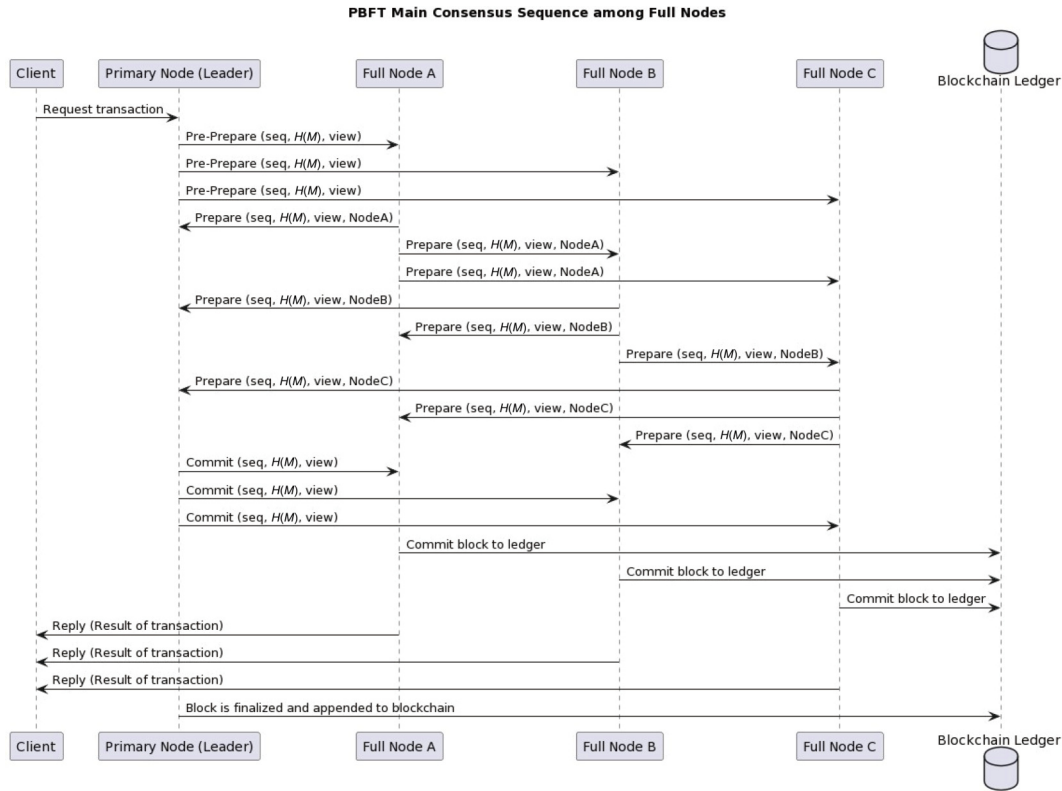


Fig. 3 Sequence flow of the main-consensus mechanism.

ensuring nodes adhere to the network's trust standards. This is done both at the time of node registration and periodically afterwards to maintain ongoing compliance. The contract (Figs. 4 and 5) includes several key functions that handle different aspects of node validation.

Contract structure:

1. Node registration function:

- Purpose: Manages the initial integration of nodes into the network.
- Functionality: Ensures that new nodes meet the network's minimum criteria before they are allowed to participate.

2. Trust score calculation function:

- Purpose: Dynamically assesses the reliability of nodes.
- Functionality: Calculates the trust score of nodes by evaluating their activity, performance, and the results of past validations.

3. Identity verification function:

- Purpose: Confirms the authenticity of nodes.
- Functionality: Verifies the credentials of nodes by cross-referencing them with an on-chain registry or utilizing a decentralized identity verification service.

4. Continuous validation function:

- Purpose: Ensures sustained compliance of nodes with network standards.
- Functionality: Periodically re-evaluates nodes to confirm they continue to meet the required trust and performance standards.

5. Event handlers:

- Purpose: Provide notifications and updates regarding the validation process.
- Functionality: Generates and emits events to signal successful validations, instances of validation failures, or updates in trust scores.

Contract state variables

// Mapping from node addresses to their trust scores.
mapping (address to uint) private trustScores;

// Mapping indicating whether a node is currently considered valid. mapping (address to bool) private isNodeValid;

// The minimum trust score required for a node to be considered valid. uint private minTrustScore;

Edge computing is dynamic because node validation mechanisms influence network integrity and resilience. The BSDSF uses smart contracts and advanced node validation protocols to automate, clarify, and ensure

Algorithm 2 Main consensus algorithm**Inputs:** N = total number of full nodes in the network f = maximum number of faulty nodes, where $N \geq 3f + 1$ M = pre-consensus data block**Node State:** $s(S, \log, \text{view}, \text{seq})$

Pre-Prepare Phase:

Leader node L broadcasts $\langle \text{PRE-PREPARE}, \text{view}, \text{seq}, H(M) \rangle$ to all nodes.

Prepare Phase:

for each node i **do** **if** seq is within range and M is valid **then** Broadcast $\langle \text{PREPARE}, \text{view}, \text{seq}, H(M), i \rangle$. **end if****end for**

Commit Phase:

for each node i **do** **if** receives $2f + 1$ PREPARE messages **then** Broadcast $\langle \text{COMMIT}, \text{view}, \text{seq}, H(M), i \rangle$. **end if****end for**

Reply Phase:

for each node i **do** **if** receives $2f + 1$ COMMIT messages **then** Append M to S and send reply to client. **end if****end for**

Checkpoint Step:

Periodically, broadcast $\langle \text{CHECKPOINT}, H(S), n, i \rangle$.

View Change Step:

if leader failure or suspicion **then** Trigger view change to $v + 1$ with new leader.**end if**

that data cannot be changed. These mechanisms are designed to adapt dynamically to the network's ever-changing topology, in which nodes can join or leave at any time and node conditions can change quickly. BSDSF employs dynamic node scoring and continuous monitoring to ensure that nodes' trustworthiness is determined by their current behavior and contribution to the network, making the system naturally resistant to malicious behavior.

The BSDSF node validation mechanisms respond well to network changes. Smart contracts ensure that evaluation is up to date and impartial. Active nodes update their trust scores in real time. Continuous monitoring and automated revalidation assist in identifying and mitigating risks associated with hacked

Algorithm 3 Trust score management and node validation**Step 1: Initialize variables**Let N be the set of all nodes $N = \{n_1, n_2, \dots, n_k\}$.For each node n_i , initialize a trust score $\text{TS}(n_i)$.Define minTS as the minimum trust score threshold.Define $\text{IDV}(n_i)$ as the identity verification status of node n_i .**Step 2: Define trust score update function**Define $\text{UpdateTrustScore}(n_i, \Delta)$ where Δ is the change in trust score.**Step 3: Define identity verification function**Define $\text{VerifyIdentity}(n_i)$ that returns true if verified.**Step 4: Define validation function**Define $\text{ValidateNode}(n_i)$: If $\text{IDV}(n_i)$ is true and $\text{TS}(n_i) \geq \text{minTS}$, then valid.**Step 5: Node registration**For a new node n_j : Initialize $\text{TS}(n_j)$, set $\text{IDV}(n_j)$, call $\text{ValidateNode}(n_j)$.**Step 6: Periodic trust score evaluation**For each node n_i in N , evaluate performance and interactions.**Step 7: Handling identity changes**For affected node n_i , call $\text{VerifyIdentity}(n_i)$ and update $\text{IDV}(n_i)$.**Step 8: Emit events**

Activate on change in validation status or trust score.

Step 9: Data recording

Record all trust scores and validation statuses in a persistent blockchain.

Step 1: Node validation conclusion

Verify all nodes in the network have current and accurate validation status.

Trust Score update: $\text{TS}(n_i) = \text{TS}(n_i) + \Delta$ (if $\Delta > 0$), else $(\text{TS}(n_i) - \Delta, 0)$.**Node validation:**

$$\text{IsValid}(n_i) = \begin{cases} 1, & \text{if } \text{IDV}(n_i) = \text{true and } \text{TS}(n_i) \geq \text{minTS} = 0; \\ 0, & \text{otherwise} \end{cases}$$

or untrustworthy nodes. The dynamic approach allows BSDSF to maintain network integrity as members change. It ensures that only trustworthy nodes utilize consensus and data sharing mechanisms.

Its consensus mechanisms are also designed to be adaptable based on node scores and network locations, demonstrating the framework's versatility in a number of ways. BSDSF's adaptability allows it to maintain high security and efficiency in the face of network changes. By fostering a secure, resilient, and adaptive data sharing and processing environment, BSDSF addresses edge computing network dynamics and establishes a new standard for network integrity and trust in decentralized computing settings.

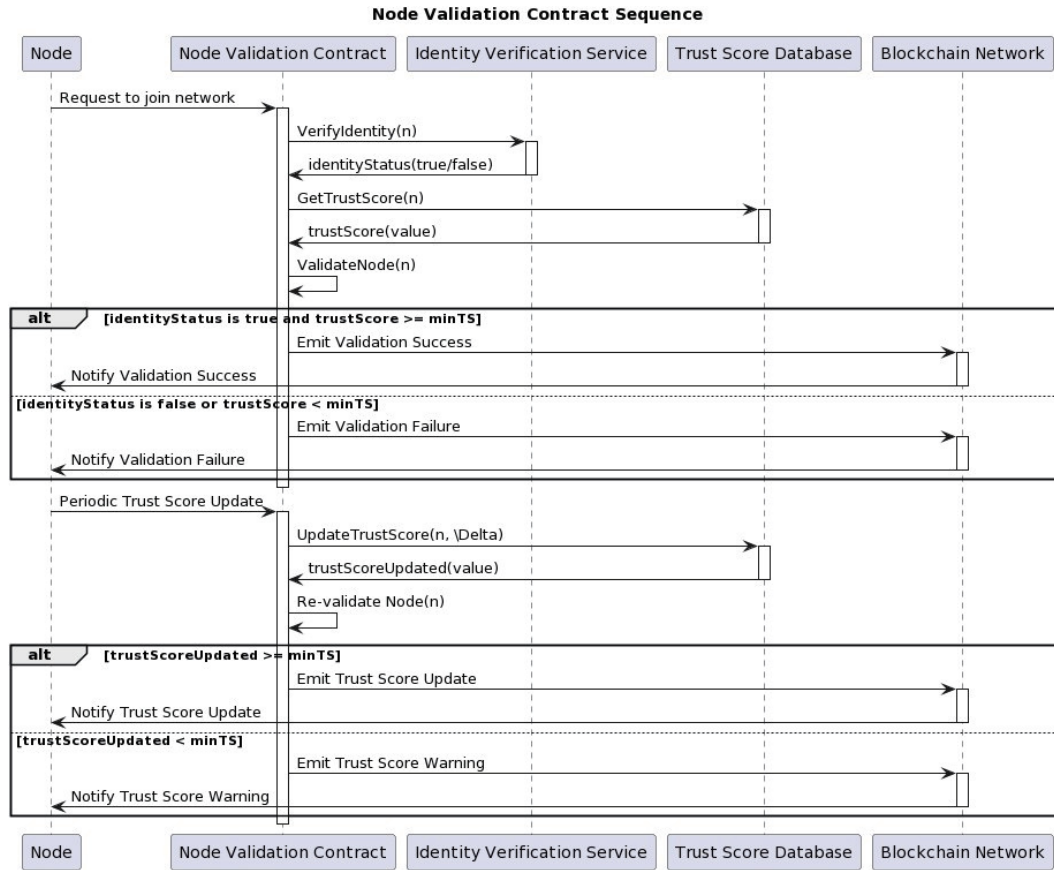


Fig. 4 Sequence flow of the node validation contract.

3.5 Data validation contract

The data validation contract is designed to autonomously ensure the integrity and authenticity of data transactions within the blockchain. This smart contract acts as an automated gatekeeper, meticulously evaluating incoming data against a comprehensive set of predefined rules and criteria, such as format adherence, content accuracy, and signature authenticity, to determine its validity.

At its core, the contract processes structured data inputs, including essential elements like digital signatures and metadata, to conduct thorough checks. These checks encompass format and content validation, digital signature authentication using the sender's public key, and integrity verification through advanced hashing techniques. The contract's logic is encapsulated in functions that execute these validations in a stepwise manner, beginning with data receipt and format checking, followed by content assessment, signature verification, and integrity checks. This process culminates in a final validation outcome, which is then communicated through emitted events, detailing

the validation results for transparency and auditability.

Key to the contract's efficacy are considerations for security and efficiency, such as gas optimization and robust exception handling, ensuring that the contract operates seamlessly and economically. Deployment of the contract follows rigorous testing and auditing, with ongoing maintenance to adapt to new validation requirements and network evolutions, guaranteeing that the contract remains a reliable arbiter of data validity in the dynamic BSDSF ecosystem.

The consensus contract in the BSDSF plays a vital role. It embeds the logic of a BFT variant consensus mechanism, enabling the system to reach agreement on the blockchain's state even amidst faulty or malicious nodes. This contract orchestrates the consensus process among network nodes, ensuring the reliability and integrity of the blockchain system.

The algorithm steps of data validation contract are outlined below:

3.6 Smart contract-based data validation

The integration of data validation innovated by the BSDSF, represents a critical enhancement in ensuring

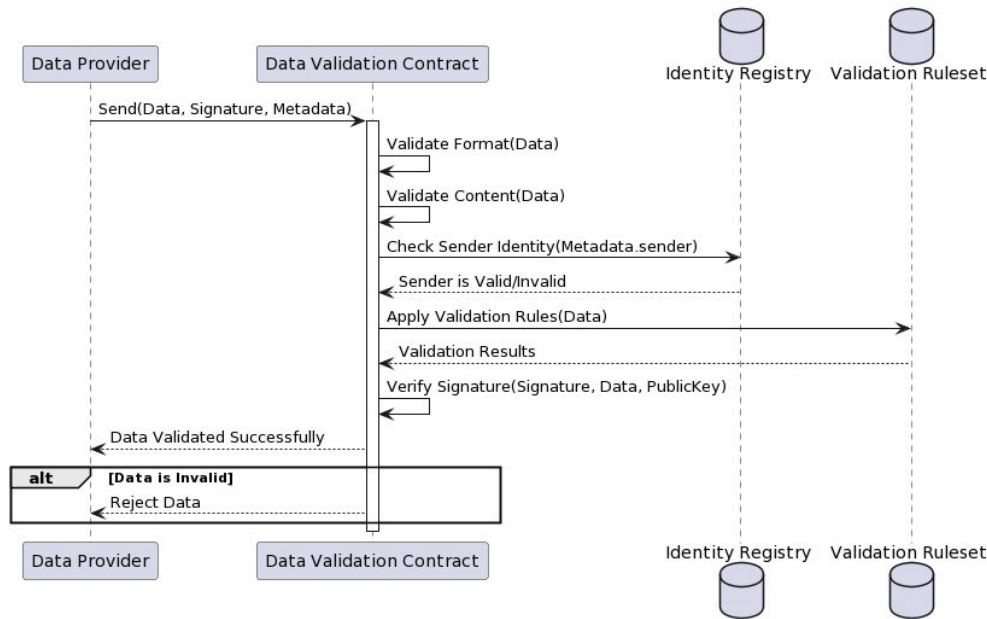


Fig. 5 Sequence flow of the data validation contract.

data integrity and security across decentralized networks. The BSDSF utilizes smart contracts for data validation, which are programmed to handle a diverse array of data types—ranging from simple numerical values and text to more complex structures such as multimedia files, documents, and real-time data from IoT devices.

Handling varying data types: The smart contracts within BSDSF are designed with flexibility to accommodate the specific characteristics and validation requirements of different data types. For instance:

- **Textual and numerical data:** Validation rules may include format checks, completeness verification, and consistency assessments against predefined patterns or schemas.

- **Multimedia files:** Beyond basic format validation, smart contracts may employ algorithms to verify file integrity, ensuring that the content has not been tampered with or corrupted.

- **Real-time IoT data:** Considering the dynamic and voluminous nature of IoT data, smart contracts implement temporal validation to ensure timeliness and relevance of data, alongside anomaly detection algorithms to identify deviations from expected patterns.

Complex validation scenarios: BSDSF's data validation process Algorithm 4 shines in its ability to manage complex scenarios, such as cross-source data validation and conditional validation processes. This

capability is particularly important in edge computing environments where data may originate from heterogeneous sources and under varying conditions.

- **Cross-source validation:** Smart contracts can cross-reference data from multiple sources to validate accuracy and authenticity, essential for applications requiring data aggregation from diverse inputs.

- **Conditional validation:** The validation logic can adapt to specific conditions or external inputs, allowing for dynamic adjustment of validation criteria based on real-time data, network status, or specific application requirements.

By integrating these advanced data validation techniques within the blockchain framework, BSDSF ensures not only the integrity and security of data shared across the network but also enhances the adaptability and robustness of the system to meet the diverse needs of edge computing applications.

4 Experimental Study

The simulation model for the BSDSF integrated the capabilities of iFogSim^[20] and SIMBLOCK^[21] to replicate the framework's performance and activities under varied conditions. In this two-layered simulation approach, iFogSim was employed to represent the edge computing aspects, while SIMBLOCK was tasked with simulating the blockchain operations inherent to the BSDSF.

Within this model, SIMBLOCK was specifically

Algorithm 4 Data validation process**Let:** D = data input to the contract. σ = digital signature associated with D . M = metadata, including timestamp t and sender identifier s . P_s = public key of the sender. V = set of validation rules. H = hash function. R = registry of approved identities.

Data reception:

Receive(D, σ, M)Validate D and σ are not null and M contains t and s .

Format validation:

for each rule $v_f \in V$ concerning format **do** Check(D, v_f) **if** D fails v_f **then** Reject(D). **end if****end for**

Content validation:

for each rule $v_c \in V$ concerning content **do** Check(D, v_c) **if** D fails v_c **then** Reject(D). **end if****end for**

Signature verification:

if VerifySignature(σ, D, P_s) returns false **then** Reject(D).**end if**

Authenticity check:

if $s \notin R$ **then** Reject(D).**end if**

Integrity check:

Compute $h = H(D)$.**if** h does not match the hash within M **then** Reject(D).**end if**

Finalize validation:

if all above steps are passed **then** Approve(D).**end if****Definitions:**Check(D, v): Applies the rule v to D , returns true if D passes.Reject(D): Terminates validation, indicates D is invalid.Approve(D): Completes validation, indicates D is valid.VerifySignature(σ, D, P_s): Verifies σ using P_s against $D = 0$

configured to capture the complexities of a blockchain network. This included simulating the detailed aspects of node communication, the propagation of transactions throughout the network, and the nuanced process of block execution as per the BFT consensus algorithm. To ensure a realistic representation of blockchain operations, various parameters such as network topology, block size, and the intervals between blocks were meticulously adjusted to reflect actual operating conditions.

Simultaneously, iFogSim focused on modeling the edge computing element of the BSDSF. It simulated the distribution and execution of computational tasks across a network, taking into consideration the critical aspects of latency, resource allocation, and the inherent constraints of edge devices. The objective was to closely emulate the operations of edge computing within the BSDSF, acknowledging the unique challenges and limitations posed by edge computing environments.

To ensure a cohesive simulation experience, an interface was established between iFogSim and SIMBLOCK, maintaining a seamless exchange of data and events. This interface was pivotal in ensuring that the blockchain operations simulated by SIMBLOCK directly influenced the edge computing operations portrayed by iFogSim. The interplay between these two simulations allowed for a comprehensive examination of the BSDSF's performance, capturing the intricate relationship between edge computing and blockchain operations within the framework. This holistic approach provided valuable insights into the BSDSF's functionality, offering a detailed and nuanced understanding of its behavior under a variety of simulated conditions.

Table 1 outlines the simulation parameters and respective values used for the simulation of the BSDSF, providing a comprehensive view of the setup designed to assess the framework's performance in a simulated environment accurately:

4.1 Deployment environment

This study requires the cloud-based execution environment Google Colab^[22] for a number of reasons.

For large-scale simulations, Colab provides strong computational resources without requiring local high-performance computing infrastructure. Collaboratively, researchers can modify simulation parameters and code in real time on the platform and exchange insights.

Table 1 System parameters and their optimal values.

Parameter	Optimal value
Number of nodes	50–200 nodes
Transaction rate	25–100 transactions per millisecond
Block size	1 MB
Block interval	30 s
Latency	100 ms
Bandwidth	1 Gbit/s
Consensus algorithm	PBFT
Node failure rate	0.1% per hour
Data validation time	5 milliseconds
Network topology	Decentralized
Smart contract size	20 kb
System uptime	99.90%

Colab's simulation environment can be scaled with ease because of the strong backend infrastructure of Google's servers.

Integration: To streamline workflow, Colab integrates with GitHub and Google Drive for version control and data storage. Not requiring any setup The cost and time savings of running simulations without setup are substantial.

4.2 Simulation parameters

To represent the dual nature of the simulation, several parameters were set, including the number of nodes. An emulated deployment scale network with between 50 and 200 nodes.

Transaction rate: To test system throughput under load, adjust 10–100 transactions per second.

1 MB and 10–60 s blocks have been used to measure block propagation times and chain growth.

To simulate communication delays and transfer rates, which have an impact on consensus and data sharing, latency and bandwidth were configured.

4.3 Security analysis

The BSDSF security analysis assesses the blockchain infrastructure, consensus protocol, data contracts, and data transfer procedure for their ability to withstand different security threats and attacks.

Blockchain infrastructure:

- **Immutability and Tamper-Resistance:** By limiting changes made in the past, blockchain ensures data integrity. By distributing data among a network of nodes, BSDSF lowers the risk of targeted attacks and centralized failure incidents.

- **Privacy:** Despite blockchain's transparency,

BSDSF can use encryption or zero-knowledge proofs to secure sensitive data.

BFT variant consensus protocol:

- The BFT consensus protocol is designed to handle up to one-third of malfunctioning or malicious nodes, ensuring network consensus.

- **Sybil attack resistance:** When a malicious entity creates numerous false identities, identity and trust score evaluations shield nodes from Sybil attacks.

Smart contracts:

- The risk of human error or malicious tampering is decreased by smart contracts' automatic enforcement of predefined rules.

- Regular audits and the capability to update smart contracts in response to vulnerabilities found are two important aspects of system security maintenance.

- Smart contracts provide access control over the permissions and actions of network participants, thereby averting unwanted access.

Data transfer protocol:

- During transmission, data is encrypted, ensuring privacy and guarding against interception.

- **Edge-level pre-consensus:** Rapid edge validation guards against malicious data entering the core network.

Failure and attack handling:

- **Node failure detection:** To lower the risk of node failure, the system keeps an eye on nodes and reacts automatically to any abnormalities in operation.

- Smart contracts allow the network to isolate and eliminate malicious nodes.

Potential vulnerabilities and countermeasures:

- Although the BFT mechanism is robust, it could be compromised by a majority takeover of the network. Control of this kind is challenging due to the validation procedures needed for node participation. It is possible to take advantage of weaknesses in the code of smart contracts. By conducting extensive testing and auditing, this is lessened.

- **Endpoint security:** Despite the network's security, user devices may still be at risk. User endpoint security must be robust.

- **Network attacks:** DDoS and DoS are particularly dangerous. By putting in place techniques like rate limitation and network partitioning, BSDSF can counteract these threats.

- BSDSF offers robust security for data sharing in networks utilizing edge computing. BSDSF's

architecture is designed to anticipate and counteract a wide range of potential security, data attacks protection, system availability, and trustworthiness threats. Retaining the integrity and robustness of the framework necessitates proactive security measures, frequent updates, and ongoing monitoring.

4.4 Performance analysis

4.4.1 Transaction throughput

By analyzing the graphs in Fig. 6 generated from the transaction throughput models data over four iterations, the performance and scalability of the BSDSF are compared with those of BSDSS^[10] and DSSS^[19].

Iteration 1 (50 Nodes, 25 Transactions/ms): BSDSF does well with small network loads. It starts strong and keeps up a high throughput. At 240 s, BSDSF throughput hits its peak. This shows it's great at handling transactions. BSDSS and DSSS follow the same path but at a lower pace, implying constraints or subpar transaction processing.

Iteration 2 (100 Nodes, 50 Transactions/ms): BSDSF reaches a high throughput of 100 tx/s twice, at 120 and 180 s. This shows its impressive scalability. It also beats both BSDSS and DSSS in performance. There's a higher fluctuation in throughput in all models, more likely a result of network load. BSDSF continues to lead, showing its robustness under tougher conditions.

Iteration 3 (150 Nodes, 75 Transactions/ms):

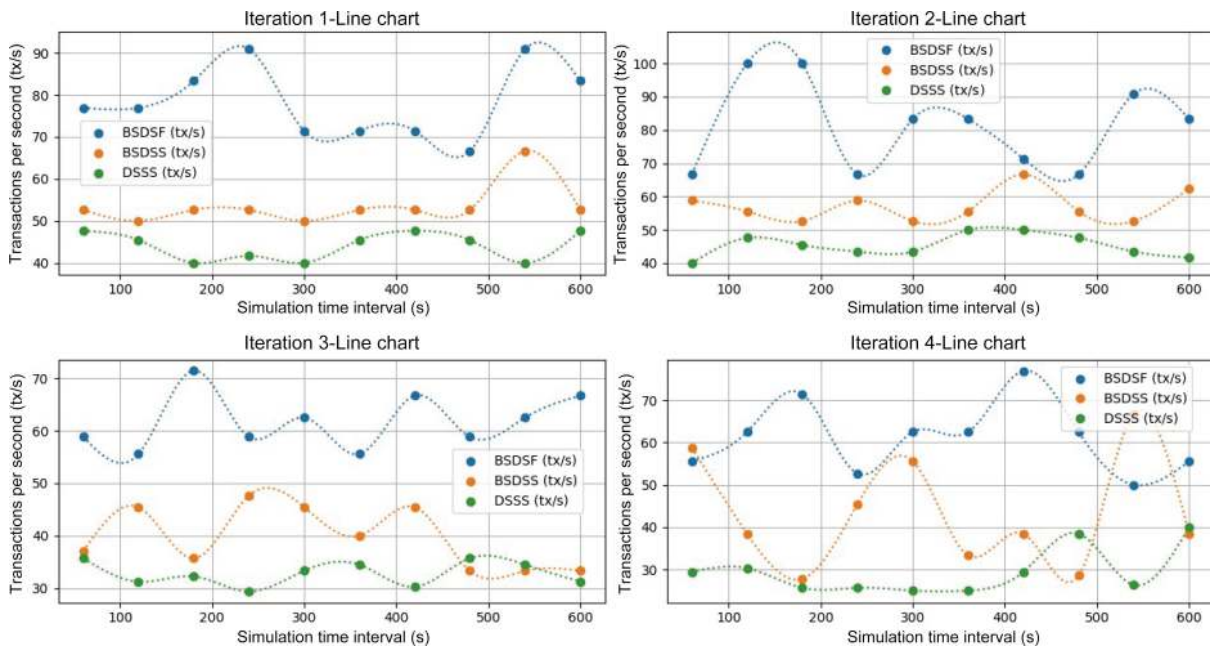


Fig. 6 Transaction throughput observed from BSDSF, BSDSS, and DSSS.

BSDSF gets better with adaptive features managing higher loads. This happens at 180 seconds. BSDSS and DSSS lag behind because of their higher node count and transaction rate.

Iteration 4 (200 Nodes, 100 Transactions/ms):

Even when the transaction rate tops in Round 4, BSDSF stays ahead. It handles changing network complexities well. In contrast, other models show a decline in transaction performance, especially DSSS. This implies problems with managing large number of transactions.

Overall observations: When it comes to a blockchain simulation, BSDSF stands out in transaction processing. It consistently surpasses the other two models in terms of transaction throughput. Though BSDSS performs better than DSSS, it still trails BSDSF, especially when transaction speeds rise. With the lowest throughput, DSSS may not be suited for high-load environments with many nodes and high transaction rates. Performance varies for all models leading us to think more research might be needed in transaction handling and enhancements. The graph clearly shows that BSDSF is the ideal framework to handle variable loads and node counts without compromising on transaction throughput in systems that need high transaction processing.

4.4.2 Transaction latency:

The performance of the Blockchain-Driven Secure Data Sharing Framework (BSDSF) in comparison to

BSDSS and DSSS can be understood by comparing transaction latency graphs in Fig. 7 over the course of four iterations.

Iteration 1 (50 Nodes, 25 Transactions/ms): Indicating superior processing, BSDSF shows the lowest latency across all time intervals. The latency of DSSS is consistently higher than that of BSDSS and DSSS, indicating that it might be the least efficient model. Between 240 and 540 s, BSDSF latency drastically decreases, indicating peak performance.

Iteration 2 (100 Nodes, 50 Transactions/ms): Once more, BSDSF performs better than the others at 120- and 180-second intervals and has lower latency. DSSS shows better performance than BSDSS, but BSDSF lags behind, particularly in the middle of the simulation. Indicating scalability problems, DSSS latency peaks at 60 s and shows a trend toward higher latency.

Iteration 3 (150 Nodes, 75 Transactions/ms): Although iteration 2's latency is marginally higher than it is in BSDSF, it is still the lowest and most consistent. There are latency spikes in BSDSS, particularly at 180 seconds, which could be a sign of ineffective transaction volume control. With a peak latency of 240 seconds, DSSS exhibits the highest latency, underscoring scalability concerns.

Iteration 4 (200 Nodes, 100 Transactions/ms): Even though network load has caused a slight increase in readings, BSDSF has the lowest latency. While

BSDSS peaks at 180 s, DSSS reaches its maximum latency at 300 and 360 s. Although DSSS shows improvement toward the end of the simulation, it still has issues with the highest node count and transaction rate.

Overall observations: We find from our examinations that the framework reliably boasts the smallest amount of transaction delay in all cases. It seems to handle rising demands and complexity effectively. On the other side, BSDSS's performance is inconsistent when juxtaposed with DSSS and fails to measure up in scalability. For DSSS, as number of nodes and transaction rates increased, latency takes a hit - not the best fit for high-volume situations. Although more nodes and quickened transaction rates hike up latency for all models, the framework sways us with its lower latency - its consensus method appears more trustworthy and efficient.

4.4.3 Block propagation time

By comparing the block propagation time graphs of four different models, it is possible to gain insight into the efficiency of the Blockchain-Driven Secure Data Sharing Framework (BSDSF) in comparison to two other models, BSDSS and DSSS.

Iteration 1 (50 Nodes, 25 Transactions/ms): Under light network loads, BSDSF has the quickest block propagation time, suggesting that it is the best block dissemination protocol. BSDSS and DSSS both have longer propagation times, with DSSS having the

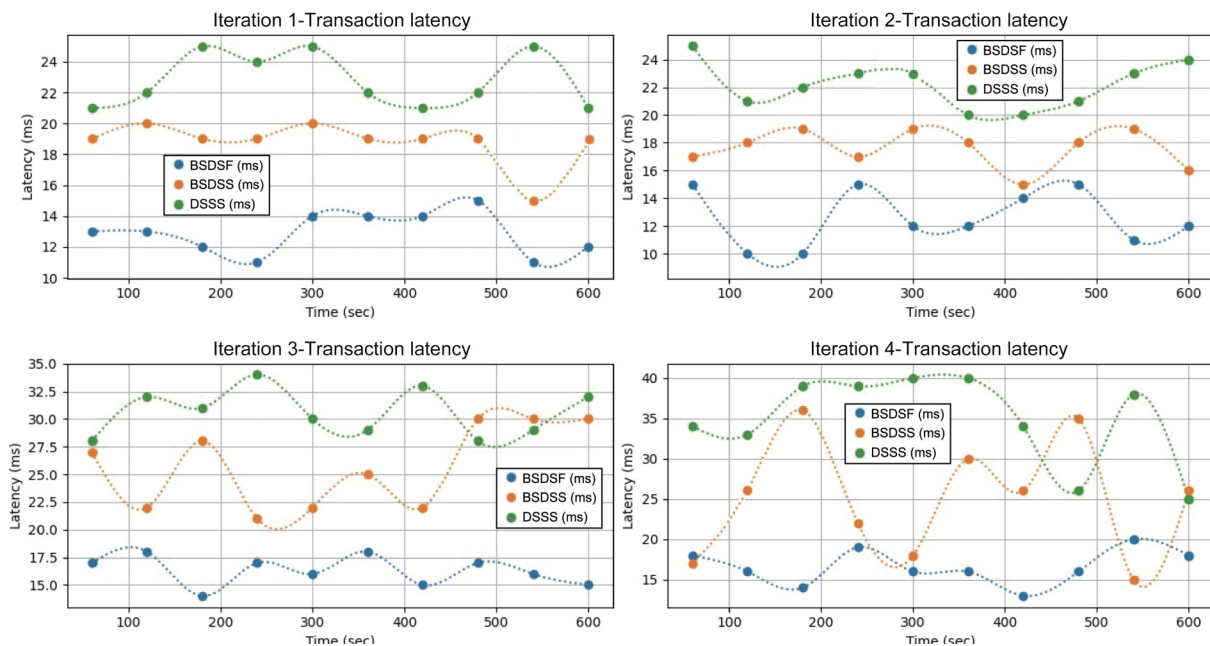


Fig. 7 Transaction latency observed from BSDSF, BSDSS, and DSSS.

longest, indicating possible inefficiencies in their block propagation mechanisms.

Iteration 2 (100 Nodes, 50 Transactions/ms): Although BSDSF once more shows the quickest propagation times, there is a discernible increase in time, most likely as a result of the doubled network load. In terms of performance, it still works better than other models. Both BSDSS and DSSS lengthen the propagation time, but DSSS occasionally outperforms BSDSS, suggesting problems with scalability when under load.

Iteration 3 (150 Nodes, 75 Transactions/ms): BSDSF has the quickest propagation time despite having more nodes and a higher transaction rate. It shows that efficiency can be preserved under extreme pressure. With peaks at 540 and 600 s, BSDSS shows increased variation in propagation time. The DSSS has the longest propagation times, 180 and 240 s, which confirms the scalability problems of earlier iterations.

Iteration 4 (200 Nodes, 100 Transactions/ms): Even at the highest simulation load, BSDSF's performance is comparable to that of BSDSS, but it still has the advantage in propagation efficiency. All models have longer transaction times, indicating that managing more nodes and transactions is more difficult. In order to maintain a shorter time, BSDSF shows resilience. The longest propagation times, peaking at 480 s for DSSS, may put a limit on

applications involving real-time block confirmation.

General observations: All iterations show that BSDSF consistently has the lowest block propagation times, indicating a scalable and efficient algorithm. BSDSS's efficiency falls short of BSDSF's when network conditions deteriorate. In all scenarios, DSSS shows the highest latency, suggesting that it may have trouble with block propagation in larger or busier networks.

To sum up, BSDSF appears to be the most reliable block propagation method under the tested conditions, maintaining shorter propagation times and possibly enhancing throughput and transaction confirmation rates. This holds significance for block chain networks, where the integrity and performance of the network depend on the propagation of information.

4.4.4 Smart contract execution time

The performance comparison between BSDSS and DSSS and the BSDSF is presented based on the comparison of smart contract execution times over four iterations Fig. 8.

Iteration 1 (50 Nodes, arrival rate: 25 Transactions/ms): The smart contract execution times of BSDSF are the lowest, indicating effective smart contract processing in a smaller network. The execution times of BSDSS are comparatively faster, peaking at 60 s. The longest execution times are consistently observed with DSSS, suggesting that it is

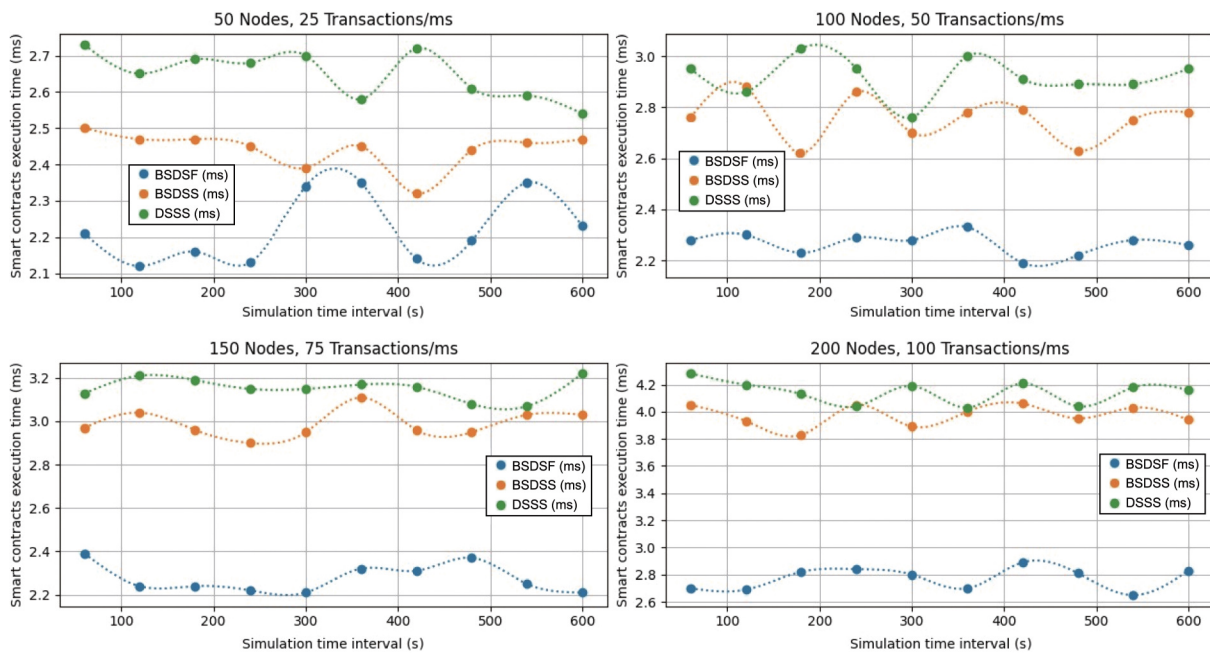


Fig. 8 Block propagation time observed from BSDSF, BSDSS, and DSSS.

less well-suited for smart contracts.

Iteration 2 (100 Nodes, arrival rate: 50 Transactions/ms): As the size of the network doubles, BSDSF leads with the fastest execution times, demonstrating strong scalability. The execution times of BSDSS and DSSS are increased, with DSSS having the longest durations, approximately 180 seconds.

Iteration 3 (150 Nodes, arrival rate: 75 Transactions/ms): All models experience longer execution times due to the increased number of nodes and transactions, but BSDSF keeps performing better. BSDSS increases more, particularly at 60 s, suggesting problems with performance under heavier loads. The performance of smart contract applications may be impacted by the fact that DSSS still has the longest execution times, peaking at the beginning of the simulation.

Iteration 4 (200 Nodes, arrival rate: 100 Transactions/ms): In this test, we checked 200 nodes processing 100 transactions every millisecond. The framework handled the high traffic well. Despite more processing time, it still outperformed other models. Significant data came when the DSSS took more than 4 milliseconds. Both the BSDSS and DSSS took more time, indicating a possible efficiency and scalability problem.

Overall findings: The proposed framework did well in running smart contracts in all tests, showing it's right for this task. BSDSS, the runner-up, showed some struggle when the network intensified. Given its long processing times, DSSS isn't ideal for tasks that need quick reflections on contracts.

Regardless of network traffic intensity, the proposed framework processed smart contracts fastest. That's crucial, as performance influences network speed and user experience. The increasing processing times of BSDSS and DSSS under heavier loads suggest that these models might need more work to perform better in high-traffic situations.

4.4.5 Consensus completion time

When analyze how long it takes to complete network tasks in the four trials, it's clear that the BSDSF is effective. It does a good job reaching agreement compared to BSDSS and DSSS. This is true even when the network sizes and transaction speeds change. See Fig. 9.

Iteration 1 (50 Nodes, 25 Actions/ms): Simply put, BSDSF consistently gives us the quickest network times. This means it's a good choice for regardless of

the network size. BSDSS operates a bit slower, hitting its slowest time of 120 seconds. It seems to have some processing delay. At 60 s, DSSS is the slowest, hinting it may not work as smoothly.

Iteration 2 (100 Nodes, 50 Transactions/ms): BSDSF's larger network size increases consensus times in comparison to Iteration 1, but it still has the shortest times, demonstrating its scalability. Peak consensus times for BSDSS vary, with notable increases at 240 and 480 s, suggesting problems with higher loads. With noticeable peak, suggesting scaling problems, DSSS once more takes the longest.

Iteration 3 (150 Nodes, 75 Transactions/ms): Even with increased network load, BSDSF's consensus times—which peak at 300 s—remain competitive. Both BSDSS and DSSS show a general upward trend in transaction times, with DSSS consistently lagging, indicating potential inefficiencies in processing higher transaction volumes and more nodes.

Iteration 4 (200 Nodes, 100 Transactions/ms): When the network gets bigger, all models take more time. Still, BSDSF works best. It seems to handle extra load quite well. Although it's not as varied as DSSS, BSDSF is slower, especially at 60 s. With times going above 3.3 milliseconds in this round, DSSS is the slowest. This might not be good for widespread use.

Overall findings: BSDSF is quickest for all sizes and actions. Next is BSDSS, although it falls short compared to BSDSF's efficiency, especially when things get bigger. DSSS is always the slowest, raising concerns about its ability to handle growth and high demand.

BSDSF's strong showing in finishing network times spotlights its possible use as a steady system for blockchain networks. These networks need quick agreement, especially as they get bigger. The speed of reaching agreement is super important for systems that rely on prompt data checking and agreement for safety and working properly.

4.5 Comparative analysis

The BSDSF is an evidence to edge computing's development, designed to improve security, latency, and network efficiency using blockchain technology. BSDSF's two-tiered consensus protocol is designed to meet edge computing's immediate data processing needs, reducing latency. Smart contracts automate node validation and data verification, speeding up operations and transactions. The framework's advanced BFT

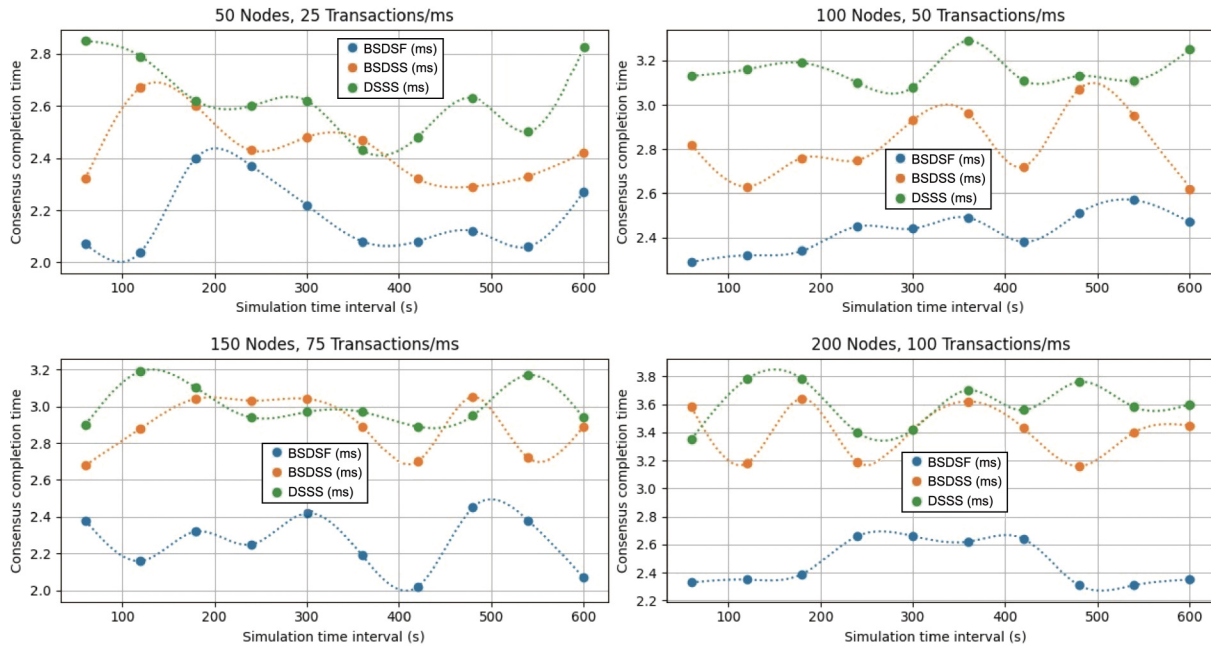


Fig. 9 Consensus completion time observed from BSDSF, BSDSS, and DSSS.

mechanism protects data and systems from a variety of security threats. BSDSF reduces bandwidth consumption and speeds data transaction rates by processing data closer to its source, meeting modern edge computing applications' massive data needs. The framework's advanced encryption protocols protect data during transmission, boosting data integrity. BSDSF is scalable and flexible enough to handle the growing ecosystem of edge devices, making it a versatile and robust edge computing solution. BSDSF has redefined digital trust and efficiency in edge computing with these strategic enhancements, making it an indispensable technological asset.

On the other side, the BSDSS loves to adapt. It uses Deep Reinforcement Learning (DRL) and the A3C formula for smart learning. BSDSS plans the sharing of data and resources, involving MEC and D2D communications. It takes user privacy very seriously.

BSDSS thrives on being scalable and efficient, courtesy of its learning algorithms. It is a network of brains sharing knowledge and data.

Simply put, the DSSS recommended in the IoT network design plan offers the needed layers. From device to device message exchanges at the network's edge to cloud storage and processing, everything is covered. It's possible due to using blockchain for data location pointers, not every transaction. By doing this, they dodge storage limitations and keep blockchain's safety assurances.

DSSS makes a strong network ideal for the needs of IoT devices in these settings. Each of the three proposed solutions for secure data distribution in networks - BSDSF, BSDSS, and DSSS - has a unique method. BSDSF's goal is to create more structured and timely sharing via consensus on an automated script or smart contract. BSDSS, on the other hand, takes a learning and adaptation approach to manage data validations. These methods tackle secure data distributing in scattered computing.

The BSDSF has significantly improved the user experience by incorporating node and data validation contracts, which has boosted transaction speed and reliability. By automating the verification process with smart contracts, BSDSF has sped up transactions and reduced delays, making it easier for users to interact with the network. This automation has increased the reliability of transactions by ensuring data integrity and reducing the likelihood of mistakes or security breaches. As a result, users have a dependable and efficient system. The advanced technology behind blockchain translates into tangible benefits for users, such as faster processing times and increased confidence in the system's capabilities. This development highlights BSDSF's role in bridging the gap between sophisticated blockchain technologies and their practical application, ensuring that users reap the full benefits of secure and effective data sharing in decentralized environments.

5 Conclusion

The Blockchain-Driven Secure Data Sharing Framework (BSDSF) accelerates edge computing. This solution focuses on the most pressing trust, data sharing, and security issues. BSDSF addresses secure data exchange issues by combining edge computing speed, smart contract efficiency, and blockchain's decentralized ledger reliability. Byzantine Fault Tolerance mechanisms and a dual-tier consensus protocol ensure quick and secure transaction execution. Smart contracts speed up data sharing, node identity verification, and consensus rule compliance, reducing errors and increasing security. BSDSF significantly improves the end-user experience by defaulting to faster and more secure data transactions. Self-driving cars and smart infrastructure require real-time processing, so increasing reliability and efficiency is critical. IT also enhances daily user interactions. BSDSF also incorporates the most recent blockchain and edge computing research into its design and functionality. This integration of cutting-edge research distinguishes BSDSF as a novel solution and significant contribution to secure and effective data sharing technologies. BSDSF aims to improve the technical framework of edge computing as well as many other fields by providing a secure data sharing platform. To adapt to the ever-changing digital landscape, it will redefine data integrity, confidentiality, and secure information exchange standards. BSDSF's broad applicability to future data-oriented networks marks a significant step toward more secure, responsive, and user-friendly computing environments.

Acknowledgement

This work was supported by the Deanship of Scientific Research at King Khalid University (No. RGP2/230/44).

References

- [1] R. Yang, F. R. Yu, P. Si, Z. Yang, and Y. Zhang, Integrated blockchain and edge computing systems: A survey, some research issues and challenges, *IEEE Commun. Surv. Tutorials*, vol. 21, no. 2, pp. 1508–1532, 2019.
- [2] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, Data security and privacy-preserving in edge computing paradigm: Survey and open issues, *IEEE Access*, vol. 6, pp. 18209–18237, 2018.
- [3] U. Bodkhe, D. Mehta, S. Tanwar, P. Bhattacharya, P. K. Singh, and W.-C. Hong, A survey on decentralized consensus mechanisms for cyber physical systems, *IEEE Access*, vol. 8, pp. 54371–54401, 2020.
- [4] H. Garcia Molina, F. Pittelli, and S. Davidson, Applications of Byzantine agreement in database systems, *ACM Trans. Database Syst.*, vol. 11, no. 1, pp. 27–47, 1986.
- [5] J. Pan and J. McElhannon, Future edge cloud and edge computing for Internet of Things applications, *IEEE Internet Things J.*, vol. 5, no. 1, pp. 439–449, 2018.
- [6] Y. Yuan and F.-Y. Wang, Blockchain and cryptocurrencies: Model, techniques, and applications, *IEEE Trans. Syst. Man Cybern. Syst.*, vol. 48, no. 9, pp. 1421–1428, 2018.
- [7] W. Yu, F. Liang, X. He, W. G. Hatcher, C. Lu, J. Lin, and X. Yang, A survey on the edge computing for the Internet of Things, *IEEE Access*, vol. 6, pp. 6900–6919, 2018.
- [8] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, Data security and privacy-preserving in edge computing paradigm: Survey and open issues, *IEEE Access*, vol. 6, pp. 18209–18237, 2018.
- [9] M. Castro and B. Liskov, Practical byzantine fault tolerance, *OsDI*, vol. 99, no. 1999, pp. 173–186, 1999.
- [10] L. Liu, J. Feng, Q. Pei, C. Chen, Y. Ming, B. Shang, and M. Dong, Blockchain-enabled secure data sharing scheme in mobile-edge computing: An asynchronous advantage actor-critic learning approach, *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2342–2353, 2021.
- [11] M. Firdaus and K.-H. Rhee, On blockchain-enhanced secure data storage and sharing in vehicular edge computing networks, *Appl. Sci.*, vol. 11, no. 1, p. 414, 2021.
- [12] A. Karati and S. K. Das, Federated secure data sharing by edge-cloud computing model, in *Proc. GLOBECOM 2022 - 2022 IEEE Global Communications Conf.*, Rio de Janeiro, Brazil, 2022, pp. 1362–1367.
- [13] J. Kang, R. Yu, X. Huang, M. Wu, S. Maharjan, S. Xie, and Y. Zhang, Blockchain for secure and efficient data sharing in vehicular edge computing and networks, *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4660–4670, 2018.
- [14] M. Gawas, H. Patil, and S. S. Govekar, An integrative approach for secure data sharing in vehicular edge computing using Blockchain, *Peer-to-Peer Netw. Appl.*, vol. 14, no. 5, pp. 2840–2857, 2021.
- [15] M. M. H. Onik, A. A. H. Tanna, M. A. Emon, M. A. A. Rifat, and M. H. Miraz, Privacy-preserved secure medical data sharing using hierarchical blockchain in edge computing, *Annals of Emerging Technologies in Computing*, no. AETiC, p. 6(4), 2022.
- [16] J. Pan, J. Cui, L. Wei, Y. Xu, and H. Zhong, Secure data sharing scheme for VANETs based on edge computing, *EURASIP J. Wirel. Commun. Netw.*, vol. 2019, no. 1, p. 169, 2019.
- [17] K. Karthikeyan and P. Madhavan, Building a trust model for secure data sharing (TM-SDS) in edge computing using HMAC techniques, *Comput. Mater. Continua*, vol. 71, no. 3, pp. 4183–4197, 2022.
- [18] L. Zhang, H. Zhang, C. Liu, and K. Liu, A secure data request and sharing model based on consortium blockchain in vehicular edge computing environment, in

Proc. IEEE Int. Symp. on Product Compliance Engineering-Asia (ISPCE-CN), Chongqing, China, 2020, pp. 1–6.

- [19] L. Zhang, M. Peng, W. Wang, Z. Jin, Y. Su, and H. Chen, Secure and efficient data storage and sharing scheme for blockchain-based mobile-edge computing, *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 10, p. e4315, 2021.
- [20] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, iFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, Edge and Fog computing environments, *Softw. Pract. Exp.*, vol. 47, no. 9, pp. 1275–1296, 2017.
- [21] Y. Aoki, K. Otsuki, T. Kaneko, R. Banno, and K. Shudo, SimBlock: A blockchain network simulator, in *Proc. IEEE INFOCOM 2019 - IEEE Conf. Computer Communications Workshops (INFOCOM WKSHPS)*, Paris, France, 2019, pp. 325–329.
- [22] E. Bisong, *Building Machine Learning and Deep Learning Models on Google Cloud Platform: A Comprehensive Guide for Beginners*. Berkeley, CA, USA: Apress, 2019.



Fuad A. M. Al-Yarimi was an associate professor of computer science at King Khalid University, Saudi Arabia. He obtained the PhD degree from Jawaharlal Nehru University, India, in 2014. He has published numerous articles in peer-reviewed journals and conferences, focusing on data mining, machine learning,

artificial intelligence, information privacy and security, and cloud computing. His research has developed innovative techniques in these fields, contributing to the advancement of intelligent systems, data protection, and cloud-based technologies.



Ramzi Salah received the PhD degree from Universiti Kebangsaan Malaysia (UKM), Malaysia. He currently works at University Kebangassan Malaysia. His research interests include entity recognition and speech recognition.



Khaled Mohamoud obtained the PhD degree in computer science from IIC University of Technology, Cambodia, in 2018 and the MS degree from International Islamic University Islamabad, Pakistan. His research interests include artificial intelligence, machine learning, deep learning, data science, and wireless sensor

networks.