SyncChain — A Blockchain-Based Supply Chain Management Simulation

Angela Varghese
Student, Dept. of ISE
R V College of Engineering
Bengaluru, Karnataka, India
angelavarghese.is23@rvce.edu.in

Arshia Sirohi
Student, Dept. of ISE
R V College of Engineering
Bengaluru, Karnataka, India
arshiasirohi.is23@rvce.edu.in

Arya Shetty
Student, Dept. of ISE
R V College of Engineering
Bengaluru, Karnataka, India
aryashetty.is23@rvce.edu.in

***Abstract*—Modern supply chains demand transparency, data integrity, and resilience against disruptions. Traditional systems often lack the ability to trace product provenance or assess network vulnerabilities efficiently. This paper introduces SyncChain, a blockchain-based supply chain simulator that integrates secure transaction recording with real-time network visualization and resilience analysis. The system models a multi-tier supply chain network comprising suppliers, processing units, manufacturers, distributors, and retailers. Each transaction is mined into a block containing cryptographically hashed metadata, ensuring immutability and traceability.**

The simulator utilizes Dijkstra's algorithm to compute optimal routing paths based on cost, time, and distance. Network resilience is evaluated using articulation point detection to identify critical nodes whose failure could disrupt the entire supply chain. A dynamic web interface allows users to upload custom node and edge data in JSON format, view live updates, and analyze transaction flows through interactive visualizations powered by vis.js and Flask-SocketIO.

By combining blockchain integrity with graph-theoretic analysis, SyncChain serves as a modular, extensible platform for testing, learning, and validating supply chain models. The system provides actionable insights into both performance optimization and structural weaknesses, making it suitable for educational use, logistics planning, and supply network research.

***Index Terms*—Blockchain, Supply Chain, Network Resilience, Dijkstra Algorithm, Articulation Points, Visualization, Flask, SocketIO.**

I. Introduction

Real-world supply chains often suffer from opaqueness, making fraud detection, delay diagnosis, and accountability difficult. Blockchain provides tamper-evident traceability, and when combined with network analysis, enables systemic insights into structural resilience and bottlenecks.

A. What are Supply Chains?

Supply chains refer to the interconnected network of entities, resources, processes, and technologies involved in the production and distribution of goods and services. These chains typically include suppliers, manufacturers, distributors, retailers, and consumers. The efficiency and coordination of these networks directly affect operational performance, delivery timelines, and product quality. Modern supply chains span multiple geographic regions and depend heavily on data flows and inter-organizational trust.

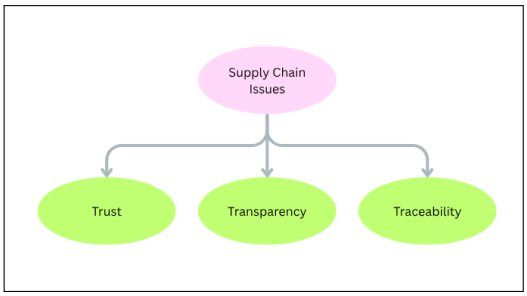

Fig. 1. Challenges Faced in Supply Chains

B. Challenges in Modern Supply Chains

Despite advancements in digital logistics, modern supply chains face significant structural and operational challenges. Among the most critical issues are:

- Trust: Supply chain participants often lack a verifiable means to authenticate the origin, condition, and handling of products. This leads to dependencies on intermediaries and paper-based verification, increasing the risk of fraud and delays.
- Transparency: Visibility across the entire supply chain remains limited, especially in multi-tier networks. Lack of transparency hampers real-time decision-making and hinders response during disruptions or recalls.
- Traceability: The inability to track a product's complete lifecycle, from raw material to consumer, results in ineffective recalls, regulatory non-compliance, and damaged consumer trust. Conventional databases are centralized, making data manipulation and access bottlenecks common.

II. Problem Statement

This project aims to address the core issues of trust, transparency, and traceability in supply chains by leveraging blockchain-based transaction logging and graph analysis. Specifically, it:

- Provides tamper-evident traceability using blockchain for each product's path.
- Enhances transparency by visualizing the supply chain as a directed graph.

- Identifies structural vulnerabilities using articulation point detection

III. Proposed Framework

This work presents a blockchain-based supply chain simulator that integrates network analysis and visualization to assess both logistical efficiency and structural robustness. The system simulates real-time transactions, computes optimal delivery paths using Dijkstra's algorithm, and identifies network vulnerabilities through articulation point detection. The solution supports customizable data inputs, dynamic graph updates, and serves as a testbed for evaluating trust and transparency in a controlled environment. This hybrid framework offers both a didactic model and a scalable foundation for real-world implementation.

IV. Literature Review

Supply chain transparency has emerged as a critical factor in modern logistics and ethical business practices. In his article *"The Transparent Supply Chain," New [5] emphasizes that a lack of visibility across global supply chains often leads to inefficiencies, ethical violations, and diminished consumer trust. He argues that transparency, once considered a liability is now a strategic necessity that can enhance customer loyalty and operational accountability.

Building upon this foundation, blockchain technology has been proposed as a promising enabler of such transparency. Saberi et al. [7] investigate the role of blockchain in sustainable supply chain management, highlighting its capacity to promote traceability, reduce information asymmetry, and support long-term environmental goals. Their study shows that the immutable and decentralized structure of blockchain can address systemic challenges such as fraud and data manipulation.

Similarly, Longo et al. [3] conduct an experimental analysis demonstrating how blockchain enhances trust and collaboration across supply chain actors. Their findings reveal that verifiable and tamper-proof data sharing significantly reduces conflict and improves efficiency in multi-tier supplier networks.

In a simulation-based study, Morelli et al. [4] integrate blockchain with lean supply chain principles. They show that smart contracts combined with cloud infrastructure facilitate low-cost, flexible, and cooperative operations, particularly under dynamic demand conditions. This supports the idea that blockchain is not only a transparency tool but also an efficiency enhancer.

Furthermore, Doorey [6] provides a compelling real-world account of Nike and Levi-Strauss adopting public factory disclosures, driven by stakeholder pressure. While his study is not blockchain-specific, it illustrates how digital systems capable of recording and disclosing operations like blockchain could institutionalize such transparency norms across the industry.

Collectively, these works position blockchain as a strategic solution to persistent issues in supply chain governance namely trust, traceability, and transparency. However, they also indicate that technological adoption must be supported by organizational will and systemic redesign for meaningful impact.

V. Objectives

The problem statement presents many objectives that address multiple interconnected challenges, elaborating on each:

- The objective is to simulate how blockchain can record transactions immutably across a distributed ledger. This ensures tamper-proof tracking of products and associated metadata throughout the supply chain. The model aims to enhance data reliability between suppliers, processors, and consumers.
- Nodes representing suppliers, manufacturers, distributors, and retailers are rendered in an interactive graph. Each edge carries metadata like cost, time, and distance to simulate real-world routing. This provides a visual interface to analyze connectivity and network dynamics.
- The project calculates the shortest paths between supply chain actors based on cost or distance. This optimization supports decision-making for routing goods efficiently. Results also serve as a performance baseline for network health.
- Articulation point detection highlights critical nodes whose failure can fragment the supply chain. Identifying such points informs redundancy planning. It also quantifies resilience in terms of node connectivity and potential bottlenecks.
- Transactions are verified using block mining and hash comparisons to ensure authenticity. Each block contains transactional history, mimicking real-world product tracking. This establishes traceability and builds trust between all network participants.

VI. Methodology

The proposed system simulates a secure and transparent supply chain network by integrating blockchain principles with network theory and visualization techniques. The methodology is organized into the following major components:

1) Network Modeling:
   The supply chain is abstracted as a directed graph where each node represents a stakeholder, such as suppliers, processing units, manufacturers, distributors, and retailers. Edges denote product flow, and each carries attributes like cost, time, and distance. The graph is constructed using the networkx library in Python, with the ability to load custom nodes.json and edges.json files for scalability.
2) Blockchain Integration:
   A simple blockchain structure is implemented where each transaction (product transfer from one node to another) is recorded in a block. Each block includes a timestamp, previous hash, and a list of transactions.

Hashes are generated using SHA-256 to ensure immutability. Mining is simulated by verifying the consistency of transaction chains and maintaining data integrity.

3) Path Optimization using Dijkstra's Algorithm:
The system calculates the shortest paths between all node pairs using Dijkstra's algorithm based on edge costs. This provides insight into optimal routing and helps simulate efficient supply chain logistics. The algorithm is custom-implemented rather than using built-in methods to maintain transparency.
4) Resilience Analysis using Articulation Point Detection:
To analyze network robustness, articulation points (critical nodes whose removal increases disconnected components) are identified in the undirected form of the graph. This helps identify supply chain chokepoints or nodes requiring redundancy planning.
5) Visualization and Frontend Interaction:
The frontend, built with HTML, CSS, and JavaScript, uses the vis.js library to visualize the network interactively. API endpoints (/api/network, /api/blockchain, etc.) serve data dynamically. Users can view node types, transaction history, network metrics, and simulated disruptions in real time.
6) Simulation Results and Analytics:
The dashboard displays computed statistics such as node count, average path cost, number of valid routes, and detected articulation points. This provides a practical and analytical interface to understand the system's performance and behavior under simulated conditions.

Technology Stack and Rationale

The project employs a lightweight, modular set of technology stack used.

- Python was chosen for its rich ecosystem and simplicity. Libraries like networkx support efficient graph modeling and algorithms (Dijkstra, articulation points), while native support for file handling and JSON processing simplifies data input/output.
- Flask (Web Framework) enables the creation of RESTful API endpoints to serve network data, blockchain blocks, statistics, and simulation results. It's lightweight, easy to integrate with other Python components, and requires minimal configuration, making it ideal for student projects and rapid deployment.
- Flask-SocketIO (Real-Time Communication):
Used to simulate real-time updates of blockchain and network states. SocketIO enables push-based communication between server and client, helping reflect new blocks or changes in the network instantly on the frontend.
- JavaScript and vis.js:
The vis.js library is used to render interactive network graphs in the browser. It allows users to visually inspect node types, paths, and articulation points. JavaScript is also used to dynamically fetch and display data from Flask APIs using fetch().

```
dijkstra(graph, source)
// Finds shortest paths from source to all other nodes in a weighted graph
// Input:
// - graph: adjacency list with weights {node: [{to, cost}, ...]}
// - source: starting node
// Output: Dictionary of shortest distances {node: distance}

1. Initialize distance[source] = 0, others = ∞
2. Initialize visited = empty set
3. Initialize priority_queue = [(0, source)]

4. While priority_queue is not empty:
    a. Pop (cost, current) from priority_queue
    b. If current in visited, skip
    c. Mark current as visited
    d. For each neighbor of current:
        i. new_cost = cost + edge cost
        ii. If new_cost < distance[neighbor]:
            - Update distance[neighbor] = new_cost
            - Add (new_cost, neighbor) to priority_queue

5. Return distance
```

Fig. 2. Pseudo Code of Dijkstra's Algorithm

```
articulation_points(graph)
// Identifies nodes which, if removed, increase the number of connected
components
// Input: graph: undirected adjacency list {node: [neighbors...]}
// Output: List of articulation points
1. Initialize:
   - time = 0
   - visited = set
   - disc = {}  // discovery time
   - low = {}   // lowest reachable
   - parent = {}
   - aps = set()
2. Define DFS(node):
   a. visited.add(node)
   b. disc[node] = low[node] = time++
   c. child_count = 0
   d. For each neighbor:
       i. If neighbor not visited:
           - parent[neighbor] = node
           - DFS(neighbor)
           - low[node] = min(low[node], low[neighbor])
           - If parent[node] is None and child_count > 1 → aps.add(node)
           - If parent[node] is not None and low[neighbor] >= disc[node] →
aps.add(node)
           - child_count += 1
       ii. Else if neighbor != parent[node]:
           - low[node] = min(low[node], disc[neighbor])
3. For each node in graph:
   - If not visited, DFS(node)
4. Return aps
```

Fig. 3. Pseudo Code of Finding Articulation Points

- HTML/CSS (User Interface):
A minimal UI is created using HTML for structure and CSS for layout and styling. Focus is placed on readability and data presentation over design complexity.
- JSON (Data Format):
Nodes and edges are loaded from JSON files to allow modular input and testing. JSON ensures easy interchangeability between backend and frontend, and supports batch edits for larger simulations.

```
calculate_hash(index, timestamp, transactions, prev_hash)
// Generates a unique hash for a block using its contents
// Input:
// - index: block number
// - timestamp: creation time
// - transactions: list of transactions
// - prev_hash: hash of previous block
// Output: SHA-256 hash string
1. Concatenate index + timestamp + str(transactions) + prev_hash into a
string
2. Use SHA-256 to hash the string
3. Return the hex digest
```

Fig. 4. Pseudo Code of Hashing Function

```
mine_block(difficulty)
// Finds a nonce such that the block's hash starts with difficulty number
of zeros
// Input: difficulty: integer (e.g. 2 → hash starts with "00")
// Output: Valid hash and nonce
1. nonce = 0
2. Loop:
    a. hash = calculate_hash(index, timestamp, transactions, prev_hash,
nonce)
   b. If hash starts with '0' * difficulty:
        - break
   c. Else:
        - nonce += 1
3. Return nonce and hash
```

Fig. 5. Pseudo Code of Mining Block

VII. Implementation Details

A. Algorithms and Pseudo Code Used

In Fig 2, the Dijkstra's Algorithm finds the shortest path from a source node to all other nodes in a weighted graph with non-negative edge weights. Used for computing optimal paths in the supply chain network.

In Fig 3, the algorithm finds the articulation points in the graph and the critical nodes using Depth First Search. It is concerned with detecting points of vulnerability.

In Fig 4, the algorithm converts input data into a fixed 256-bit hash. Even small changes in input drastically alter the output. Ensures data integrity in each block of the blockchain.

In Fig 5, the algorithm finds a nonce such that the hash of the block data plus nonce meets a difficulty condition. Used to validate and add new blocks securely to the chain.

B. Network Graph Construction

The supply chain network is represented as a directed graph using the networkx Python library. Each node represents an entity in the chain, such as suppliers, manufacturers, distributors, or retailers, and each edge represents a directed flow of goods or information, associated with metrics like cost, time, and distance. The data for nodes and edges is loaded from external .json files, allowing flexibility and scalability. This structure enables not only visualization but also computational analysis, such as pathfinding and fault detection.

C. Blockchain Simulation

Each transaction in the supply chain (e.g., movement of a product from supplier to factory) is encapsulated as a structured JSON object. A set of such transactions is grouped into a block. Each block contains metadata including an index, timestamp, and the cryptographic hash of the previous block. A new hash is computed for every block using the SHA-256 hashing algorithm, ensuring data immutability and tamper resistance. This mimics real-world blockchain behavior in a simplified and controlled environment, sufficient to demonstrate traceability, provenance, and auditability.

D. Shortest Path and Critical Node Analysis

The graph structure allows computation of optimal paths using Dijkstra's algorithm, where the edge cost is the optimization metric. This helps simulate route optimization in logistics. Additionally, articulation points (critical nodes whose failure disconnects the network) are detected using NetworkX's built-in articulation point detection. These points highlight potential vulnerabilities or bottlenecks in the supply chain and emphasize the importance of redundancy and resilience.

E. Backend API Layer (Flask)

The endpoints expose data and functionality from the backend (Flask sever) to frontend (JavaScript). They allow front end to interact with the backend. The endpoint receives an HTTP GET request, and a JSON file is returned.

- */api/network*: Returns the network graph structure in JSON format.
- */api/blockchain*: Serves the current state of the blockchain.
- */api/all_products*: Lists all product transactions recorded in blocks.
- */api/network_stats*: Returns metrics such as average cost and path health.
- */api/articulation_points*: Returns critical nodes in the supply chain.

These APIs modularize the data access and make the application dynamic and easy to extend.

F. Frontend Visualization

The frontend is built using HTML, vanilla JavaScript, and the vis.js visualization library. On page load, data is fetched asynchronously using the fetch() API. Network graphs are rendered interactively with custom color schemes for different node types, and chart.js is used to show node-type distribution. Blockchain data is displayed as expandable panels, showing transaction details and hash integrity. Articulation points and network statistics are also displayed for real-time monitoring.

G. Real-Time Communication

To simulate live updates, Flask-SocketIO is used. Whenever a new block is added (e.g., a product moves from one node to another), the backend emits a block_added event. The frontend listens for these events and updates the display automatically. This real-time feedback loop reflects real supply chain systems where timely updates are critical for decision-making and monitoring.

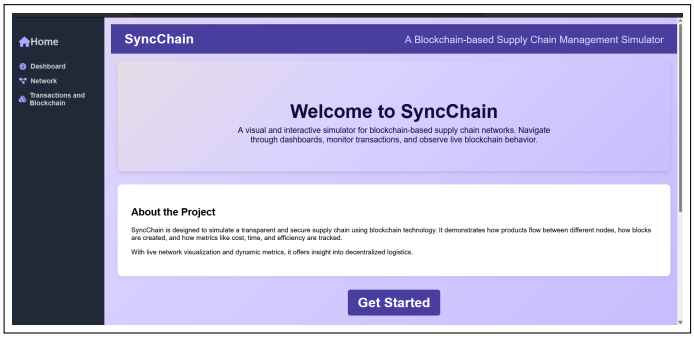

Fig. 6. Landing Page

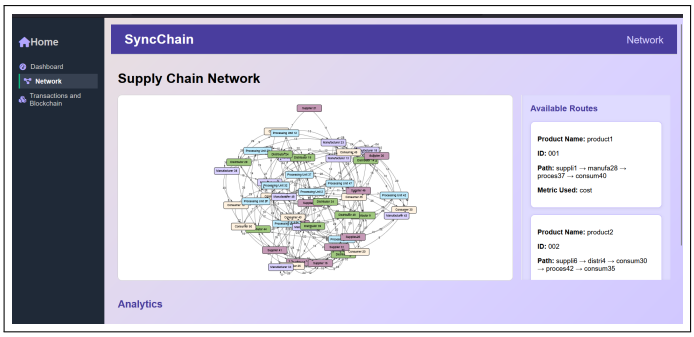

Fig. 7. Network Visualisation and Available Routes

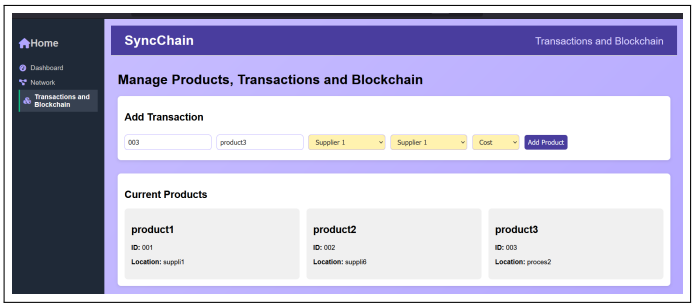

Fig. 8. Product Form

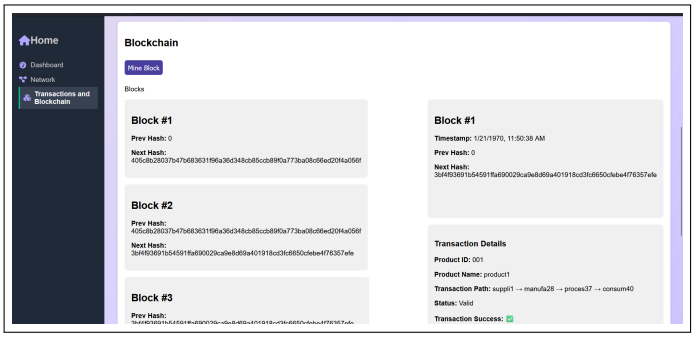

Fig. 9. Blockchain

VIII. Outcomes

A. User Interface and Experience

Looking at Fig 6, the landing page offers a clean dashboard with statistics on transactions, nodes, and blockchain state. Includes charts and summaries.

The Network Visualisation page, Fig 7, gives interactive visualization of supply chain nodes (e.g., suppliers, manufacturers, distributors) and their connections. Includes edge metrics like cost, time, and distance.

In Fig 9, the Blockchain viewer shows blocks with details like timestamp, previous and next hashes, and validity status. Includes transaction data and verification results.

In Fig 8, the Product form and transaction dashboard displays each product's journey through the supply chain as a series of verified transactions. Includes source, destination, and path with success status.

B. Challenges Faced

- Node-Graph Complexity and Visualization:
  Managing and rendering a network of many nodes and edges with real-time updates led to performance issues and required optimization in both frontend rendering and backend processing.
- Handling Dynamic Data Inputs:
  Accepting external JSON inputs for nodes and edges introduced parsing errors, structural inconsistencies, and debugging complexity, especially when validating user-supplied formats.
- Deployment Constraints on PythonAnywhere:
  Flask-SocketIO could not be fully utilized due to lack of WebSocket support on the free-tier environment, requiring fallback implementations for real-time interactivity.
- Integration Between Blockchain and Supply Chain Flow:
  Ensuring each transaction in the supply chain was correctly recorded and visualized on the blockchain simulator required precise synchronization and often revealed race conditions or state mismatches.
- Scalability and Real-Time Throughput:
  As the number of transactions or nodes increases, the system performance degrades noticeably, limiting simulation size without advanced optimization techniques.
- Supply Chain Attacks and Threat Modeling:
  The system does not handle malicious actors like counterfeit data injection, double spending, or Sybil attacks, which are critical in actual blockchain-secured logistics.

IX. Future Scope

This project lays the groundwork for a scalable and transparent supply chain simulation, but there are several promising directions for future development. First, the current simulation can be extended with a fully decentralized backend using real blockchain technologies like Ethereum or Hyperledger Fabric. This would allow integration of smart contracts to automate conditions such as payment release upon delivery, verification of authenticity, or automatic alerts on delays. Additionally, incorporating secure user authentication and role-based access

control can simulate how real-world stakeholders, suppliers, auditors, regulators, interact within a blockchain-based network.

Second, IoT integration can provide live data from physical sensors to enhance the trustworthiness of the supply chain. For instance, temperature or GPS sensors could report real-time product conditions, which are then immutably recorded on the blockchain. Furthermore, incorporating machine learning models could allow for anomaly detection, fraud prediction, or demand forecasting based on historical transaction data. Visual analytics, mobile accessibility, and multilingual support are also essential for adoption in global supply chains and can be considered as progressive enhancements.

X. Conclusion

The project successfully demonstrates the potential of integrating blockchain technology into supply chain networks to enhance trust, transparency, and traceability. By simulating a decentralized ledger system that records each product transaction immutably, the platform mitigates common supply chain issues such as data tampering and lack of accountability. The network graph visualization, Dijkstra-based path optimization, and articulation point detection offer additional insights into the structural health and efficiency of the supply chain. This implementation serves as a proof of concept that secure and transparent logistics can be modeled using open-source tools and minimal hardware resources.

Despite simplifications in consensus mechanisms and decentralized validation, the simulator captures key blockchain properties relevant to real-world applications. The modular architecture allows extensibility into real-time IoT data inputs, smart contract enforcement, and full-stack peer-to-peer validation. The work contributes to the broader discourse on digitalizing supply chains using verifiable technologies and paves the way for more robust, scalable, and autonomous systems that can withstand logistical, regulatory, and cybersecurity challenges in critical infrastructure sectors.

References

[1] J. Lohmer, N. Bugert, and R. Lasch, "Analysis of resilience strategies and ripple effect in blockchain-coordinated supply chains: An agent-based simulation study," *International Journal of Production Economics*, vol. 228, p. 107882, 2020. [Online]. Available: https://doi.org/10.1016/j.ijpe.2020.107882

[2] V. Varriale, A. Cammarano, F. Michelino, and M. Caputo, "Sustainable Supply Chains with Blockchain, IoT and RFID: A Simulation on Order Management," *Sustainability*, vol. 13, no. 11, p. 6372, Jun. 2021. [Online]. Available: https://doi.org/10.3390/su13116372

[3] F. Longo, L. Nicoletti, A. Padovano, G. d'Atri, and M. Forte, "Blockchain-enabled supply chain: An experimental study," *Computers & Industrial Engineering*, vol. 136, pp. 57–69, 2019. [Online]. Available: https://doi.org/10.1016/j.cie.2019.07.026

[4] D. A. Morelli, P. S. de A. Ignacio, and N. L. Zanutim, "Blockchain-Based Lean Supply Chain: A Simulation Approach," *International Journal of Innovation and Technology Management*, vol. 21, no. 05, p. 2450038, Aug. 2024. [Online]. Available: https://doi.org/10.1142/S021987702450038X

[5] S. New, "The transparent supply chain," *Harvard Business Review*, vol. 88, no. 10, pp. 76–82, Oct. 2010.

[6] D. J. Doorey, "The Transparent Supply Chain: From Resistance to Implementation at Nike and Levi-Strauss," *Journal of Business Ethics*, vol. 103, no. 4, pp. 587–603, Nov. 2011. [Online]. Available: https://doi.org/10.1007/s10551-011-0882-1

[7] M. Saberi, M. Kouhizadeh, J. Sarkis, and L. Shen, "Blockchain technology and its relationships to sustainable supply chain management," *International Journal of Production Research*, vol. 57, no. 7, pp. 2117–2135, Apr. 2019. [Online]. Available: https://doi.org/10.1080/00207543.2018.1533261

[8] K. Korpela, J. Hallikas, and T. Dahlberg, "Digital Supply Chain Transformation toward Blockchain Integration," in *Proc. 50th Hawaii Int. Conf. on System Sciences*, 2017.

[9] S. Treiblmaier, "The impact of blockchain on supply chains: a theory-based research framework and a call for action," *Supply Chain Management*, vol. 23, no. 6, pp. 545–559, Oct. 2018. [Online]. Available: https://doi.org/10.1108/SCM-01-2018-0029

[10] S. Kim and M. Laskowski, "Toward an ontology-driven blockchain design for supply-chain provenance," *Intelligent Systems in Accounting, Finance and Management*, vol. 25, no. 1, pp. 18–27, 2018.

[11] M. Casino, T. Dasaklis, and C. Patsakis, "A systematic literature review of blockchain-based applications: Current status, classification and open issues," *Telematics and Informatics*, vol. 36, pp. 55–81, Mar. 2019.

[12] A. Reyna, C. Martín, J. Chen, E. Soler, and M. Díaz, "On blockchain and its integration with IoT. Challenges and opportunities," *Future Generation Computer Systems*, vol. 88, pp. 173–190, Nov. 2018.

[13] IBM Institute for Business Value, "The Trust Factor: The Blockchain Opportunity in Supply Chain Transparency," IBM, 2017. [Online]. Available: https://www.ibm.com/thought-leadership/institute-business-value/report/blockchain-supplychain

[14] M. Kouhizadeh and J. Sarkis, "Blockchain practices, potentials, and perspectives in greening supply chains," *Sustainability*, vol. 10, no. 10, p. 3652, Oct. 2018.

[15] A. Francisco and D. Swanson, "The supply chain has no clothes: Technology adoption of blockchain for supply chain transparency," *Logistics*, vol. 2, no. 1, p. 2, Mar. 2018.

[16] C. Lin, D. He, X. Huang, K. Choo, and A. V. Vasilakos, "BSeIn: A blockchain-based secure mutual authentication with fine-grained access control system for industry 4.0," *Journal of Network and Computer Applications*, vol. 116, pp. 42–52, Jul. 2018.

[17] P. Zhang, D. Schmidt, J. White, and G. Lenz, "Blockchain technology use cases in healthcare," in *Advances in Computers*, vol. 111, pp. 1–41, 2018. [Online]. Available: https://doi.org/10.1016/bs.adcom.2018.03.006

[18] H. Kim and M. Laskowski, "Toward an Ontology-Driven Blockchain Design for Supply Chain Provenance," in *2018 IEEE Int. Conf. on Internet of Things*, pp. 1172–1177, 2018. [Online]. Available: https://doi.org/10.1109/ICIoT.2018.00038

[19] M. Pournader, A. Kach, and J. Talluri, "A Review of the Existing and Emerging Topics in the Supply Chain Risk Management Literature," *Decision Sciences*, vol. 51, no. 4, pp. 867–919, 2020.