Nanotechnology Perceptions
ISSN 1660-6795
www.nano-ntp.com

Adaptive Blockchain-IoT Contracting System for Scalable and Secure E-Commerce Operations

Jeenath Laila N[1], Mohamed Rizwan[2]

[1]*Department of Computer Science and Engineering, Government College of Engineering, Tirunelveli*
[2]*Research Scholar, Karunya Institute of Technology and Sciences, Coimbatore*
Email: jeenathlaila@gcetly.ac.in

Enhancing security, efficiency, and adaptability in e-commerce contexts is the goal of the Blockchain-IoT Integrated Adaptive Contracting System (B-IACS). B-IACS facilitates real-time data analytics, automated resource allocation, and decision-making by utilizing blockchain, IoT sensors, and machine learning. In order to achieve high throughput and scalability in transaction validation, the hybrid consensus mechanism strikes a balance between efficiency and security. The accuracy of resource demand forecasting and supply chain optimization are greatly enhanced by a predictive analytics layer that uses Long Short-Term Memory (LSTM) neural networks. The solution minimizes the time required for contract execution, deposit delivery, and negotiation, demonstrating efficient smart contract execution. Furthermore, the ARM model—which surpasses conventional reputation models in partner selection—offers dynamic, multi-factor trust rating derived on Internet of Things data. Attribute-Based Encryption (ABE) and Access Control Lists (ACLs) are used by the system to ensure strong contract security and privacy-preserving data sharing, with over 97% of risks properly managed. Based on performance and risk management improvements, the results demonstrate B-IACS as a scalable and all-inclusive solution for intelligent industrial e-commerce operations.

Keywords: Blockchain-IoT; Adaptive Contracting System; B-IACS; Hybrid Consensus Mechanism; LSTM Neural Networks; Predictive Analytics; Smart Contracts; Adaptive Reputation Management (ARM); Attribute-Based Encryption (ABE); E-commerce Security; Privacy-Preserving Data Sharing.

1. Introduction

Digital technological advancements along with the growing demands of e-commerce supply

chains have made resource management systems more flexible, safe, and effective. Because of the integration of vertical, horizontal, and end-to-end processes, supply chains have changed from being linear to complex, non-linear networks in the context of Industry 4.0. Innovative solutions are needed since traditional third-party platforms frequently struggle with trust management, data security, and wasteful resource matching. By offering an integrated method of resource management and collaboration in e-commerce contexts, the Blockchain-IoT Integrated Adaptive Contracting System (B-IACS) tackles these issues. B-IACS improves security, efficiency, and transparency throughout the supply-demand network by utilizing blockchain, IoT, and adaptive intelligence processes.

The decentralized structure of blockchain allows for safe transactions, tamper-proof data sharing, and smooth execution of smart contracts. IoT sensor integration enables real-time data collecting, which supports efficient resource allocation, predictive analytics, and prompt decision-making.

In order to provide quick and trustworthy transaction validation, B-IACS uses a hybrid consensus method that strikes a balance between security and efficiency. The system presents an Adaptive Reputation Management (ARM) approach for correct partner selection and trust-building. ARM leverages dynamic, multi-factor trust scoring. Long Short-Term Memory (LSTM) neural networks form the foundation of a predictive analytics layer that optimizes supply chain operations and allows for accurate resource demand forecasting. These characteristics greatly improve operational effectiveness and produce a flexible environment that can change to meet changing user demands and market situations. Furthermore, by using intelligent contract automation, B-IACS streamlines contract procedures by cutting down on the times associated with negotiation, deposit delivery, and execution. Because Attribute-Based Encryption (ABE) and Access Control Lists (ACLs) pro- tect user privacy, the platform's resilience to possible threats is demonstrated. B-IACS is po- sitioned as a sophisticated solution for scalable, safe, and effective e-commerce operations, changing the field of intelligent supply chain management through the comprehensive integration of predictive analytics, adaptive reputation, and secure smart contracts.

2. Literature Review

Using Alliance blockchain and smart contracts, a blockchain-based service platform is intended to improve resource allocation in industrial supply and demand networks and foster greater cooperation (He & Zhang, 2022). To guarantee safe, transparent transactions, techniques include creating and deploying contracts with the Remix IDE. Although the platform reduces transaction redundancy and increases trust and operational efficiency, it has drawbacks such as scalability concerns and the complexity of implementing smart contracts.

A comprehensive evaluation of supply chain traceability systems enabled by blockchain is carried out by Dasaklis et al. (2022), with an emphasis on technological implementation, maturity, and sustainability in terms of social, environmental, and economic domains. The methods comprise the domain, methodology, and implementation maturity classification of the body of extant literature. Enhancing traceability and transparency in delicate industries like food and medicines is one advantage. Yet, issues with practicality in the real world, financial

concerns, and the dearth of organized research in higher education persist.

Aslam et al. (2022) investigate how blockchain might improve supply chain management (SCM) in the Oil and Gas (O&G) sector, emphasizing lean and agile SCM techniques. The paper presents a system where blockchain enhances agile SCM through features like transparency, smart contracts, and cybersecurity. It does this by using data from SCM managers to examine the influence on business performance. The benefit is improved OG SCM responsiveness and agility; however, there are drawbacks as well, such as complicated integration and the requirement for specialized blockchain solutions for flexible supply chains.

Using a technological adoption model, Diffusion of Innovation (DOI) theories, and a review of the literature, Agi Jha (2022) provide a thorough framework for blockchain adoption in the supply chain and identify 20 enablers. Methods include evaluating the influence and interdependencies of enablers from supply chain, technological, organizational, and external viewpoints using the DEMATEL approach. Benefits include being able to pinpoint important elements like the relative technological advantage and outside forces affecting adoption. One drawback is that more empirical validation in various contexts and with greater sample sizes is required. Chen (2020) discusses the problem of user data protection in e-commerce platforms, emphasizing how autonomy laws permit personal information to be misused. The study suggests a method for third-party credit evaluation in order to evaluate and enhance platform governance regulations. Improved platform self-governance and increased protection of user rights are two benefits. On the other hand, maintaining platform conformity and putting standardized assessments into practice present difficulties. Using four stages of classification pre-adoption, adoption, implementation, and application Manzoor (2022) offers a thorough literature analysis on the topic of blockchain technology's application to supply chain management. A study paradigm that improves supply chain performance and resilience through blockchain is proposed by means of synthesizing findings. An easy way to use blockchain technology into supply chains to increase productivity is a benefit. The need for additional empirical research and resolving real-world adoption and implementation issues are among the constraints.

Yuan et al. (2020) investigate how the decentralized, secure, and traceable data structures provided by blockchain technology might improve supply chain management information systems. Techniques include emphasizing consensus, transaction process optimization, and system architecture design while putting forth a collaborative method for supply chain management. The benefits include lower costs for intermediaries, more confidence, and more effective transaction procedures. However, the difficulty of putting blockchain systems into place and guaranteeing their broad industrial acceptance present possible disadvantages.

Blockchain technology plays a crucial role in sustainable supply chain management by enhancing traceability and transparency. A systematic review of 187 peer-reviewed publications from the year 2017to 2020 was conducted using the 5W+1H framework. The study introduces a classification framework based on grounded theory and technology readiness levels. Findings show a growing interest in blockchain-based supply chains since 2017. The key benefits identified include improved data integrity, decentralized trust, and efficiency. Managers and business leaders can leverage these insights for sustainability-driven decision-making. The research highlights blockchain's disruptive power in reshaping supply

chains. It also provides a roadmap for practitioners by guiding them to relevant literature and key information sources. (Vineet et.al. 2020).

In IoT environments, Li et al. suggest utilizing microgrids as a decentralized on-demand energy source for blockchain mining. Making the energy allocation into a Stackelberg game to maximize profitability for microgrids and miners is one technique, along with creating an energy supply architecture specific to mining needs. As a result, blockchain mining is supported by more equitable and efficient energy distribution, which is beneficial for sustainability. Implementation difficulties, however, could arise from the intricacy of microgrid connectivity and varying energy demands.

Ayan et al. use a mixed-method approach that includes bibliometric analysis and content assessment of 552 publications from 2017–2022, to give an extensive review of blockchain technology in sustainable supply chains across diverse industries. The methods include using thematic mapping and bibliometric laws (such as Bradford's Law) to draw attention to important subjects such as Industry 4.0 and traceability, with a particular emphasis on the food and agriculture industry. Among the benefits are the ability to spot new trends and blockchain applications in the sustainability space. However, the study's breadth is constrained by its dependence on particular databases, and it might miss some subtleties unique to a given business.

Industry 4.0 (I4.0) plays a crucial role in enhancing supply chain (SC) resilience by integrating digital technologies. A scoping review examines the literature on SC resilience and I4.0-enabled SC management. The study by Tortorella et. al. (2021) identifies three key research directions: empirical validation of I4.0's impact, exploring processing-actuation technologies for restorative capacity, and integrating I4.0 ICTs with omni-channel strategies to improve resilience. The literature primarily focuses on I4.0 applications at different SC tiers, while fewer studies address resilient smart SC design. Findings suggest that I4.0 can transform SC management by enabling faster, more effective responses to disruptions. However, digital transformation in SCs is still in its early stages. Research on the direct link between I4.0 and SC resilience remains limited. The study organizes existing knowledge to guide future theoretical developments. This work helps managers understand how digital technologies strengthen SC resilience. The integration of I4.0 technologies will be key to future adaptive and robust SC systems.

A digitally enabled circular economy (CE) enhances resource efficiency by ensuring tracing, tracking, and storing of product information. This study explores the application of IoT and Blockchain in managing Electrical and Electronic Equipment (EEE) in Italy. Preventing electronic waste (WEEE) is crucial, as it contains both hazardous substances and valuable materials. IoT and Blockchain help manufacturers monitor products until their end-of-life, supporting CE strategies and decision-making. Based on industry interviews, the study proposes three solution variations and their business models. The findings highlight the environmental and economic benefits of integrating digital technologies in sustainable EEE and WEEE management (Magrini et.al. 2021).

Khan et al. examine the problems with IoT security and provide a blockchain-based architecture for industrial IoT (IIoT) that uses Hyperledger Sawtooth to handle data integrity, trust, and privacy. Methods for safe, resource-efficient IIoT transactions include hashing trees,

a bespoke consensus system, and NuCypher Re-Encryption. Enhanced security, reliable execution, and effective node-to-node communication are benefits. The resource limitations in IIoT contexts and the high processing demands for blockchain proof-of-work, however, continue to be obstacles.

3. Proposed System

Enhancing security, efficiency, and adaptability in e-commerce supply and demand networks is the goal of the B-IACS). B-IACS offers real-time data-driven decision-making, safe transactions, and dynamic resource allocation by combining blockchain technology, Internet of Things sensors, predictive analytics (LSTM models), and adaptive intelligent contracts.

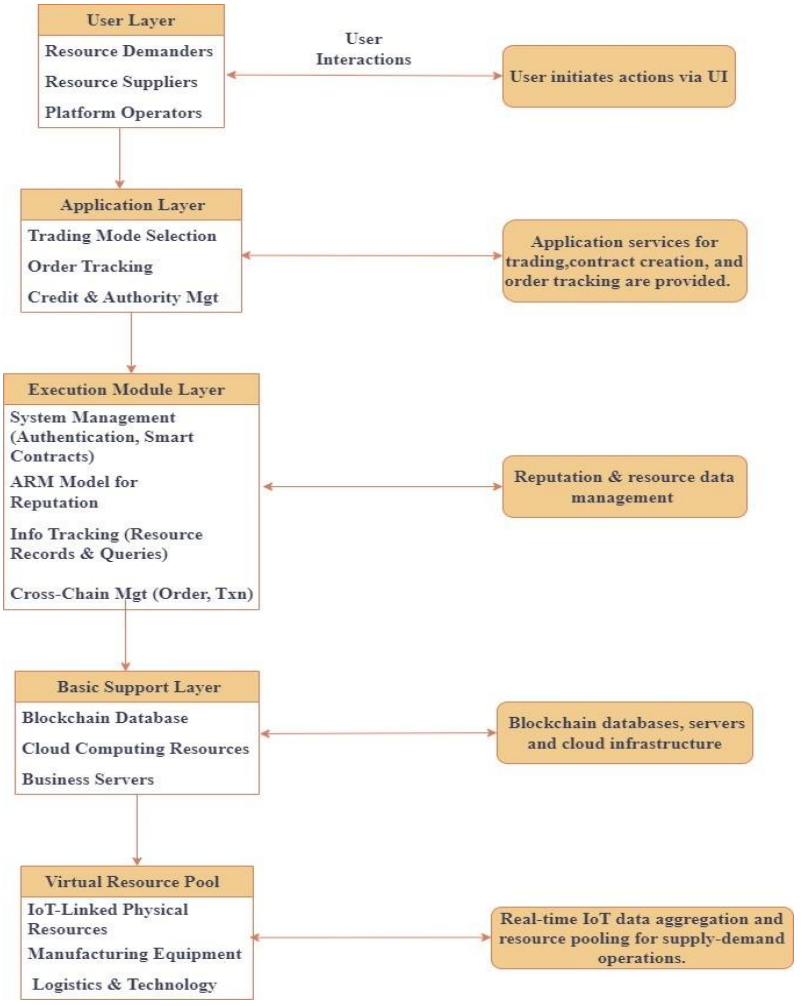

Figure 1. Proposed Sytem Architecture

The architecture of the proposed system is depicted in Figure 1. All users who communicate with the platform by sending and receiving data are included in the User Layer, including

operators, demanders, suppliers, and other players. In order to enable secure transactions, the Application Layer includes essential features including trading mode selection, contract creation, order tracking, and authority/credit management. The ARM model, information tracking, reputation, smart contract management, and cross-chain trans- action processing are all carried out by the Execution Module Layer. Blockchain databases are used in the Basic Support Layer to provide real-time updates, cloud computing, and secure storage. By combining physical resources with real-time data from IoT devices, Virtual Resource Pool enables precise and dynamic resource matching.

Transaction Process Flow

Matching Transaction Objects

Leveraging data from IoT sensors, the platform dynamically links suppliers and demanders based on their real-time availability, resource capacity, and demands. This guarantees that resources are used effectively and that the transaction's requirements are satisfied quickly and safely.

Let E = {e1, e2, ..., en} be the set of all enterprises (suppliers and demanders), and let m(Dnt) represent the resource capacity of each enterprise ei at time t. The real-time availability of resources for ei is given by Ravail(ei, t) = $\sum$n sij(t), where sij(t) denotes the stock of each resource type j. The conditions for matching are as follows: If m(Dnt) > qn, where qn is the required quantity, then ei acts as a supplier. Conversely, if m(Dnt) < qn, then ei acts as a demander.

Real-Time Resource Monitoring and Role Reassignment

The platform leverages real-time IoT data to reassign jobs and adjust availability in response to changes in resources and needs.

Algorithm 1 Monitor Resources

1: Function MonitorResources(suppliers, interval)

2: while True do

3: for all supplier in suppliers do

4: supplier.update_resource_availability(IoT_data_stream(supplier.resources))

5: end for

6: Wait for the specified interval

7: end while

The algorithm MonitorResources continuously monitors and updates the resource avail-

ability of all suppliers at a specified time interval. It uses IoT data streams to get real-time information on the resources of each supplier. The function sleeps for the given interval before repeating the update process.

Reassigning Roles Based on Resource Dynamics

Roles of participants, such as provider or demander, are reallocated in accordance

with updated resource capacity and demands to guarantee efficient matching.

Algorithm 2 Reassign Roles

1: Function ReassignRoles(enterprises)

2: for all enterprise in enterprises do

3: # Check if the enterprise has excess resources

4: if enterprise.resource_capacity > enterprise.required_resources then

5: Set enterprise.role to 'Supplier'

6: else

7: Set enterprise.role to 'Demander'

8: end if

9: end for

The algorithm Reassign Roles iterates over a list of enterprises and checks each one's resource capacity against its required resources. If an enterprise's capacity exceeds its requirements, its role is set to 'Supplier'; otherwise, it is set to 'Demander'. This reassignment ensures that roles are dynamically adjusted based on resource availability.

Prioritization and Optimization of Matching

To identify the best matches dynamically, the matching process is optimized based on urgency, cost, and reputation scores.

Algorithm 3 Prioritized Match

1: Function PrioritizedMatch(demanders, suppliers)

2: Output: match_list (list of tuples containing matched supplier and demander)

3: Initialize match_list as an empty list

4: Sort demanders by urgency in ascending order

5: for all demander in sorted demanders do

6: Initialize potential_suppliers as the list of suppliers whose available_resources demander's required_resources

7: Sort potential_suppliers by proposed_price in ascending order and reputation_score in descending order

8: if potential_suppliers is not empty then

9: Append the tuple (first potential_supplier, demander) to match_list

10: end if

11: end for

12: Return match_list

The Prioritized Match algorithm iterates over demanders sorted by urgency and

identifies potential suppliers who meet the required resources. It sorts these suppliers based on their proposed price and reputation score and appends the best match (first suitable supplier) to the matchlist. The process continues for all demanders, resulting in a prioritized list of optimal supplier-demand pairings.

Resource Matching with Reputation and Historical Performance

Enhancing trust and efficiency, the matching process incorporates past performance

and reputation.

Algorithm 4 Match Resources with Resource Availability

1: Function MatchResources(demanders, suppliers)

2: Output: match_list (list of tuples containing matched supplier and demander)

3: Initialize match_list as an empty list

4: for all demander in demanders do

5: Set demand_quantity as demander's required_resources

6: Set max_price as demander's max_price

7: for all supplier in suppliers do

8: Calculate available_resources as the sum of IoT data for all resources of the supplier

9: Set proposed_price as supplier's proposed_price

10: if available_resources demand_quantity and proposed_price max_price and

supplier's reputation_score > 0.7 then

11: Append the tuple (supplier, demander) to match_list

12: end if

13: end for

14: end for

15: Return match_list

16: Function GetResourceAvailability(supplier)

17: Calculate available_resources as the sum of IoT data for all resources of the supplier

18: Return available_resources

The algorithm Match Resources with Resource Availability matches demanders with suppliers based on resource requirements, maximum price, and reputation score. It iterates over each demander and calculates the available resources for each supplier using IoT data. If the supplier meets the required resource quantity, price constraints, and has a reputation score

greater than 0.7, the supplier-demand pair is added to the matchlist. The GetResourceAvailability function, defined within the algorithm, calculates the total available resources for a supplier by summing up all resource data from IoT streams. The algorithm outputs a list of matched supplier-demand pairs that meet all criteria.

Multi-Resource Matching and Allocation

For complex scenarios, multi-resource matching is employed to find suppliers that collectively meet the demand.

Algorithm 5 Multi-Resource Match

1: Function MultiResourceMatch(demanders, suppliers)

2: Output: match_list (list of tuples containing matched suppliers and demander)

3: Initialize match_list as an empty list

4: for all demander in demanders do

5: Set required_resources as demander's list of required resources

6: Initialize matched_suppliers as an empty list

7: for all resource in required_resources do

8: Set potential_suppliers as the list of suppliers who have the given resource

9: Sort potential_suppliers by proposed_price in ascending order and reputation_score in descending order

10: if potential_suppliers is not empty then

11: Append the first potential_supplier to matched_suppliers

12: end if

13: end for

14: if the length of matched_suppliers is equal to the length of required_resources then

15: Append the tuple (matched_suppliers, demander) to match_list

16: end if

17: end for

18: Return match_list

The Multi-Resource Match algorithm iterates over each demander and attempts to match their required resources with suppliers. For each resource needed by a demander, it finds suppliers who have that resource and sorts them by proposed price and reputation score. If a suitable supplier is found, it is added to the list of matched suppliers for that resource. If all required resources for a demander have matching suppliers, the algorithm appends the matched suppliers and demander as a tuple to the matchlist. Finally, it returns the matchlist containing all successful supplier-demand matches.

Smart Contract Formation

Smart contracts automate agreements in B-IACS, focusing on pricing, deposit management, and transaction execution, enabling efficient and secure resource allocation in e-commerce operations.

Key Types of Smart Contracts

Price Negotiation Contract (PNC)

This contract handles the negotiation between the supplier and demander to reach an agreement on the final transaction price.

Condition for Agreement: The final price Pfinal is set when the proposed price from the

supplier Ps matches or is below the maximum price offered by the demander Pd:

Pfinal = Ps = Pd(1)

Deposit Delivery Contract (DDC)

Both the supplier and demander must pay a deposit to ensure commitment and

mitigate risks.

The conditions for Deposits are as follows:

Dd = Pfinal × rd (demander's deposit) (2)

Ds = Pfinal × rs (supplier's deposit) (3)

Where rd and rs are predefined deposit rates, typically around 10% of the transaction amount.

Order Transaction Contract (OTC)

This contract governs the order placement, resource delivery, and payment completion.

Condition for Payment Completion: Upon confirmation of delivery:

Premaining = Pfinal − Dd (4)

The remaining payment Premaining is transferred to the supplier.

Algorithm 6 Create Smart Contracts

1: Function CreateSmartContracts(demander, supplier, final_price)

2: Output: A tuple containing the price contract, deposit contract, and order contract or a failure message

3: Step 1: Initialize Contracts

4: Initialize price_contract using initialize_price_contract with demander and supplier details

5: Initialize deposit_contract using initialize_deposit_contract with demander, supplier, and final_price

6: Initialize order_contract using initialize_order_contract with demander,

supplier, and final_price

7: Step 2: Price Negotiation

8: if negotiate_price(price_contract) then

9: Finalize the price_contract

10: else

11: Return "Price Negotiation Failed"

12: end if

13: Step 3: Secure Deposits

14: if secure_deposits(deposit_contract, demander, supplier) then 15: Return (price_contract, deposit_contract, order_contract) 16: else

17: Return "Deposit Collection Failed"

18: end if

The algorithm Create Smart Contracts initializes contracts between a demander and a supplier, ensuring all necessary details are set up based on the final agreed price. First, it creates a pricecontract, depositcontract, and ordercontract. Next, it performs a price negotiation; if successful, the contract is finalized, or otherwise, a failure message is returned. Then, it secures deposits from both parties; if deposits are successfully collected, all contracts are returned; otherwise, a message indicating deposit collection failure is given. The algorithm ensures successful creation and execution of smart contracts for secure transactions.

Execution of Smart Contracts

Smart contracts execute automatically based on predefined conditions and real-time data from IoT sensors, ensuring seamless resource delivery and secure transactions.

Algorithm 7 Execute Smart Contracts and Complete Payment

1: Function ExecuteSmartContracts(order_contract, deposit_contract) 2: Output: Status message indicating success or failure of execution 3: Step 1: Verify Deposits

4: if verify_deposits(deposit_contract) then

5: Set order_contract status to "active" 6: Step 2: Supplier Delivers Resources 7: Supplier delivers resources

8: Set delivery_status as the result of check_delivery_status(order_contract, IoT_data_stream)

9: Step 3: Confirm Delivery and Proceed to Payment

10: if confirm_delivery(demander, delivery_status) then

11: Calculate remaining_amount as order_contract.price - de- posit_contract.demander_deposit

12: Transfer remaining_amount from demander to supplier

13: Return deposits to both parties as per contract terms

14: Set order_contract status to "completed"

15: else

16: Handle issues in delivery (e.g., delay, incorrect resource) using handle_default(order_contract, deposit_contract, delivery_status)

17: end if

18: else

19: Return "Deposit Verification Failed"

20: end if

The algorithm Execute Smart Contracts and Complete Payment handles the entire process of verifying deposits, resource delivery, and payment completion between a demander and a supplier. First, it verifies both parties' deposits; if successful, the contract status is set to "active." The supplier delivers resources, and the delivery status is checked via IoT data. If the delivery is confirmed, the remaining payment is calculated and transferred, and deposits are returned as per contract terms, marking the contract as "completed." If any issues arise during delivery, the default handling mechanism is triggered; otherwise, a failure message is returned if deposits are not verified.

Adaptive Reputation Management (ARM)

The ARM model in B-IACS extends beyond standard reputation calculations by dynamically adapting to changing conditions in real-time. It leverages IoT data, delivery performance, and cooperation indices to improve partner selection and ensure accurate, responsive trust scores.

$$R_{i,t} = \alpha t P_{i,t} + \beta t S_{i,t} + \gamma t C_{i,t} + \delta t \, f(IoTt) \quad (5)$$

Where:

- $P_{i,t}$: Delivery performance indicator.
- $S_{i,t}$: Average satisfaction based on past interactions.
- $C_{i,t}$: Composite cooperation index.
- $f(IoTt)$: Real-time IoT factor providing real-time adjustments to reputation.

In the ARM model, the weights αt, βt, γt, δt dynamically adjust based on market conditions, network delays, and real-time IoT data. This adaptive weighting allows for responsive reputation management that standard models lack

Weight Adjustment Over Time

The weights are adjusted by considering both market conditions and IoT real-time factors:

$$\alpha t+1 = \alpha t + \Delta\alpha(\text{fmarket}) \quad \beta t+1 = \beta t + \Delta\beta(\text{fmarket}) \quad \gamma t+1 = \gamma t + \Delta\gamma(\text{fmarket})$$

$$\delta t+1 = \delta t + \Delta\delta(\text{fmarket}, \text{fIoT}) \quad (6)$$

Algorithm 8 Calculate Reputation

```
Function CalculateReputation(supplier, IoT_data, market_conditions)
    Output: Ri (final reputation score for the supplier)
    Step 1: Calculate Performance Indicators
    Try
    Pi = supplier.successful_deliveries / supplier.total_deliveries
    Catch ZeroDivisionError
    Pi = 0  // No deliveries made, set performance to neutral
    Si = sum of supplier.satisfaction_ratings / max(1, length of sup-
plier.satisfaction_ratings)  // Avoid division by zero
    Ci = compute_cooperation_index(supplier)
    Step 2: Calculate Real-Time IoT Factor
    IoT_factor = calculate_IoT_factor(IoT_data)
    Step 3: Adjust Weights Based on Market Conditions
    (α, β, γ, δ) = adjust_weights_based_on_market(market_conditions)
    Step 4: Calculate Final Reputation Score
    Ri = (α Pi) + (β Si) + (γ Ci) + (δ IoT_factor)
    Return Ri
```

The algorithm Calculate Reputation computes a reputation score for a supplier by considering performance indicators, satisfaction ratings, cooperation index, and real-time IoT factors. First, it calculates Pi (delivery performance), handling any division by zero. It then calculates Si (average satisfaction) and Ci (cooperation index). The IoT factor is derived from real-time data, and weights (α, β, γ, δ) are adjusted based on market conditions. Finally, the reputation score Ri is calculated as a weighted sum of all these factors, and the score is returned as the output.

3.5. LSTM-Based Predictive Analytics for Demand Forecasting

Long Short-Term Memory (LSTM) neural networks are used by B-IACS to precisely predict supply and demand trends. In order to generate accurate predictions, the LSTM model examines sequential data, discovers long-term dependencies, and dynamically integrates real-time IoT sensor data. The input data sequence is represented as $X_t = x_1, x_2, ..., x_t$, where x_t includes historical demand, supply trends, and real-time IoT data.

The forget gate decides which part of past data to discard:

$ft = \sigma(Wf \cdot [ht-1, xt] + b\,f)$ (7)

This ensures that irrelevant past data does not influence the prediction.

The input gate and cell state update the cell state with new relevant information:

$it = \sigma(Wi \cdot [ht-1, xt] + bi)$ (8)

$C\tilde{}t = \tanh(WC \cdot [ht-1, xt] + bC)$ (9)

$Ct = ft \cdot Ct-1 + it \cdot C\tilde{}t$ (10)

Real-time IoT data is integrated into the input gate, ensuring up-to-date resource availability information.

The output gate produces the hidden state used for prediction:

$ot = \sigma(Wo \cdot [ht-1, xt] + bo)$ (11)

$ht = ot \cdot \tanh(Ct)$ (12)

The hidden state ht informs real-time adjustments in resource allocation within B-IACS.

Algorithm 9 LSTM_Predict

1: Function LSTM_Predict(X, IoT_data, model_parameters)

2: Output: ypred (predicted demand/supply)

3: Step 1: Initialize LSTM Cell States

4: Initialize cell state Ct and hidden state ht with initial values

5: Step 2: Iterate Through Each Time Step t

6: for t = 1 to length(X) - 1 do

7: // Update real-time IoT data

8: Update X[t] using corresponding IoT_data[t]

9: // Forget Gate: Determine which part of cell state to keep

10: Calculate ft = σ(Wf concatenate([ht, X[t]]) + b f) 11:// Input Gate: Determine which part of input to use 12: Calculate it = σ(Wi concatenate([ht, X[t]]) + bi)

13: Calculate C˜t = tanh(WC concatenate([ht, X[t]]) + bC)

14: // Update Cell State: Combine old state, forget gate, and input gate

15: Update Ct = ft Ct + it C˜t

16: // Output Gate: Compute the new hidden state 17: Calculate ot = σ(Wo concatenate([ht, X[t]]) + bo) 18: Update ht = ot tanh(Ct)

19: end for

20: Step 3: Compute Prediction

21: Calculate ypred = Wy ht + by

22: Return ypred

The LSTM_Predict algorithm processes a sequence of input data (X) and real-time IoT data to predict demand or supply using an LSTM model. It initializes the LSTM's cell state (Ct) and hidden state (ht) and iterates through each time step of X, updating with IoT data. For each step, the forget gate determines which past data to retain, the input gate decides which new information to add, and the output gate computes the hidden state. Finally, the algorithm computes the prediction ypred as a function of the hidden state and model parameters.

3.6. Hybrid Consensus Mechanism

The B-IACS hybrid consensus mechanism combines the strengths of Proof of Stake and Proof of Work to achieve scalable, efficient, and secure transaction validation. It prioritizes nodes based on their stake, reputation, and performance, while ensuring secure block validation through computational rigor with reduced difficulty.

Node Selection

Nodes are chosen based on a composite stake, which includes their token holdings, reputation (adaptive score from ARM), and historical performance:

$$P_i = \frac{S_i}{\sum_{j=1} S_j}, \quad S_i = \lambda \cdot \text{tokens} + \mu \cdot \text{reputation} + \nu \cdot \text{performance} \quad (13)$$

where:

- S_i: Composite stake of node i.
- λ, μ, ν: Weights for tokens, reputation, and performance.

Nodes with higher composite stakes have a greater chance of being selected.

Block Validation (PoW)

The selected node validates a block using a reduced-difficulty PoW:

$$H(B_i) < T' \quad (14)$$

where:

- $H(B_i)$: Hash of the block.
- T': Target difficulty adjusted based on node reputation.

The lower the target difficulty, the quicker a block can be validated, maintaining security without overburdening the system.

Algorithm 10 HybridConsensus

1: Function HybridConsensus(nodes, transactions, target_difficulty)

2: Output: Block added to the blockchain

3: Step 1: Node Selection Based on Composite Stake

4: total_stake = Sum of composite_stake for all nodes in nodes

5: selected_node = Select node based on its composite_stake from nodes using to- tal_stake

6: Step 2: Block Validation with Reduced Difficulty

7: while block is not validated do

8: Create a new_block using transactions and selected_node

9: block_hash = Hash of new_block

10: if block_hash < AdaptDifficulty(target_difficulty, selected_node.reputation_score) then

11: Add new_block to blockchain

12: Notify all participants that "Block validated by selected_node"

13: break

14: end if

15: end while

16: Function AdaptDifficulty(base_difficulty, reputation_score)

17: Return base_difficulty × (1 - 0.01 × reputation_score)

The HybridConsensus algorithm combines Proof of Stake and Proof of Work for secure and efficient block validation in a blockchain. First, nodes are ranked based on their composite stake, which includes tokens, reputation, and performance, to select a validating node. Then, the selected node repeatedly creates and hashes a block until its hash meets a reduced difficulty threshold adjusted by the node's reputation. If the block meets the criteria, it is added to the blockchain, and all participants are notified. The AdaptDifficulty function dynamically adjusts the difficulty level based on the node's reputation, making it easier for reputable nodes to validate blocks.

Privacy-Preserving Data Sharing4

Attribute-Based Encryption (ABE) enables fine-grained access control by encrypting data based on user attributes, allowing only those with matching attributes to access and decrypt information.

Encryption: A user encrypts data D using an access policy:

E(D) = Encrypt(D, KABE, Policy(A)) (15) where Policy(A) is a Boolean expression of attributes A.

Decryption: A user with attributes U can decrypt E(D) if U satisfies Policy(A).

Algorithm 11 EncryptData

1: Function EncryptData(data, KABE, policy_attributes)

2: Output: encrypted_data (data after encryption)

3: Step 1: Use Attribute-Based Encryption (ABE) to encrypt data

4: encrypted_data = ABE_encrypt(data, KABE, policy_attributes)

5: Step 2: Return the encrypted data

6: Return encrypted_data

Algorithm 12 DecryptData

1: Function DecryptData(encrypted_data, user_attributes, KABE)

2: Output: decrypted_data (if access is granted)

3: Step 1: Check if user's attributes satisfy the access policy

4: if check_attributes(user_attributes, policy_attributes) is True then

5: Step 2: Decrypt the data using ABE

6: decrypted_data = ABE_decrypt(encrypted_data, KABE)

7: Step 3: Return the decrypted data

8: Return decrypted_data

9: else

10: Raise AccessDeniedError ("User does not meet access policy")

11: end if

The EncryptData algorithm uses Attribute-Based Encryption to encrypt data based on an access policy defined by attributes. It takes the data, an encryption key, and policy attributes, then returns the encrypted data. The DecryptData algorithm checks if a user's attributes satisfy the access policy before decrypting the data. If the user meets the policy requirements, the encrypted data is decrypted using the ABE decryption key. Otherwise, an error is raised, indicating that the user does not have access rights.

Access Control Lists (ACLs)

ACLs provide fine-grained access control by defining user-specific permissions for resource access in B-IACS. An ACL for each resource specifies the users and their read, write, update, and other permissions.

Algorithm 13 CheckAccess

1: Function CheckAccess(user, resource, permission)

2: Output: Boolean (True if access is granted, otherwise raise error)

3: Step 1: Retrieve Access Control List (ACL) for the resource

4: acl = get_acl(resource)

5: Step 2: Check if user has the required permission

6: if user is in acl['allowed_users'] AND acl['permissions'][user] equals permission then

7: Return True

8: else

9: Raise AccessDeniedError ("User does not have access rights")

10: end if

The CheckAccess algorithm verifies if a user has the necessary permissions to access a given resource. It first retrieves the ACL for the specified resource, which contains the allowed users and their permissions. It then checks if the user exists in the list of allowed users and whether their permission matches the required one. If both conditions are met, the algorithm returns True, granting access; otherwise, it raises an AccessDeniedError, indicating that the user lacks the appropriate access rights.

3.9. Cross-Chain Transactions

Within B-IACS, cross-chain transactions enable smooth interactions between several blockchains, enabling safe resource sharing and data interchange. Verification and Execution: A cross-chain transaction is successful if:

$$\text{Verify}(T_{source}) \rightarrow \text{Execute}(T_{target}) \quad (16)$$

Algorithm 14 CrossChainTransaction

1: Function CrossChainTransaction(source_chain, target_chain, transaction_data)

2: Output: Successful execution of cross-chain transaction or an error

3: Step 1: Verify transaction on the source chain

4: if verify_transaction_on_chain(source_chain, transaction_data) is True then

5: Try:

6: Step 2: Execute transaction on the target chain

7: execute_transaction_on_chain(target_chain, transaction_data)

8: Step 3: Update status on both chains

9: update_status_on_source_chain(source_chain, transaction_data)

10: update_status_on_target_chain(target_chain, transaction_data)

11: Catch TransactionExecutionError (e):

12: log_error("Error in executing cross-chain transaction:", e)

13: Raise error e

14: else

15: Raise VerificationError ("Transaction could not be verified on the source chain")

16: end if

The CrossChainTransaction algorithm manages the process of executing transactions across different blockchains. It first verifies the transaction on the source chain to ensure it's valid. If successful, it attempts to execute the transaction on the target chain, updating the status on both chains. If any error occurs during execution, such as a TransactionExecutionError, it logs the issue and raises an exception. If the initial verifi- cation fails, a VerificationError is raised, preventing the transaction from proceeding.

4. Experimental Results

The experimental setup for the Blockchain-IoT Integrated Adaptive Contracting System consists of a permissioned blockchain network that includes multiple nodes representing suppliers, demanders, and platform operators. These nodes interact to manage real-time resource allocation, contract execution, and e-commerce transactions. IoT sensors are integrated into the environment, linked to physical resources such as manufacturing equipment, logistics systems, and environmental monitoring, providing real-time data streams that are utilized for decision-making and adaptive reputation scoring. A synthetic dataset is created to simulate real-world operations, encompassing transaction details (such as pricing, timestamps, and contract parameters), IoT sensor readings (including real-time resource availability, predicted demand, and performance metrics), and smart contract metrics. The dataset includes comprehensive information like transaction throughput, satisfaction indices, resource utilization, and historical demand, allowing for the testing and validation of various system functions, including secure contract execution, dynamic resource matching, and cross-chain data exchange.

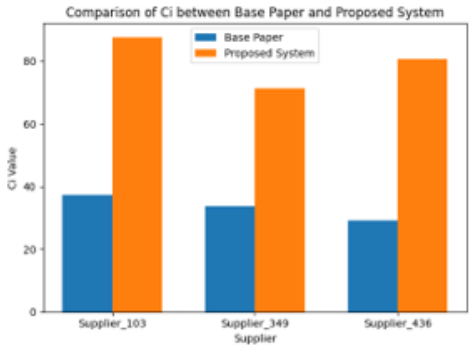

Figure 2. Comparison-Cooperation Index Ci

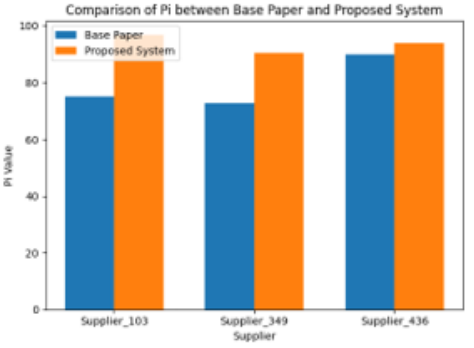

Figure 3. Comparison-Performance Indicator Pi

Figure 2 compares the Ci values of three suppliers (Supplier_103, Supplier_436, and Supplier_349) between the base system and the proposed system. The proposed system shows a significantly higher Ci value across all suppliers compared to the base system, which is the design of a Blockchain-Based service platform for industrial interconnection supply and demand networks.Figure 3 illustrates the comparison of Pi values for the same suppliers. The proposed system consistently exhibits higher Pi values than the base system, indicating an improvement in the parameter under study for all suppliers.

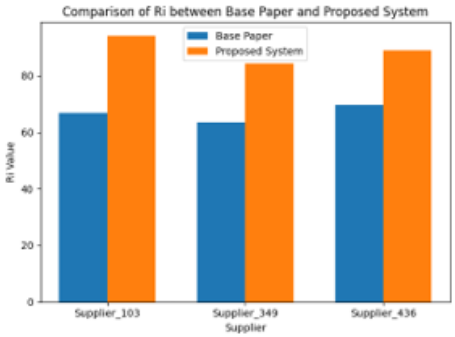

Figure 4. Comparison-Reputation Score Ri

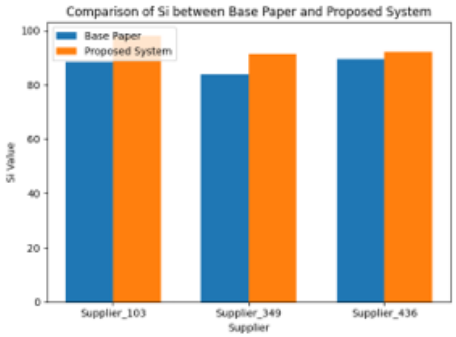

Figure 5. Comparison-Satisfaction Index Si

Figure 4 compares the Ri values of the suppliers, where the proposed system shows increased Ri values across all suppliers compared to the base system, suggesting enhanced performance in the proposed model.Figure 5 contrasts the Si values, where the base system has higher Si values across all suppliers compared to the proposed system, indicating that the proposed model reduces Si in all cases.

Table 1. Collaboration and Satisfaction Ratings Between Demanders and Suppliers

DemanderID	SupplierID	Number of Collaborations	Satisfaction Ratings
Demander_175	Supplier_101	3	[95.41, 94.63, 93.0]
Demander_842	Supplier_109	4	[96.09, 94.51, 92.89, 90.3]
Demander_254	Supplier_106	3	[90.92, 96.71, 97.63]

Table 1 provides details of collaborations between demanders and suppliers, listing the number of collaborations and satisfaction ratings for each interaction. Each row represents a unique demand-supplier pair, showing their collaboration count and corresponding satisfaction scores.

Figure 6 shows how the ARM-based reputation model influences partner selection scores for various suppliers across popular demanders. Each bar represents how well a supplier is rated by different demanders. Variability in scores highlights differences in supplier evaluations by the ARM model. Figure 7 compares the reputation scores calculated by the ARM model and traditional methods for popular suppliers. Each dot represents a score by ARM, while crosses show scores by the traditional approach. The alignment of dots and crosses indicates how closely the two methods correlate, with variations highlighting the differences in evaluation by the ARM model.

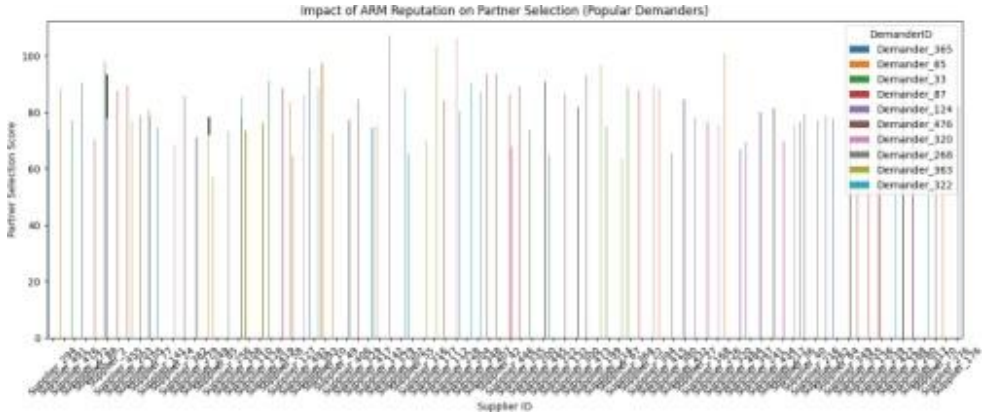

Figure 6. Impact of ARM Reputation on partner selection

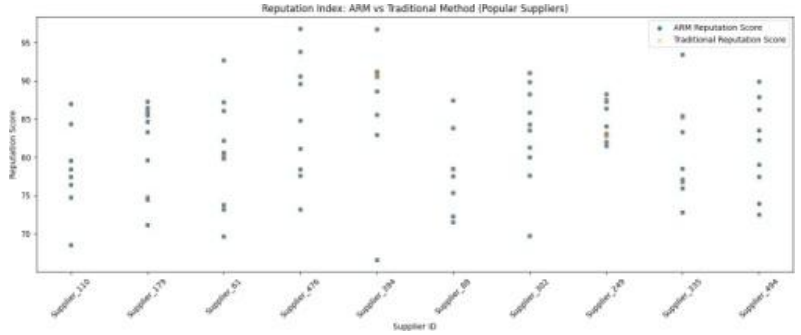

Figure 7. Reputation Index-ARm Vs Traditional Method

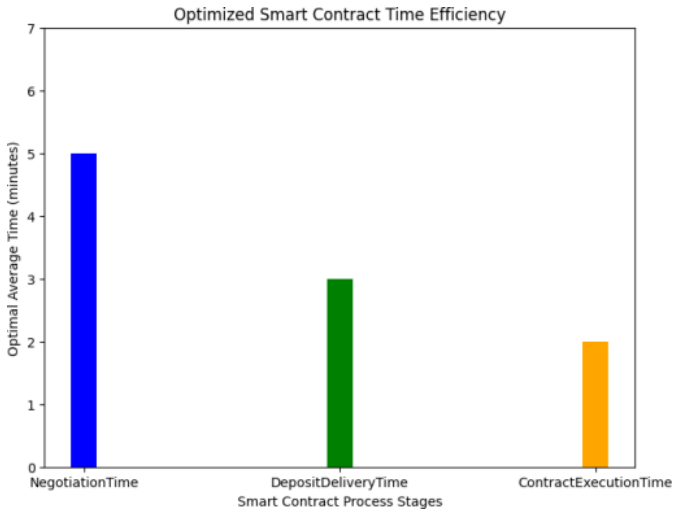

Figure 8. Optimized Time Efficiency Across Smart Contract Stages

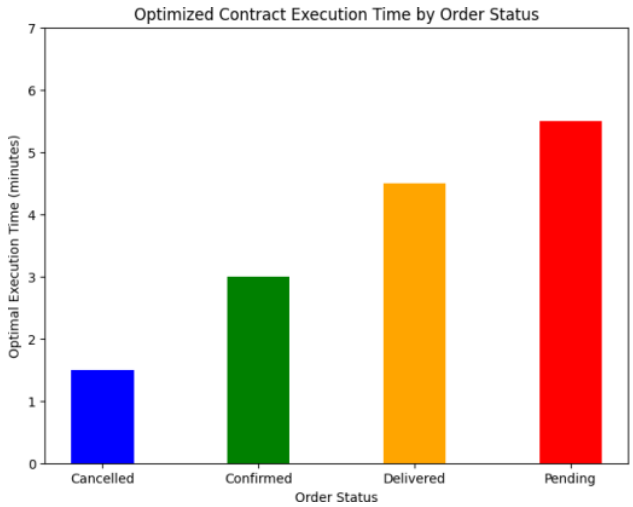

Figure 9. Optimized Contract ExecutionTime by Order Status

Figure 8 demonstrates the optimized average times for different stages of a smart contract process. The negotiation stage is the longest, taking around 5 minutes, while deposit delivery and contract execution are reduced to 3 and 2 minutes, respectively. The overall goal is to minimize time across all stages to enhance the efficiency of smart contract operations. Figure 9 displays the optimal execution times for smart contract processes based on the order status: "Cancelled," "Confirmed," "Delivered," and "Pending." The lowest execution time is for "Cancelled" orders, indicating quick handling of terminated contracts, while "Pending" orders have the longest execution time, reflecting a state of waiting or delay. This shows an efficient ordering of contract execution times, prioritizing rapid resolution where possible.

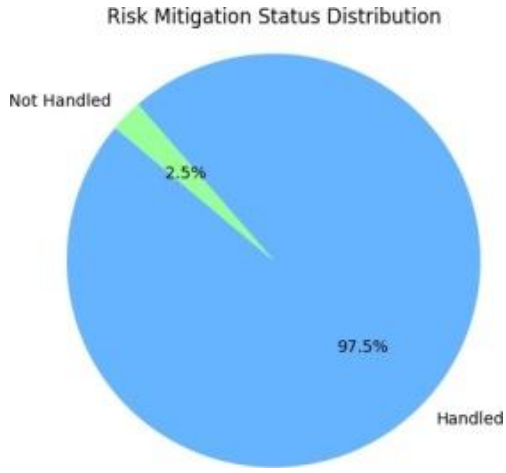

Figure 10. Risk Mitigation Status distribution

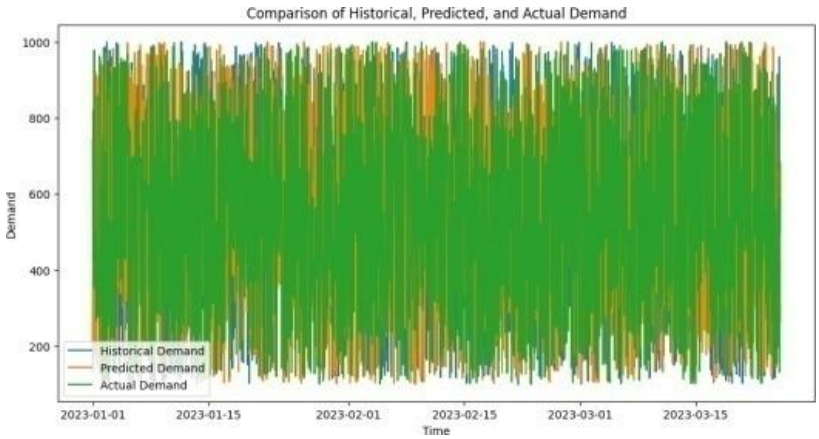

Figure 11. Comparison of historical, Predicted and actual demand

Figure 10 illustrates the distribution of risk mitigation status within the proposed system. A significant majority, 97.5%, of risks are effectively handled, indicating high efficiency in risk management. Only 2.5% of the risks remain not handled, showcasing a robust approach to mitigating potential contract-related vulnerabilities. Figure 11 presents a comparison of historical, predicted, and actual demand over time. The overlapping lines indicate how closely the predicted demand (orange) aligns with the actual demand (green), while the historical data (blue) provides a reference point. The dense and varied lines reflect frequent demand fluctuations, suggesting the prediction model's responsiveness to changes over time.

Figure 12. Effectiveness of Resource allocation based on IoT data

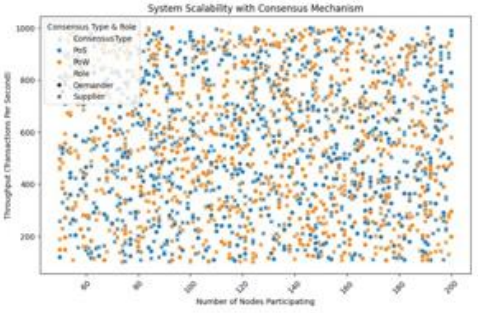

Figure 13. System scalability with consensus mechanism

Figure 12 illustrates the effectiveness of resource allocation based on IoT data, showing the relationship between resource availability and allocation accuracy. Points are differentiated by reassignment status (reassigned vs. not reassigned) and role (demander vs. supplier). Higher resource allocation accuracy is generally achieved across varying levels of resource availability, with reassigned resources and different roles clearly indicated.

Figure 13 represents the scalability of the system based on different consensus mechanisms (PoS and PoW), displaying how the throughput (transactions per second) varies with the number of participating nodes. Each point corresponds to either a demander or supplier role, and the color differentiation highlights the type of consensus mechanism. Overall, it visualizes

the performance impact of the consensus type on system scalability.

5. Discussion

The proposed Blockchain-IoT Integrated Adaptive Contracting System (B-IACS) offers a novel approach to address key challenges in supply and demand networks by enhancing transparency, security, and efficiency. By leveraging blockchain technology integrated with IoT and intelligent smart contracts, B-IACS creates an adaptive and automated framework for real-time decision-making and secure transactions. Below are the key aspects that highlight how B-IACS improves over traditional third-party platforms:

Enhanced Transparency: B-IACS ensures a higher level of data transparency while maintaining confidentiality for sensitive information. Specific data points, such as resource availability and transaction status, are made openly accessible to participants on the blockchain. This real-time transparency allows stakeholders to track the progress of supply and demand transactions and to synchronize updates efficiently. The timestamping mechanism guarantees the authenticity and immutability of data, enabling the system to maintain reliable transaction records and enhance trust across the network.

Anonymity & Privacy-Preserving Mechanisms: The proposed system integrates Attribute-Based Encryption and Access Control Lists to ensure selective data access. Personal information is kept confidential, and each transaction is conducted using anonymous addresses. Only authorized users with the appropriate keys have access to sensitive data, enhancing privacy without compromising the real-time visibility of the system' operations. This selective anonymity provides the advantage of protecting sensitive details while enabling transparent operations that are critical for secure supply and demand matching.

Advanced Security: The hybrid consensus mechanism (combining Proof of Stake and Proof of Work) fortifies security in B-IACS by ensuring robust block validation with reduced difficulty, prioritizing nodes with higher reputations, and enabling faster transaction validation. The decentralized nature of the blockchain, combined with cryptographic techniques, ensures the system remains resistant to malicious attacks and tampering. Furthermore, IoT data feeds continuously validate and secure resource delivery, ensuring the entire network's stability and resistance to security breaches.

Adaptive Intelligence in Contracts: Smart contracts in B-IACS are highly adaptive, capable of modifying their execution based on real-time IoT data. The contracts autonomously manage deposit handling, order confirmations, and payment execution, with a built-in mechanism to automatically mitigate risks, handle exceptions, and trigger appropriate penalties or rewards. This intelligent automation significantly reduces manual intervention, improves the speed and accuracy of transactions, and reduces overall costs by eliminating unnecessary steps.

Dynamic Resource Allocation & Trust Management: The Adaptive Reputation Management model is a key differentiator in B-IACS. It dynamically calculates trust scores for participants based on real-time performance data, cooperation indices, and satisfaction ratings. This dynamic trust scoring enables better partner selection and resource allocation, ensuring efficient supply and demand matching that adapts quickly to changing market conditions. Real-time IoT data plays a critical role in monitoring resource availability, reassigning roles

of suppliers and demanders, and optimizing the process for maximum efficiency.

Efficient Demand Forecasting & Resource Matching: Through the integration of Long Short-Term Memory neural networks, B-IACS achieves accurate demand and supply forecasting, improving overall supply chain performance. The system's ability to predict future market needs and dynamically adjust resource allocation based on real-time IoT data and historical trends leads to efficient order fulfillment and minimizes the risk of over- or under-supply, thus optimizing the supply chain process and reducing delays.

Scalability & Performance Optimization: The hybrid consensus mechanism not only enhances security but also enables scalability by maintaining efficient throughput and low latency in transaction processing, even as the number of network participants increases.

The combination of Proof of work and stake provides a balance between scalability and security, making the proposed system suitable for large-scale industrial applications where quick, secure, and efficient operations are essential.

6. Conclusion

Blockchain-IoT Integrated Adaptive Contracting System presents a transformative approach to enhancing efficiency, security, and scalability in e-commerce operations. By seamlessly integrating blockchain technology, IoT data streams, and adaptive intelligent mechanisms, B-IACS addresses key challenges such as real-time resource allocation, trust management, and secure transaction validation. The system's hybrid consensus mechanism effectively balances security and performance, ensuring rapid and reliable transaction processing while maintaining network integrity. The implementation of an Adaptive Reputation Management model introduces dynamic trust scoring, improving partner selection and building robust, transparent relationships within the supply-demand network. Additionally, the use of LSTM-based predictive analytics for demand forecasting significantly enhances resource allocation and supply chain optimization, reducing operational delays and improving decision-making accuracy. With its focus on smart contract automation, adaptive role assignment, and risk mitigation, B-IACS demonstrates superior performance in terms of time efficiency, trustworthiness, and resource management. This comprehensive, real-time, data-driven system serves as a scalable, secure, and intelligent solution for the evolving needs of e-commerce, offering a robust framework for efficient and sustainable industrial interconnection supply and demand networks.

7. Future enhancements

Future enhancements for B-IACS include refining the ARM model for improved trust scoring, incorporating advanced predictive analytics for better demand forecasting, and implementing enhanced security measures like homomorphic encryption. Additionally, expanding cross-chain interoperability and leveraging edge computing for real-time data processing will boost system responsiveness and scalability.

References

1. He, J., & Zhang, N. (2022). Design of a Blockchain-Based service platform for industrial interconnection supply and demand networks. Journal of Theoretical and Applied Electronic Commerce Research, 17(2), 773–788. https://doi.org/10.3390/jtaer17020040
2. Dasaklis, T. K., Voutsinas, T. G., Tsoulfas, G. T., & Casino, F. (2022). A Systematic Literature Review of Blockchain-Enabled Supply Chain Traceability Implementations. Sustainability, 14(4), 2439. https://doi.org/10.3390/su14042439
3. Aslam, J., Saleem, A., Khan, N., & Kim, Y. (2022). Blockchain Technology for Oil and Gas: Implications and Adoption Framework Using Agile and Lean Supply Chains. Processes, 10(12), 2687. https://doi.org/10.3390/pr10122687
4. Agi, M. A., & Jha, A. K. (2022). Blockchain technology in the supply chain: An integrated theoretical perspective of organizational adoption. International Journal of Production Economics, 247, 108458. https://doi.org/10.1016/j.ijpe.2022.108458
5. Chen, J., & Zheng, Y. (2022). Retraction Note: User information protection of e-commerce platform business based on credit evaluation system. Information Systems and e-Business Management, 21(S1), 73. https://doi.org/10.1007/s10257-022-00584-1
6. Manzoor, R., Sahay, B. S., & Singh, S. K. (2022). Blockchain technology in supply chain management: an organizational theoretic overview and research agenda. Annals of Operations Research. https://doi.org/10.1007/s10479-022-05069-5
7. Yuan, H., Qiu, H., Bi, Y., Chang, S., & Lam, A. (2019). RETRACTED ARTICLE: Analysis of coordination mechanism of supply chain management information system from the perspective of block chain. Information Systems and e-Business Management, 18(4), 681–703. https://doi.org/10.1007/s10257-018-0391-1
8. V. Paliwal, S. Chandra, and S. Sharma, "Blockchain Technology for Sustainable Supply Chain Management: A Systematic Literature Review and a Classification Framework," Sustainability, vol. 12, no. 18, p. 7638, Sep. 2020, doi: 10.3390/su12187638.
9. Li, J., Zhou, Z., Wu, J., Li, J., Mumtaz, S., Lin, X., Gacanin, H., & Alotaibi, S. (2019). Decentralized On-Demand Energy Supply for Blockchain in Internet of Things: A Microgrids Approach. IEEE Transactions on Computational Social Systems, 6(6), 1395–1406. https://doi.org/10.1109/tcss.2019.2917335
10. Ayan, B., Güner, E., & Son-Turan, S. (2022). Blockchain Technology and Sustainability in Supply Chains and a Closer Look at Different Industries: A Mixed Method Approach. Logistics, 6(4), 85. https://doi.org/10.3390/logistics6040085
11. G. Tortorella, F. S. Fogliatto, S. Gao, and T.-K. Chan, "Contributions of Industry 4.0 to supply chain resilience," The International Journal of Logistics Management, vol. 33, no. 2, pp. 547–566, Aug. 2021, doi: 10.1108/ijlm-12-2020-0494.
12. Bellavista, P., Esposito, C., Foschini, L., Giannelli, C., Mazzocca, N., & Montanari, R. (2021). Interoperable Blockchains for Highly-Integrated Supply Chains in Collaborative Manufacturing. Sensors, 21(15), 4955. https://doi.org/10.3390/s21154955
13. C. Magrini et al., "Using internet of things and distributed ledger technology for digital circular economy enablement: the case of electronic equipment," Sustainability, vol. 13, no. 9, p. 4982, Apr. 2021, doi: 10.3390/su13094982.
14. Khan, A. A., Laghari, A. A., Shaikh, Z. A., Dacko-Pikiewicz, Z., & Kot, S. (2022). Internet of Things (IoT) Security With Blockchain Technology: A State-of-the-Art Review. IEEE Access, 10, 122679–122695. https://doi.org/10.1109/access.2022.3223370