A Lightweight Smart Contract Blockchain Platform for Secure and Efficient SME Transaction Systems

Sabam Parjuangan*, Suhardi, I Gusti Bagus Baskara Nugraha
School of Electrical Engineering and Informatics, Institut Teknologi Bandung, Bandung, Indonesia

***Abstract*—Small and medium enterprises (SMEs) require secure, efficient, and low-cost digital transaction systems. However, many blockchain-based platforms are designed for large-scale applications and impose significant computational overhead, making them unsuitable for resource-constrained SMEs. This study proposes a lightweight smart contract blockchain platform tailored for SME-scale service environments. The system implements a modular smart contract architecture integrated with a lightweight blockchain and automates key transactional processes, including balance top-ups, service ordering, order confirmation, and payment execution, while ensuring data integrity through a simplified Proof-of-Work mechanism. System performance is evaluated using a Design of Experiment (DOE) framework with a full factorial design and analyzed through Analysis of Variance (ANOVA). The results show that execution time remains below 5 seconds under workloads of up to 20 concurrent transactions, with CPU utilization below 55%. ANOVA results indicate that transaction concurrency and smart contract complexity significantly affect performance, while block size has a limited impact. Security evaluation confirms resistance to unauthorized access, double-spending, and reentrancy attacks.**

Keywords*—Smart contract; blockchain; SME digitalization; performance evaluation; transaction systems

I. Introduction

Blockchain technology has fundamentally transformed the design of digital transaction systems by enabling decentralized, transparent, and tamper-resistant data management[1]. Since its introduction, blockchain has evolved from cryptocurrency systems into a general-purpose infrastructure applied in finance, logistics, healthcare, and service platforms[2], [3]. Smart contracts represent one of the most critical innovations within blockchain ecosystems, enabling automated execution of business logic without trusted intermediaries. Recent studies demonstrate that smart contracts improve transaction transparency, reduce operational errors, and enhance trust among distributed parties[4]. Their adoption has increased significantly due to advances in programmable blockchain platforms and digital service integration. These developments position smart contracts as a foundational component for next-generation digital transaction systems [5]–[7].

Despite these advantages, blockchain and smart contract technologies still face substantial challenges related to scalability, interoperability, performance overhead, and security vulnerabilities. Such limitations become more critical in environments with constrained computational resources and limited technical expertise. Small and medium enterprises (SMEs) are particularly affected, as many existing blockchain platforms are designed for large-scale financial or industrial deployments[8]. Empirical studies indicate that high transaction costs, complex infrastructure, and excessive resource consumption hinder blockchain adoption among SMEs. Consequently, lightweight architectures and simplified consensus mechanisms are increasingly investigated to support SME-scale operations. Addressing these constraints remains an open research challenge in applied blockchain systems [9]–[11].

The application of smart contract-based transaction systems within service-oriented SMEs offers significant potential to enhance efficiency, transparency, and cost-effectiveness. By automating processes such as balance top-ups, ordering, confirmation, and payment execution, smart contracts can reduce reliance on intermediaries and centralized payment systems. However, existing academic studies predominantly focus on large-scale or cross-organizational blockchain deployments. Empirical investigations targeting SME-scale service contexts remain limited, particularly those addressing performance trade-offs and affordability. Recent research highlights the need for blockchain solutions that explicitly consider SME operational constraints. Furthermore, structured experimental approaches such as Design of Experiment (DOE), combined with statistical analysis, remain underexplored in evaluating blockchain-based transaction systems[12], [13]. Structured experimental approaches, such as DOE combined with ANOVA, provide a systematic framework to evaluate factor interactions and optimize system performance [14].

Service-intensive sectors such as cafés provide a representative environment for studying these requirements due to their high transaction frequency and limited infrastructure capacity. Therefore, there is a need for a lightweight, efficient, and secure blockchain-based transaction platform specifically designed for SME-scale service environments and supported by rigorous experimental evaluation.

Accordingly, this study aims to design and implement a lightweight smart contract-blockchain-based transaction platform tailored for SME-scale service environments, and to evaluate its performance and security characteristics using a structured experimental methodology. The proposed system integrates modular smart contracts with a simplified Proof-of-Work consensus mechanism to ensure data integrity under low-to-moderate transaction volumes. The research emphasizes automation of core transactional workflows while maintaining acceptable performance, security, and resource efficiency, with explicit consideration of affordability and usability for non-technical SME actors.

*Corresponding author.

The main contributions of this study include the development of a modular lightweight blockchain architecture tailored for SME-scale service environments, along with a deterministic smart contract workflow that incorporates a balance-locking mechanism to prevent double spending and ensure transactional consistency. In addition, this study presents a structured performance evaluation framework based on Design of Experiment (DOE) and Analysis of Variance (ANOVA) to identify dominant factors affecting system performance under varying workloads. Furthermore, a comprehensive security validation approach is introduced to address common smart contract vulnerabilities, including unauthorized access, double spending, and reentrancy attacks in SME-scale deployments. The remainder of this study is organized as follows: Section II reviews related work, Section III presents the proposed method, Section IV discusses the results and evaluation, and Section V concludes the study.

II. Related Works

Smart contracts are self-executing programs on blockchain platforms that provide automated and deterministic execution of business logic without trusted intermediaries, enabling transparency and reduced operational cost in decentralized systems. They have been widely studied for improving transaction efficiency and trust assurance in digital ecosystems, and recent research highlights their increasing relevance in transactional and governance scenarios beyond financial services[6], [15]. Several works explore smart contract enhancements to address performance and practical deployment issues, such as execution latency and resource constraints. Research by Jiang et al. proposes lightweight consensus mechanisms to improve throughput and scalability in decentralized IoT networks[10]. Other studies investigate architectural and design strategies to maintain smart contract efficiency while upholding security assurances in distributed environments. For example, the integration of modular frameworks with smart contracts has been shown to enhance both performance and interoperability between subsystems in federated settings [16].

Security remains a central concern in smart contract development, as vulnerabilities such as reentrancy, arithmetic errors, and access control flaws can lead to critical exploits. Recent research emphasizes systematic detection and mitigation of such vulnerabilities through formal analysis and verification techniques [11], which are essential to ensure robust operation in real-world deployments. Interoperability across heterogeneous blockchain networks and external systems has also been studied, with frameworks demonstrating how smart contracts can facilitate secure interactions between modular services and permissioned chains [16]. These approaches reinforce the importance of flexible design patterns that support secure cross-chain communication and distributed consensus.

Scalability and economic feasibility constitute additional research focal points. Lightweight consensus protocols and optimized execution models have been shown to reduce transaction latency and resource consumption, making blockchain platforms more suitable for constrained environments such as edge computing and SME operations [10], [11]. In particular, federated and permissioned frameworks based on microservices and smart contracts illustrate how distributed cloud environments can leverage decentralized mechanisms for scalability without incurring prohibitive overhead.

III. Proposed Method

This study adopts a systematic empirical methodology that integrates iterative prototyping and structured experimental evaluation to develop and assess a lightweight smart contract–blockchain transaction platform tailored for service-oriented SME environments. The methodological framework combines design science principles with controlled experimentation to ensure both practical relevance and analytical rigor, as commonly applied in blockchain system engineering research [11], [17].

A. Iterative Prototyping and System Development

An iterative prototyping approach is employed to construct a functional transaction system encapsulating core SME service workflows, including balance top-up, service ordering, order confirmation, and automated payment execution. Prototyping has been widely adopted in blockchain and distributed system research to manage evolving requirements and to validate complex transactional logic before large-scale deployment[17]. The system architecture is designed modularly, integrating smart contract components with a custom lightweight blockchain based on a simplified Proof of Work consensus mechanism. The prototype is implemented using JavaScript and Node.js, with early verification conducted through unit and integration testing to ensure correctness of transaction flows, role-based access control, and data integrity.

B. Transaction Processing Workflow

Transaction execution follows a deterministic smart contract workflow that enforces security and consistency throughout the transaction lifecycle. Each user request is validated based on role authorization, followed by transaction type identification and balance verification. For payment-related transactions, a balance-locking mechanism is applied to prevent double spending during execution, a technique commonly used in secure blockchain transaction processing[7]. Validated transactions are then forwarded to the blockchain layer, where they are packaged into blocks and permanently recorded using a lightweight Proof of Work mechanism. This separation between smart contract execution and blockchain validation improves modularity and enhances system robustness[10].

To formalize the transaction execution process described above, Algorithm 1 presents a high-level pseudocode representation of the smart contract–blockchain workflow. The algorithm abstracts the essential security checks, deterministic execution steps, and blockchain recording procedures required to ensure transaction integrity, prevent double spending, and guarantee immutability within resource-constrained SME environments.

Algorithm 1 describes the deterministic execution flow of transactions within the proposed smart contract–blockchain system. The algorithm begins by validating the user's role to enforce access control and prevent unauthorized operations. For payment-related transactions, balance sufficiency is verified, and a balance-locking mechanism is applied to avoid double-spending during execution. Once validated, the corresponding smart contract logic is executed, and the transaction state is updated accordingly. Valid transactions are then added to a pending transaction pool and permanently recorded on the blockchain through block generation and validation using a lightweight Proof of Work mechanism. This workflow ensures transaction integrity, consistency, and immutability while maintaining computational feasibility for SME-scale service environments.

Algorithm 1: Transaction Processing Workflow

```
Input  : Request R, User role U
Output : Transaction result (Success / Reject / Error)

Begin
// Step 1: Authorization check
   if not Authorized(U) then
      return Reject
   else
// Step 2: Balance validation (for payment-related requests)
   if R requires payment then
     if not SufficientBalance(R) then
   return Reject
         else
            LockBalance(R)
         end if
      end if

// Step 3: Smart contract execution
    status ← ExecuteSmartContract(R)
      if status = Failed then
       UnlockBalance(R)
         return Error
      else
        UpdateTransactionState(R)
         AddTransactionToPool(R)
      end if

// Step 4: Block generation and validation
      retry ← 0
      maxRetry ← N

      while retry < maxRetry do
         blockStatus ← GenerateAndValidateBlock()

         if blockStatus = Valid then
            AppendBlockToLedger()
            return Success
         else
            retry ← retry + 1
         end if
      end while

// Step 5: Failure handling
      UnlockBalance(R)
      return Error
   end if
End
```

C. Experimental Scenario and Factors

System performance is evaluated using a Design of Experiment (DOE) framework based on a full factorial 2^3 design, which is widely used in empirical blockchain performance studies to analyze factor effects in a statistically rigorous manner [11], [18]. Three experimental factors are varied: transaction concurrency (5 and 20 concurrent transactions), smart contract logic complexity (basic top-up versus combined order and payment logic), and maximum block size (5 and 10 transactions per block). Performance metrics include transaction execution time, CPU utilization, and memory consumption. Experimental runs are conducted under controlled simulated workloads using synthetic transaction data to ensure repeatability and objective comparison.

To support the experimental evaluation, synthetic transaction data were generated to simulate realistic SME service operations. These transactions represent typical service workflows, including balance top-up, menu ordering, and payment execution, and are designed to reflect the characteristics of transaction-intensive service environments such as cafés. Each experimental run involves between 100 and 500 transactions, depending on the workload configuration defined in the DOE. The transactions are categorized into three primary types: 1) balance top-up, 2) service order submission, and 3) payment processing. The distribution of transaction types follows a uniform random pattern to ensure unbiased workload variation across experimental scenarios.

Each transaction record includes essential attributes such as transaction ID, user role, transaction type, timestamp, and payload data. The average payload size ranges between 200 and 500 bytes, depending on the transaction type and associated metadata. This configuration ensures that the generated dataset approximates realistic service transaction behavior while maintaining controlled experimental conditions. The use of synthetic data enables repeatable and controlled experimentation, allowing consistent evaluation of system performance under varying levels of transaction concurrency, smart contract complexity, and block size.

Table I presents a series of test scenarios designed to evaluate the accuracy and functional reliability of each core component within the system. These scenarios not only verify whether the implemented features operate correctly, but also assess whether the automation of business logic facilitated by the smart contracts has been executed properly, securely, and in alignment with real-world café service contexts. Table II represents a critical component within the Design of Experiment (DOE) phase of this study. Rather than reiterating the table's contents, this explanation addresses the scientific significance, rationale behind factor selection, and experimental implications that underpin the evaluation design of the smart contract-blockchain system tailored for the SME sector in café service environments.

Table III presents a structured testing plan based on the Design of Experiment (DOE) methodology using a full factorial design (2^3), which incorporates three key factors of the blockchain-smart contract system to evaluate performance efficiency and operational stability within the SME context.

TABLE I. PROTOTYPING EVALUATION SCENARIO BY COMPONENT

Component	Test Scenario	Description	Success Indicator
Top-up Balance	Valid and invalid input test	User performs top-up with a certain amount; system records it as a pending transaction	Transaction recorded with status "pending"
Top-up Confirmation	Validation by service staff	Staff confirms the user's top-up transaction	Transaction status changes to "confirmed" and enters the pool process
Menu Ordering	Order submission	User orders an item; smart contract locks the corresponding balance amount	Order received by staff interface; balance is withheld
Order Confirmation	Order validation by staff	Staff approves the order and initiates preparation	Order status changes to "preparing"
Automatic Payment	Post-service transaction	After service is completed, smart contract deducts the user's balance and records the transaction	Finalized transaction; balance reduced; block is mined
Block Creation	Hash and validation test	Transactions are packaged into a block and mined	New block added and validated by isChainValid() function
Admin Functions	CRUD and monitoring	Super admin modifies menu data and monitors transaction reports	Changes saved and report accessible
Access Security	Authorization test	Unauthorized user attempts to access staff/admin functions	Access denied with appropriate error message

TABLE II. EXPERIMENTAL FACTORS AND LEVELS

Factor	Description	Low Level	High Level
A	Number of simultaneous transactions	5 transactions	20 transactions
B	Smart contract logic complexity	Basic logic (top-up only)	Complex logic (order and payment)
C	Maximum block size	5 transactions/block	10 transactions/block

TABLE III. DESIGN OF EXPERIMENTAL SCENARIO

Combination (A,B,C)	Simulation Load	Process Evaluated	Metrics Measured
(Low, Low, Low)	5 transactions, top-up only, block size 5	Light load	Execution time, CPU, memory usage
(High, Low, Low)	20 transactions, top-up only, block size 5	High load, basic logic	Same
(Low, High, Low)	5 transactions, order + payment, block size 5	Light load, complex logic	Same
(High, High, Low)	20 transactions, order + payment, block size 5	High load and complexity	Same
(Low, Low, High)	5 transactions, top-up only, block size 10	Large block size	Same
(High, Low, High)	20 transactions, top-up only, block size 10	High load, large block	Same
(Low, High, High)	5 transactions, order + payment, block size 10	Complex logic + large block	Same
(High, High, High)	20 transactions, order + payment, block size 10	Maximum stress test	Same

D. Data Collection, Security Testing, and Analysis

During experimental execution, system responses are recorded through automated logging mechanisms. In parallel, security robustness is evaluated by simulating common smart contract threats, including unauthorized access and reentrancy scenarios, following established security evaluation practices in smart contract research [4], [19]. Collected data are analyzed using Analysis of Variance (ANOVA) to determine the statistical significance of individual factors and their interaction effects on system performance. ANOVA-based analysis enables identification of dominant performance determinants and supports system parameter optimization under SME-scale operational constraints [18], [20].

E. Experimental Environment

To ensure reproducibility and provide a clear context for performance evaluation, the experimental setup of the proposed system is defined as follows. The prototype was implemented and tested on a computing environment equipped with an Intel Core i7 processor (or equivalent AMD Ryzen processor), 16 GB of RAM, and running on the Ubuntu 22.04 LTS operating system. The system was developed using Node.js (version 18) as the primary runtime environment, supporting the execution of smart contract logic and blockchain operations.

The blockchain network was configured as a single-node simulation deployed in a local environment (localhost), which is sufficient to represent SME-scale operational conditions with limited infrastructure. All experiments were conducted under controlled conditions using synthetic transaction data generated to simulate real-world service interactions, including balance top-up, order placement, and payment execution.

System performance metrics, including execution time, CPU utilization, and memory consumption, were collected using built-in system monitoring tools and logging mechanisms integrated within the application. This controlled experimental setup ensures consistency across all Design of Experiment (DOE) scenarios and enables reliable comparison of system behavior under varying workloads and configurations.

IV. RESULT, EVALUATION, AND DISCUSSION

This section presents the experimental results of the proposed smart contract–blockchain transaction system, followed by performance evaluation and discussion of the observed behaviors under different workload scenarios. The evaluation focuses on transaction execution time, CPU utilization, and memory consumption, assessed using a full factorial Design of Experiment (DOE) to analyze the effects of transaction concurrency, smart contract logic complexity, and block size.

A. Experimental Results

The experimental results demonstrate clear performance trends across the evaluated scenarios. As summarized in Table IV, transaction execution time increases as transaction concurrency and smart contract logic complexity rise. Under low concurrency conditions, the system maintains stable execution time with minimal resource consumption. However, when concurrency is increased, execution time grows more noticeably, particularly for transactions involving combined ordering and payment logic. This behavior reflects the additional computational overhead introduced by smart contract execution and transaction state management. In contrast, variations in block size show a comparatively limited impact on execution time across most experimental combinations. Increasing the maximum number of transactions per block does not result in proportional performance improvement, indicating that block generation overhead is not the dominant factor affecting system responsiveness under SME-scale workloads. These results suggest that transaction-level processing, rather than block-level configuration, plays a more critical role in determining system performance.

TABLE IV. SUMMARY OF EXPERIMENTAL RESULT BY DOE COMBINATION

A (Load)	B (Logic)	C (Block Size)	Execution Time (s)	Block Time (s)	CPU Usage (%)	Memory Usage (MB)
Low	Low	Low	1.2	2.1	14%	85
High	Low	Low	3.7	5.9	41%	160
Low	High	Low	2.3	3.2	25%	120
High	High	Low	4.8	7.3	52%	195
Low	Low	High	1.1	1.8	13%	90
High	Low	High	3.3	4.2	38%	150
Low	High	High	2.1	2.7	24%	115
High	High	High	4.2	5.8	48%	180

Table IV presents a comprehensive summary of the experimental results obtained from the full factorial 2^3 design used to evaluate the performance of the smart contract–blockchain-based system within the context of digital café services. The eight experimental combinations are derived from systematic variations of three primary factors, namely the number of simultaneous transactions, the complexity of smart contract logic, and block size. Each combination is evaluated using four key output variables, including execution time, block mining time, CPU usage, and memory consumption, to provide a holistic assessment of system efficiency and resource utilization under different operational conditions.

1) The first experimental combination represents the baseline scenario, in which the system operates under minimal transaction load, simple contract logic, and a small block size. Under these conditions, all performance metrics demonstrate optimal results, indicating efficient execution, low resource consumption, and minimal processing overhead. This scenario serves as a reference point for comparing the impact of increasing load, logic complexity, and block size in subsequent combinations.

2) In the second combination, the transaction load is increased while the contract logic and block size remain unchanged. Despite the simplicity of the contract logic, the high number of simultaneous transactions leads to a significant increase in execution time and CPU utilization. This outcome highlights the dominant influence of transaction load on system performance, demonstrating that Factor A exerts a substantial impact even when the underlying business logic remains simple.

3) Conversely, the third combination isolates the effect of contract logic complexity by increasing logic complexity under low transaction load. The results show moderate increases in execution time and resource usage, suggesting that contract complexity influences performance, though its impact is less severe than that of transaction load.

4) The fourth combination represents the most demanding scenario under a small block configuration, combining high transaction load with complex contract logic. In this case, execution time, block mining duration, CPU usage, and memory consumption reach their highest levels. This combination effectively represents an extreme stress condition for the system and confirms a negative interaction effect between transaction load and contract complexity, particularly when block capacity is limited.

5) The fifth and sixth combinations explore the influence of increased block size under low and high transaction loads, respectively. When the block size is increased under low load and simple logic, the system exhibits slight efficiency improvements, particularly in reduced block mining time, although transaction execution time remains largely unchanged.

6) Under high transaction load, a larger block size helps reduce block mining time compared to scenarios with smaller blocks, indicating that increased block capacity can partially mitigate performance degradation. However, CPU and memory usage remain elevated, suggesting that block size alone cannot fully offset the computational demands imposed by high transaction volumes.

7) The seventh combination evaluates system behavior under complex contract logic and a larger block size without high transaction load. The results indicate that the system maintains moderate stability and efficiency under these conditions, with CPU and memory usage remaining close to baseline levels. This suggests that increased block size can effectively support more complex logic when transaction load is controlled.

8) The eighth and final combination represents the maximum stress scenario, where high transaction load, complex contract logic, and large block size are applied simultaneously. Although overall system performance degrades under these conditions, the results remain within acceptable operational thresholds, with execution time remaining below five seconds and CPU usage below 55 percent.

As illustrated in Fig. 1, transaction concurrency and smart contract logic complexity present the steepest slopes, confirming their dominant impact on execution time, whereas block size exhibits a relatively flat trend.

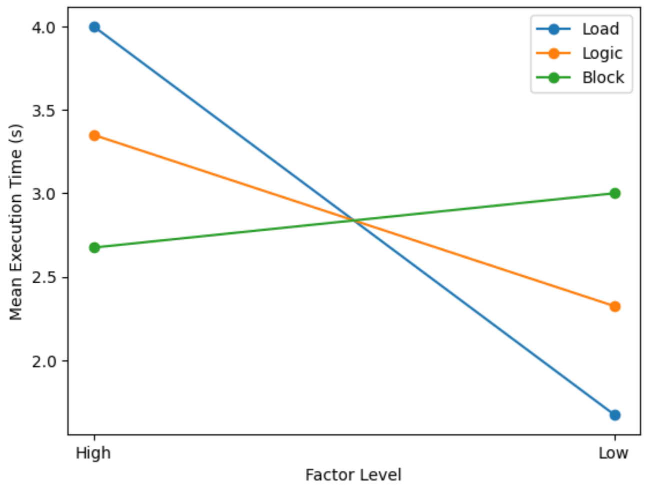

Fig. 1. Main effects plot showing the influence of transaction concurrency, smart contract logic complexity, and block size on mean transaction execution time.

These findings demonstrate that the proposed system is capable of handling demanding SME-scale operations, provided that block size and system configurations are carefully optimized to balance performance and resource consumption.

B. Performance Evaluation

CPU and memory utilization results further support the observed execution time patterns. Higher transaction concurrency leads to increased CPU usage due to parallel smart contract execution and blockchain validation processes. Memory consumption also rises moderately with increasing workload complexity, primarily due to the temporary storage of pending transactions and transaction state data. Nevertheless, resource usage remains within manageable limits, demonstrating that the proposed lightweight architecture is feasible for deployment in resource-constrained SME environments. The DOE-based evaluation confirms that transaction concurrency and smart contract logic complexity are statistically significant factors influencing system performance, while block size exhibits limited significance within the tested range. This finding indicates that optimizing concurrency handling and smart contract design is more impactful than increasing block capacity for improving system efficiency in SME-scale service contexts.

C. Security Evaluation

The security evaluation of the proposed system is conducted based on standard smart contract threat models commonly discussed in the literature, including unauthorized access, double spending, and reentrancy attacks[4], [7]. These threats represent critical vulnerabilities in blockchain-based transaction systems, particularly in resource-constrained environments where lightweight implementations are required.

The evaluation adopts a scenario-based testing approach combined with deterministic execution analysis to assess system robustness under controlled conditions. Although formal verification and static analysis tools such as Mythril and Slither are widely used for smart contract security analysis, they are not employed in this study due to the lightweight and application-oriented nature of the proposed system. Instead, the system design follows established smart contract security best practices, including role-based access control, balance-locking mechanisms, and the checks-effects-interactions pattern, which are recognized as effective countermeasures against common vulnerabilities[19]. This approach ensures that the security evaluation remains aligned with practical deployment constraints in SME environments.

Following this evaluation framework, the experimental results indicate that the proposed system successfully resists common smart contract threats, including unauthorized access and reentrancy attacks.

Security testing results indicate that the proposed system successfully resists common smart contract threats, including unauthorized access and reentrancy attacks. These outcomes are directly attributed to the deterministic transaction execution workflow and balance-locking mechanism implemented at the smart contract layer. By enforcing role-based access control and preventing concurrent balance modification during transaction execution, the system maintains transaction integrity and prevents double-spending. The security evaluation confirms that essential security requirements can be achieved without introducing excessive computational overhead.

Security and reliability are two critical aspects in smart contracts and blockchain-based systems, given their distributed, trustless, and immutable nature. The evaluation focuses on common smart contract vulnerabilities and data consistency within the blockchain to ensure that the system is not only functionally operational but also safeguarded against exploitation and dependable in the long term. Testing was conducted across three primary scenarios:

1) Unauthorized access to contract function: The objective of this security evaluation is to ensure that only entities with appropriate permissions can invoke critical smart contract functions, such as confirmTopUp(), confirmOrder(), and releasePayment(). To assess the effectiveness of the access control mechanism, a series of authorization tests was conducted by simulating unauthorized function invocations. In these tests, a customer account attempted to invoke the confirmOrder() function, which is exclusively reserved for waiter-level accounts, and a waiter account attempted to invoke the updateMenu() function, which should be restricted to the superadmin role. The system consistently detected and rejected these unauthorized access attempts by throwing exceptions or returning explicit "Unauthorized action" responses. This behavior confirms that the smart contract functions implement explicit role-based access checks, such as conditional validations comparing the caller's role against the permitted role for each function. The results of these tests demonstrate that the system enforces authorization constraints directly at the smart contract level, thereby ensuring that privilege boundaries are maintained throughout system execution. As a result, potential privilege escalation attacks are effectively mitigated at the earliest stage of interaction, reinforcing the robustness of the access control mechanism and contributing to the overall security integrity of the proposed system.

2) Double-spending prevention: The objective of this security test is to verify that no transaction can be executed more than once and that each transaction is recorded only a single time on the blockchain ledger. In particular, the evaluation focuses on preventing double-spending scenarios, such as the submission of multiple orders using the same account balance, which represent a critical risk in transactional blockchain systems. Ensuring that each transaction is uniquely processed and immutably recorded is essential for maintaining system integrity and trustworthiness. To evaluate this aspect, a series of controlled test scenarios was conducted. In these tests, a customer first performed a balance top-up and then immediately submitted two concurrent service orders in an attempt to reuse the same balance. This approach was designed to assess whether the system would allow duplicate utilization of funds under near-simultaneous transaction conditions. In parallel, the transaction data structures within the mempool and the mined blocks were inspected to detect any occurrence of duplicate transactions, specifically by examining whether identical transaction hashes were recorded more than once either within a single block or across multiple blocks.

The experimental results demonstrate that the system effectively prevents double-spending attempts. The second transaction was consistently rejected because the account balance had already been locked by the execution of the first transaction. This behavior confirms that the smart contract enforces a withholding mechanism that reserves the required balance at the moment an order is initiated, thereby preventing concurrent transactions from accessing the same funds. Furthermore, analysis of the block structure confirms that no identical transaction hashes were recorded multiple times, indicating that duplicate transactions were neither accepted into the mempool nor committed to the blockchain.

From an interpretive perspective, these findings indicate that the withholding mechanism embedded within the smart contract plays a critical role in preventing double-spending at the application logic level. In addition, the blockchain core provides an extra layer of protection by enforcing transaction uniqueness through validation mechanisms, such as the isTransactionExist() function, which ensures that no duplicate transactions are re-committed. Together, these complementary safeguards contribute to a robust defense against double-spending attacks and reinforce the reliability and security of the proposed smart contract–blockchain system.

3) Reentrancy attack detection: The objective of this security evaluation is to determine whether the smart contract is susceptible to reentrancy attacks, in which a malicious contract repeatedly invokes a vulnerable function before the internal state of the target contract is properly updated. To assess this risk, a controlled test scenario was conducted by developing a mock attacker contract designed to recursively invoke the releasePayment() function prior to the update of the customer's balance and transaction status. In addition, instruction execution order was deliberately manipulated to delay state updates until after fund transfers to observe the system's behavior under conditions commonly associated with reentrancy vulnerabilities.

The results demonstrate that the smart contract remains resilient against such attack attempts. The contract logic ensures that critical state variables, including account balance and transaction status, are updated before any external fund transfer operations are executed, in accordance with the widely accepted checks-effects-interactions design pattern. As a result, chained or recursive invocations of active and incomplete transactions are effectively disallowed. From an interpretive standpoint, these findings confirm that the system is safeguarded against reentrancy attacks due to its strict adherence to smart contract security best practices. Furthermore, the implementation of a transaction lock mechanism ensures that transactional operations cannot overlap during execution, thereby preventing concurrent state manipulation and reinforcing the overall robustness of the contract against reentrancy-based exploitation. These results demonstrate that the proposed system achieves practical security robustness through secure design patterns and controlled validation, even without the use of heavyweight formal verification tools, making it suitable for SME-scale deployment scenarios.

D. Annova Statistical Analysis

To determine which factors have a statistically significant impact on the system's output metrics, an Analysis of Variance (ANOVA) was conducted using the experimental data. Table V presents the results for the Execution Time variable. The Sum of Squares (SS) represents the total variation that can be explained by the Total Sum of Squares (SST), as expressed in SST Eq. (1).

$$SST = \sum_{i=1}^{n}(yi - \bar{y})^2 \qquad (1)$$

Where yi denotes the experimental outcome value, and $\bar{y}$ represents the overall mean. If the number of repetitions equals 5, and the difference between the average levels A high and A low compared to the overall mean results in SSA = 15.13, this reflects the proportion of variation in execution time attributable to load changes.

Degrees of freedom (df): For each factor with 2 levels, df = 1, while for the error in Eq. (2):

$$df_{error} = N - (number\ of\ combinations) \qquad (2)$$

If N=5N = 5N=5 repetitions multiplied by 8 combinations fuals 40, and there are 7 sources of variation, then the error degrees of freedom $df_{error} = N -$(number of combinations), Thus, df_{error}=40-8=32 or in the simplified model df_{error}= 8 (with 1 degree of freedom assigned to each effect).

TABLE V. ANNOVA SUMMARY FOR EXECUTION TIME

Source of Variation	SS (Sum of Squares)	df	MS (Mean Square)	F-Value	p-Value
Factor A (Load)	15.13	1	15.13	42.9	0.0004
Factor B (Logic)	5.01	1	5.01	14.2	0.0061
Factor C (Block Size)	1.12	1	1.12	3.2	0.1087
A×B Interaction	3.97	1	3.97	11.7	0.0103
A×C Interaction	0.58	1	0.58	1.7	0.2219
B×C Interaction	0.42	1	0.42	1.3	0.2782
Error	1.77	8	0.22	—	—
Total	28.00	15	—	—	—

The analysis of variance (ANOVA) results demonstrate that transaction load (Factor A) exerts the most substantial influence on system execution time. This is supported by a sum of squares (SS) value of 15.13, an F-statistic of 42.9, and a p-value of 0.0004, indicating a highly significant effect. These results clearly show that the system's performance is highly sensitive to the number of concurrent transactions, positioning user throughput as the primary determinant of execution efficiency. Smart contract logic complexity (Factor B) also has a statistically significant effect on execution time, as reflected by an SS value of 5.01, an F-statistic of 14.2, and a p-value of 0.0061. The observed increase in execution time under more complex contract logic, such as combined ordering and payment processes, can be attributed to additional computational overhead arising from multiple state transitions, validation steps, and contract interactions during execution.

Conversely, block size (Factor C) does not exhibit a statistically significant impact on execution time, with an SS value of 1.12, an F-statistic of 3.2, and a p-value of 0.1087. While larger block sizes may contribute to improvements in block mining efficiency, their effect on overall transaction execution duration remains relatively marginal. Further analysis of interaction effects reveals that the combined influence of high transaction load and complex contract logic (A × B interaction) is statistically significant, as indicated by an SS value of 3.97, an F-statistic of 11.7, and a p-value of 0.0103. This finding suggests that system performance degradation is amplified when high load conditions coincide with complex contract logic, resulting in a compounded negative effect that exceeds the individual contributions of each factor.

In contrast, the interaction between transaction load and block size (A × C) does not reach statistical significance, with an SS value of 0.58, an F-statistic of 1.7, and a p-value of 0.2219, indicating that increasing block size neither sufficiently mitigates nor exacerbates the performance impact of high transaction volumes. Similarly, the interaction between contract logic complexity and block size (B × C) is not statistically significant, as evidenced by an SS value of 0.42, an F-statistic of 1.3, and a p-value of 0.2782, suggesting that larger block sizes do not substantially intensify the execution overhead introduced by more complex smart contract logic. The residual error term accounts for a relatively small portion of the total variation, with an SS value of 1.77 out of 28, indicating that the DOE model explains the majority of observed variance in execution time. This is further confirmed by a high coefficient of determination (R^2), reflecting the robustness and explanatory strength of the experimental model.

From a system design perspective, these findings imply that optimization efforts should primarily focus on managing transaction concurrency and simplifying smart contract logic to enhance execution efficiency, particularly under high-load conditions. Strategies such as transaction batching, load-aware execution scheduling, and modular contract decomposition can be employed to mitigate the performance impact of concurrent transactions and complex logic. Although block size optimization plays a secondary role, it can still be leveraged to improve mining efficiency and overall system throughput when appropriately configured. Collectively, the ANOVA results provide actionable insights for designing scalable and resource-efficient smart contract-blockchain systems tailored to the operational constraints of SME-scale service environments.

E. Comparative Analysis with Existing System

To further evaluate the effectiveness of the proposed system, a comparative analysis is conducted against representative blockchain-based transaction platforms reported in the literature. The comparison focuses on key aspects, including consensus mechanism, target deployment scale, execution performance, resource utilization, security capabilities, and system complexity. This analysis aims to position the proposed system within the broader landscape of blockchain solutions and to highlight its suitability for SME-scale service environments.

As shown in Table VI, the proposed system demonstrates a favorable balance between performance efficiency and security for SME-scale applications. Compared to general-purpose blockchain platforms such as Ethereum, it achieves lower execution time and reduced computational overhead, making it more suitable for resource-constrained environments.

In comparison with enterprise-oriented frameworks such as Hyperledger Fabric, which provide strong modular security features, the proposed approach offers comparable execution performance while maintaining lower system complexity and infrastructure requirements. Consequently, it is better suited for small-scale service deployments that do not require full enterprise-level capabilities.

Furthermore, compared to existing lightweight blockchain solutions designed for IoT or edge environments[23], the proposed system explicitly integrates smart contract security mechanisms, including role-based access control, balance-locking, and reentrancy protection. These features enhance transaction integrity without introducing excessive computational overhead.

Overall, the comparison indicates that the proposed approach is specifically optimized for SME-scale service environments, where moderate transaction throughput, low operational cost, and practical security guarantees are critical requirements. The performance ranges reported for existing systems are derived from representative studies in the literature and may vary depending on network configuration, workload characteristics, and deployment environments.

TABLE VI. COMPARISON WITH EXISTING BLOCKCHAIN-BASED TRANSACTION SYSTEMS

System/Approach	Consensus Mechanism	Target Scale	Execution Time	Resource Usage	Security Features	Complexity
Proposed System	Lightweight PoW	SME-scale	2-5 seconds (measured)	Low–Moderate	Access control, double-spending prevention, reentrancy protection	Low
Ethereum-based System	Proof of Work / PoS	Large-scale	10-20 seconds[21]	High	Strong security, but high computational overhead	High
Hyperledger Fabric	PBFT / Raft	Enterprise-scale	5-12 seconds[22]	Moderate–High	Access control, modular security policies	Moderate–Hig
IoT Lightweight Blockchain	Lightweight Consensus	IoT / Edge	3–8 seconds[23]	Low	Limited smart contract security features	Moderate

F. Discussion

The experimental findings highlight several important implications for blockchain adoption in SME service environments. First, the dominant impact of transaction concurrency over block size suggests that SME-scale systems benefit more from efficient transaction management than from large block configurations typically used in high-throughput blockchain platforms. This observation aligns with the operational characteristics of SMEs, where transaction volumes are moderate, and responsiveness is a critical requirement. Second, the limited performance impact of block size indicates that lightweight blockchain configurations are sufficient for service-oriented SMEs, reducing unnecessary computational and energy overhead. Compared to general-purpose blockchain platforms designed for large-scale financial ecosystems, the proposed approach demonstrates that tailored blockchain architectures can better balance efficiency, security, and affordability for SMEs. Finally, the integration of smart contract automation with a lightweight blockchain shows that secure and deterministic transaction processing can be achieved without complex infrastructure. These results support the feasibility of adopting blockchain-based transaction systems in small-scale service businesses, such as cafés, and provide empirical evidence that lightweight blockchain designs are more suitable for SME environments than conventional heavyweight platforms.

The experimental results reported in Section IV demonstrate that transaction concurrency and smart contract logic complexity are the most influential factors affecting transaction execution time and system resource utilization. Across all evaluated scenarios, higher levels of concurrent transactions combined with more complex contract logic consistently resulted in increased execution latency and CPU usage. This behavior is consistent with prior studies showing that parallel smart contract execution introduces contention over shared states and increases scheduling overhead, particularly in systems that enforce deterministic execution to maintain ledger consistency[24], [25]. The results confirm that, in SME-scale deployments, transaction-level processing dominates performance behavior.

The finding that block size exhibits a limited impact on execution time within the tested range can be explained by the operational characteristics of SME service environments. Unlike public blockchains with sustained high transaction throughput, the evaluated system processes low-to-moderate transaction volumes where block formation and propagation are not performance bottlenecks. Similar observations have been reported in analytical and experimental studies on Proof-of-Work systems, which show that increasing block size yields diminishing performance gains under moderate workloads while introducing additional risks such as orphaned blocks and inefficient resource usage[26]. The ANOVA results in this study, where block size does not reach statistical significance, are therefore consistent with established blockchain performance models.

Smart contract logic complexity further contributes to increased execution time due to additional conditional branches, state transitions, and storage operations executed during each transaction. Prior research on concurrent smart contract execution highlights that complex contracts amplify computational overhead and synchronization costs, particularly under concurrent workloads[24]. Nevertheless, the measured CPU and memory usage in this study remain within acceptable limits, indicating that modular contract design combined with a lightweight blockchain architecture can effectively support service-oriented SME applications without excessive resource consumption.

From a security perspective, the evaluation results confirm that the proposed system effectively mitigates common smart contract threats, including unauthorized access, double spending, and reentrancy attacks. The enforced balance-locking mechanism and deterministic transaction state transitions play a central role in preventing inconsistent state updates during concurrent execution. Recent studies on reentrancy vulnerability detection emphasize that defensive design patterns such as balance locking and strict execution ordering are among the most effective practical countermeasures, particularly in resource-constrained environments where heavyweight formal verification may not be feasible[27]. The absence of detected vulnerabilities in the tested scenarios supports the robustness of the adopted design approach.

The choice of a lightweight Proof-of-Work–based blockchain is further justified by recent research on consensus mechanisms tailored for IoT and edge environments, which demonstrates that simplified or hybrid consensus protocols can significantly reduce computational overhead while maintaining sufficient security for localized deployments[28]. In the context of SME café services, where trust boundaries are relatively limited and transaction volumes are moderate, such lightweight consensus designs represent a practical trade-off between security, performance, and cost.

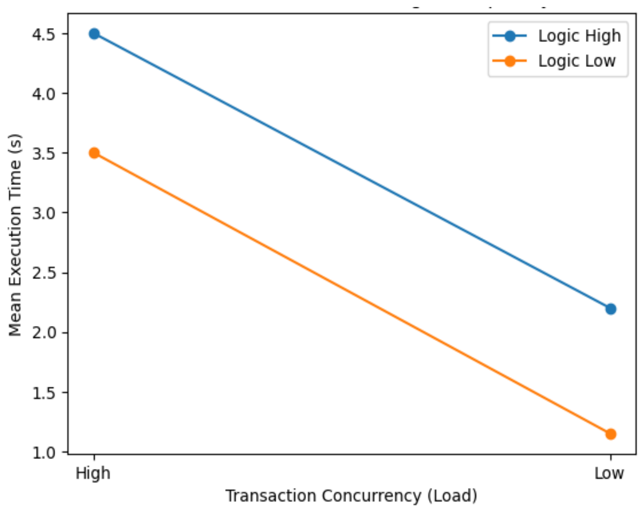

Fig. 2. Interaction plot between transaction concurrency and smart contract logic complexity.

Fig. 1 and 2 jointly provide a clear visual explanation of the performance behavior observed in the experimental results. As shown in Fig. 1, transaction concurrency exhibits the strongest main effect on execution time, followed by smart contract logic complexity, while block size demonstrates only a marginal influence within the evaluated range. This confirms the quantitative findings from the DOE and ANOVA analysis, where concurrency and logic complexity were identified as statistically significant factors. Fig. 2 further reveals a pronounced interaction between transaction concurrency and smart contract logic complexity, indicating that the impact of complex contract logic is amplified under high concurrent workloads. This non-linear behavior explains the sharp increase in execution time observed in high-load scenarios and highlights that transaction-level processing, rather than block-level configuration, constitutes the primary performance bottleneck in the proposed system. Together, these visual results reinforce the conclusion that optimizing concurrency handling and smart contract execution is more critical than increasing block capacity for achieving efficient blockchain-based transaction processing in SME-scale service environments.

Overall, the findings of this study align with contemporary blockchain research and provide empirical evidence that blockchain-based transaction systems can be effectively adapted to SME-scale service environments when architectural complexity is carefully constrained. The results suggest that optimization efforts should prioritize efficient handling of transaction concurrency and smart contract execution rather than increasing block capacity. These insights reinforce the importance of designing blockchain systems that are tailored to specific operational contexts instead of directly adopting architectures optimized for large-scale financial ecosystems.

V. Conclusion

The findings of this study contribute important insights into the applicability of smart contracts and blockchain technologies as scalable digital transaction solutions for small and medium-sized enterprises (SMEs) in service-oriented contexts. The evaluation demonstrates that a modular smart contract–blockchain architecture can deliver automation, transparency, and reliable performance under varying operational conditions, which addresses key transactional challenges commonly faced by SMEs. The results also highlight that system performance is strongly influenced by transactional concurrency and logic complexity, underscoring the need for careful architectural design when adopting decentralized solutions in high-throughput environments.

Security validation confirms that the proposed system incorporates fundamental safeguards against prevalent vulnerabilities, reinforcing the practical relevance of disciplined smart contract engineering in distributed systems. These outcomes extend existing knowledge by showing that blockchain–based solutions are not limited to large-scale industries or financial infrastructures, but can also be adapted as efficient and secure frameworks for localized transactional ecosystems.

Several limitations of this study should be acknowledged. The prototype was tested within a controlled environment, and real-world factors such as heterogeneous hardware and network variability remain unexamined. The use of a simplified consensus mechanism provides initial validation but does not fully address scalability and energy efficiency in operational deployments. Additionally, broader security threat modeling and empirical usability evaluation with actual SME operators are necessary to strengthen the evidence for widespread adoption.

Future research should therefore focus on real-world pilot implementations, exploration of alternative consensus protocols that enhance performance and sustainability, and integration with external systems such as point-of-sale devices and digital wallets to improve interoperability. Formal security verification and multi-node decentralization studies will further advance the maturity of the proposed approach. By pursuing these directions, the work presented in this study may support the development of robust and context-aware blockchain-enabled service systems suitable for the evolving needs of SMEs.

References

[1] Y. I. Alzoubi and A. Mishra, “Green blockchain – A move towards sustainability,” J. Clean. Prod., vol. 430, p. 139541, 2023, doi: https://doi.org/10.1016/j.jclepro.2023.139541.

[2] A. Datta, D. Reijsbergen, and J. Zhang, “BlockChain I / O: Enabling Cross-Chain Commerce,” IEEE Access, no. June, pp. 90915–90928, 2024.

[3] C. Li, S. Liang, J. Zhang, Q. Wang, and Y. Luo, “Blockchain-based Data Trading in Edge-cloud Computing Environment,” Inf. Process. Manag., vol. 59, no. 1, p. 102786, 2022, doi: https://doi.org/10.1016/j.ipm.2021.102786.

[4] L. Zhang et al., “A Novel Smart Contract Reentrancy Vulnerability Detection Model based on BiGAS,” J. Signal Process. Syst., vol. 96, no. 3, pp. 215–237, 2024, doi: 10.1007/s11265-023-01859-7.

[5] C. Fang, N. Ullah, M. Batumalay, W. M. Al-Rahmi, and F. Alblehai, “Blockchain technology and its impact on sustainable supply chain management in SMEs,” PeerJ Comput. Sci., vol. 11, pp. 1–30, 2025, doi: 10.7717/peerj-cs.2466.

[6] W. Müller, “Determinants of smart contract adoption in supply chains: a UTAUT-based PLS-SEM analysis,” Oper. Manag. Res., vol. 18, no. 4, pp. 1113–1124, 2025, doi: 10.1007/s12063-025-00560-1.

[7] J. Guo, L. Lu, and J. Li, “Smart Contract Vulnerability Detection Based on Multi-Scale Encoders,” Electronics, vol. 13, no. 3. p. 489, 2024. doi: 10.3390/electronics13030489.

[8] Y. Wang, Y. Wang, and Y. Zhang, “Application of the Blockchain Technology in the Vertical Value Chain Management of Enterprises,” Wirel. Commun. Mob. Comput., vol. 2022, 2022, doi: 10.1155/2022/2408027.

[9] J. Kumar, G. Rani, M. Rani, and V. Rani, “Blockchain technology adoption and its impact on SME performance: insights for entrepreneurs and policymakers,” J. Enterprising Communities, vol. 18, no. 5, pp. 1147–1169, 2024, doi: 10.1108/JEC-02-2024-0034.

[10] P. Jiang, L. Shi, B. Cao, T. Wang, B. Ji, and J. Li, “Proof-of-trusted-work: A lightweight blockchain consensus for decentralized IoT networks,” Digit. Commun. Networks, vol. 11, no. 4, pp. 1055–1066, 2025, doi: https://doi.org/10.1016/j.dcan.2024.10.011.

[11] E. U. Haque et al., “Performance enhancement in blockchain based IoT data sharing using lightweight consensus algorithm,” Sci. Rep., vol. 14, no. 1, p. 26561, 2024, doi: 10.1038/s41598-024-77706-x.

[12] H. Saed, Y. Snene Manzli, A. Jeribi, and H. Alnafisah, “Analyzing blockchain adoption in management operations through an extended UTAUT model in the Libyan context,” Sci. Rep., vol. 15, no. 1, pp. 1–19, 2025, doi: 10.1038/s41598-025-12608-0.

[13] J. J. de Leon, C. Zhang, C.-S. Koulouris, F. Medda, and Rahul, “Smart Contract Security in Decentralized Finance: Enhancing Vulnerability Detection with Reinforcement Learning,” Applied Sciences, vol. 15, no. 11. p. 5924, 2025. doi: 10.3390/app15115924.

[14] Y. Kodali and Y. V. P. Kumar, “ANOVA-Based Variance Analysis in Smart Home Energy Consumption Data Using a Case Study of Darmstadt Smart City, Germany†,” Eng. Proc., vol. 82, no. 1, pp. 1–8, 2024, doi: 10.3390/ecsa-11-20354.

[15] I. Mustafa, A. McGibney, and S. Rea, “Smart contract life-cycle management: an engineering framework for the generation of robust and verifiable smart contracts,” Front. Blockchain, vol. 6, no. January, pp. 1–19, 2023, doi: 10.3389/fbloc.2023.1276233.

[16] K. S. Alshudukhi, M. A. Khemakhem, F. E. Eassa, and K. M. Jambi, “An Interoperable Blockchain Security Frameworks Based on Microservices and Smart Contract in IoT Environment,” Electronics, vol. 12, no. 3. p. 776, 2023. doi: 10.3390/electronics12030776.

[17] A. T. Setiawan and D. Hindarto, “Development of a Prototype for a Product Recommendation System Using Blockchain Technology,” Int. J. Softw. Eng. Comput. Sci., vol. 5, no. 2, pp. 484–497, 2025, doi: 10.35870/ijsecs.v5i2.3855.

[18] H. Honar Pajooh, M. A. Rashid, F. Alam, and S. Demidenko, “Experimental Performance Analysis of a Scalable Distributed Hyperledger Fabric for a Large-Scale IoT Testbed,” Sensors, vol. 22, no. 13. p. 4868, 2022. doi: 10.3390/s22134868.

[19] R. Yu, Y. Zhang, Y. Wang, and C. Liu, “TxMirror: When the Dynamic EVM Stack Meets Transactions for Smart Contract Vulnerability Detection,” Symmetry, vol. 15, no. 7. p. 1345, 2023. doi: 10.3390/sym15071345.

[20] C. Melo, G. Gonçalves, F. A. Silva, and A. Soares, “A comprehensive hyperledger fabric performance evaluation based on resources capacity planning,” Cluster Comput., vol. 27, no. 9, pp. 12395–12410, 2024, doi: 10.1007/s10586-024-04591-4.

[21] S. Parjuangan and Suhardi, “Systematic Literature Review of Blockchain based Smart Contracts Platforms,” pp. 381–386, 2020, doi: 10.1109/icitsi50517.2020.9264908.

[22] Q. Nasir, I. A. Qasse, M. Abu Talib, and A. B. Nassif, “Performance analysis of hyperledger fabric platforms,” Secur. Commun. Networks, vol. 2018, 2018, doi: 10.1155/2018/3976093.

[23] E. Bandara, D. Tosh, P. Foytik, and S. Shetty, “Tikiri - Towards a Lightweight Blockchain for IoT,” Futur. Gener. Comput. Syst., no. 119, pp. 1–31, 2021.

[24] C. Jin, S. Pang, X. Qi, Z. Zhang, and A. Zhou, “A High Performance Concurrency Protocol for Smart Contracts of Permissioned Blockchain,” IEEE Trans. Knowl. Data Eng., vol. 34, no. 11, pp. 5070–5083, 2022, doi: 10.1109/TKDE.2021.3059959.

[25] W. Yang et al., “Adaptive Parallel Scheduling Scheme for Smart Contract,” Mathematics, vol. 12, no. 9. p. 1347, 2024. doi: 10.3390/math12091347.

[26] J. Chen, Y. Cheng, Z. Xu, and Y. Cao, “Decision on block size in blockchain systems by evolutionary equilibrium analysis,” Theor. Comput. Sci., vol. 942, pp. 93–106, 2023, doi: https://doi.org/10.1016/j.tcs.2022.11.026.

[27] L. Guo, H. Huang, L. Zhao, P. Wang, S. Jiang, and C. Su, “Reentrancy vulnerability detection based on graph convolutional networks and expert patterns under subspace mapping,” Comput. Secur., vol. 142, p. 103894, 2024, doi: https://doi.org/10.1016/j.cose.2024.103894.

[28] Z. Nie, M. Zhang, and Y. Lu, “HPoC: A Lightweight Blockchain Consensus Design for the IoT,” Applied Sciences, vol. 12, no. 24. p. 12866, 2022. doi: 10.3390/app122412866.