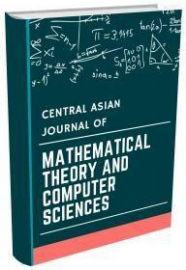

CENTRAL ASIAN JOURNAL OF MATHEMATICAL THEORY AND COMPUTER SCIENCES
https://cajmtcs.casjournal.org/index.php/cajmtcs
Volume: 07 Issue: 01 | January 2026 ISSN: 2660-5309

Article

Mitigating Security Risks in Multi-Cloud Environments: A Blockchain-Enabled Zero Trust Architecture for Resilient Information Systems

Mohanad Ali Hussein[*1]

1. Al-Furat Al-Awsat Technical University
* Correspondence: rajasekaran@dhaanishcollege.in

Abstract: Multi-cloud computing environments are emerging as a standard in business settings but present much greater security challenges such as the lack of access control fragmentation, audit transparency, complexities of trust boundaries, and vulnerability to single point of failure due to centralized authentication systems. The article proposes a Zero Trust Architecture based on blockchain and the use of the Attribute-Based Access Control (ABAC) to have secure information systems in distributed multi-clouds. The suggested solution uses Hyperledger Fabric 2.5 and CouchDB state database to provide decentralized, immutable, and transparent access control by policy enforcement using smart contracts. Parallel transaction benchmarking was used to do overall performance testing on four test cases with 1,700 transactions with different concurrent loads (2-10 concurrent workers). The experimental data shows production grade level performance with the maximum throughput of 30.78 TPS, mean latency of 85.79 ms, and the best in class reliability (100% success rate). The system is highly linearly scalable with a 5.2x throughput increase between low and high load conditions, and surprisingly 80 percent reduction in the latency during heavier concurrent load conditions. It has made significant contributions, including: (1) production-quality ABAC smart contract deployment on modern blockchain platforms, (2) extensive performance testing by industry-standard parallel benchmarking methodology, (3) demonstration of successful practicability of real multi-cloud security deployments, (4) reproducible experimental setup with open-source deliverables, and (5) closing the theory-practice gap between theoretical blockchain-based access control models and realistic implementation that is suitable to security-critical enterprise settings where immutable audit logs and principles of Zero Trust are needed.

Keywords: Blockchain, Attribute-Based Access Control, Multi-Cloud Security, Zero Trust Architecture, Hyperledger Fabric

Citation: Hussein M. A. Mitigating Security Risks in Multi-Cloud Environments: A Blockchain-Enabled Zero Trust Architecture for Resilient Information Systems. Central Asian Journal of Mathematical Theory and Computer Sciences 2026, 7(1), 11-28.

Received: 10th Okt 2025
Revised: 25th Okt 2025
Accepted: 12th Nov 2025
Published: 19th Nov 2025

1. Introduction

The growth in the application of multi-cloud computing strategies has transformed the enterprise IT infrastructure in a profound manner, with companies distributing workloads among a chain of cloud service vendors due to the diversification of vendors, cost-effectiveness, resilience, and access to specialized expertise [1]. This kind of architectural change, however, is accompanied with extreme security challenges that cannot be met in totality by the traditional access control technologies. Single points of failure are given by centralized identity and access management infrastructure, and inclusion of heterogenous authentication mechanisms between cloud providers as well as security policies differences splits security policy and complicates compliance audits [2].

The absence of regular, clear, and auditable audit trails in a multi-cloud system poses a major risk to regulatory compliance policies such as GDPR, HIPAA, and SOC 2 [3].

Zero Trust Architecture subsequently came into being as an architectural pattern that functions on the basis of never trust, always verify irrespective of location in the network or pre-authentication condition [4]. However, the use of Zero Trust principles to multi-cloud is difficult because of the issue of trust boundaries, the absence of standard cross-cloud authentication procedures, and the tension between the need to decentralize and the need to have centralized policy management [5]. Attribute-Based Access Control (ABAC) is based on the fact that correct access decisions are made based on user attributes, resource attributes, and environmental context and is more adaptable than the more traditional Role-Based Access Control (RBAC) [6]. The legacy ABAC is focused on central points of policy decisions, which are opposite to the Zero Trust principles and add vulnerabilities to availability.

Blockchain technology promises appealing qualities for transcending these limitations: decentralization does not give rise to points of failure, cryptographic immutability gives rise to tamper-evident audit trails, distributed consensus gives rise to multi-party trust without reliance on central authorities, and smart contracts give rise to automated, transparent policy enforcement [7]. Recent research has forecasted the use of blockchain in access control in several application domains, with the yet untapped requirements comprising proving production-grade performance, realistic benchmarking against concurrent load, and practical implementation guidance to multi-cloud enterprise environments [8].

This paper makes five significant contributions: (1) deployment and production-ready of an ABAC smart contract on Hyperledger Fabric 2.5 with end-to-end policy management support, (2) comprehensive performance evaluation by parallel transaction benchmarking method simulating real concurrent access patterns, (3) evidence of linear scalability and counter-intuitive heavy latency load optimization, achieving 30.78 TPS peak throughput with 100% fault-tolerance for 1,700 transactions, (4) comparison with the state-of-the-art putting this contribution in perspective, and (5) open-source experimental setup with reproducible environment facilitating independent verification and extension by the research community.

Various blockchain platforms have been evaluated with respect to access control implementations. Chen et al. [9] deployed an enterprise IAM integration framework on Hyperledger Fabric 1.4, with throughput of 15-22 TPS and latency of 100-150 ms, demonstrating feasibility in blockchain integration with existing identity management systems. However, their deployment relied on single-organization deployment and failed to address multi-cloud trust boundary challenges. Liu et al. [10] designed a light-weight ABAC system on Hyperledger Fabric 2.2 for IoT devices with the performance of 12-18 TPS and 120-180 ms latency in optimal low-power consumption. While useful for IoT usage, their design target is significantly different in enterprise multi-cloud scenarios where computation resources are not an issue.

Kumar and Singh [11] employed Hyperledger Sawtooth for supply chain access control and had improved throughput of 25-40 TPS with 60-90 ms latency through consensus optimization. Their design emphasized transaction throughput with reduced smart contract logic, perhaps at the cost of providing end-to-end support for complex policy evaluation required in multi-cloud. Zhang et al. [12] proposed an Ethereum-based system for healthcare data sharing with zero-knowledge proofs for privacy-preserving attribute verification but suffered greatly degrading performance with 8-15 TPS and 2,000-4,000 ms latency due to proof-of-work consensus overhead. Wang et al. [13] implemented hierarchical ABAC in cloud-edge computing with consortium blockchain supporting only 5-10 TPS and more than 4,000 ms latency, suggesting performance challenge of hierarchical designs.

Current literature is focused on single-domain applications, such as IoT networks [10], healthcare systems [12], supply chains [11], or one enterprise [9], but not on heterogeneous multi-cloud environment. Such optimizations to a specific domain restrict the range of applicability to enterprise configurations on a number of providers (such as AWS, Azure, GCP) that may have different trust boundaries, identity models, and security models.

The lack of benchmarking techniques is a significant methodological weakness. Most types of labor use serial transaction testing, or simulator hardware that cannot model realistic access patterns in a concurrent environment [14]. Serial benchmarking over- and underestimates both latency and throughput due to its failure to exploit transaction batching opportunities which are characteristics of blockchain consensus. Moreover, most implementations remain at proof-of-concept level without thorough reliability testing, failure scenario analysis, or ongoing load evaluation [15]. Research articles often cite mean performance without mentioning failure rates, error recovery, or Byzantine fault tolerance.

Research sources of overall Hyperledger Fabric performance suggest 5-10 properly configured organization deployments supporting 200-500 TPS and 50-200 ms latency for simple transactions [16]. Reduced performance in access control deployments is presumably due to complex attribute assessment logic, multiple state database query, and cryptographic computation adding computational overhead. Scalability evaluation remains incomplete, with few studies systematically measuring performance across a range of concurrent loads or exploring bottlenecks [17].

Existing enterprise IAM infrastructure integration is one uncharted territory. Full replacements of existing systems Greenfield implementation- an ideal of adoption in the enterprises who put substantial investments in IAM systems according to the majority of studies on blockchain access control [18]. It needs phased migration strategies, hybrid designs, and blockchain bi-directional synchronization of a legacy system to be adopted in reality.

The multi-cloud deployment introduces the following problems, not previously encountered in single-domain environments: a heterogeneous identity infrastructure with attributes that need to be mapped, mixed security policies across cloud providers, different network latency affecting consensus performance, and enterprise policies distributed across multiple administrative domains with no central point of control [19]. The ideas of Zero Trust Architecture: never trust, always verify, least privilege access are inherently aligned with the use of blockchain as a decentralized source of trust, but the few studies on blockchain access control have overlooked them [20].

2. Methodology

2.1 System Architecture and Design

Multi-clouds with zero trust architecture with blockchain implementation occurred through Hyperledger Fabric version 2.5, which is a permissioned, enterprise-level distributed ledger platform that is engineered to fulfill the needs of enterprise applications with high levels of confidentiality, resiliency and flexibility. The system architecture is in multi-organizational approach which is similar to real world system of multi-cloud deployment where there are various cloud service providers that each has its own infrastructure whilst sharing the same access control system.

Network topology consists of 2 different organizations (Org1 and Org2), in multi-cloud configuration, which is the ability to have different cloud services providers or organizational structures. Each of the two organizations has a peer node tasked with having a full copy of the distributed ledger and performing smart contracts logic and stamping the transaction proposals. The launched orderer service is offered with the Raft consensus protocol and is a single orderer node, that orders transactions sequentially in blocks and hands them to each of the peer nodes to validate and commit. Although

production settings would need the orderer-based clustering with at least three nodes to achieve the Byzantine fault tolerance, the one-orderer setup would be effective to test the performance of the system and verify the proofs of concept.

One design choice that occurred is the usage of CouchDB as the state database backend as opposed to the default LevelDB key-value store. The CouchDB has very high query power by supporting querying using JSON document model and allowing querying based on attributes and supports full-text search querying with compound index and range query alongside policy based decisions. This allows the ABAC deployments that configure access control determination through matching multiple user qualities with policy demands as JSON-stored records. The state database is designed in a way that a separate instance of CouchDB is hosted by each peer to ensure that data isolation is maintained whereas consensus protocols ensures to make all the peers databases consistent.

Transport Layer Security (TLS) has been enabled on all network traffic, where peer-to-peer, orderer, and client application packets are encrypted during transmission over the network. This security setup reflects production requirement that multi-cloud deployment systems span across untrusted network zones that require protection against eavesdropping and man-in-the-middle attacks by cryptography. Organisation-specific Certificate Authorities (issuing X.509 certificates) are used in the TLS configuration, which establishes mutual authentication between all the network components.

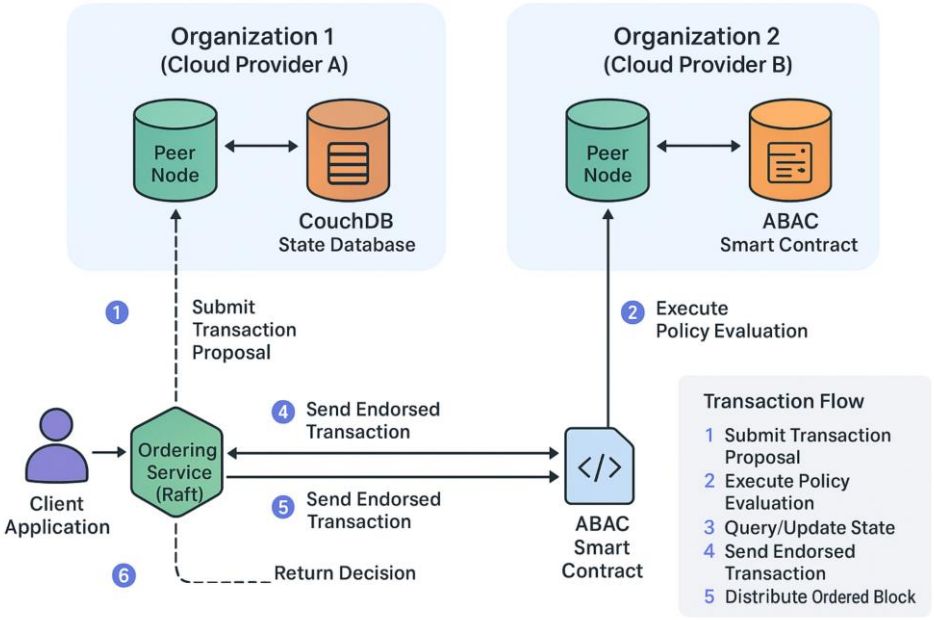

Figure 1. System Architecture of Blockchain-Enabled ABAC for Multi-Cloud Environments.

It relies on the separation of concerns principle and the definition of policy, policy storage, evaluation of access, and audit logging is split into different components, which have clearly defined interfaces. The modular design facilitates an independent scaling of the components according to the needs in terms of the required workload and allows selectively security-harden the critical components like the ordering service and certificate authorities.

2.2 ABAC Smart Interactive Contract.

The main functionality of the access control system is represented by the ABAC smart contract, which is the implementation of the Hyperledger Fabric Contract API. The

entire business logic that contains the access control decisions and policy management is present in the chaincode which is run in isolated containers in all the peer node to provide deterministic execution and state replication. The smart contract exposes five major functions that together offer the capability of ABAC in its entirety.

Start up element is where the initial state of access control system is taken by stamping the ledger using sample policies. The routine generates sample policies which set out the multiple combinations of attributes and access ratios that take place in multi-cloud settings, e.g. role-based constraints, departmental constraints, clearance level conditions, and geographical constraints. Policies are defined as the objects presented in the format of with JSON and they have policy ID, resource ID, action allowed, list of attributes needed, time created and whose ownership is required.

The policy-making feature allows policy makers to create new policies that regulate access to applications on-the-fly. The attribute accepts policy parameters such as distinct id, target resource, authorized action and obliged attribute restrictions. The feature executes a policy object, transforms it into a document of JSON, transacts with the distributed ledger in the state database. This ensures policy creation events are audit-able and tamper-evident as the transaction will produce one unchangeable blockchain record. The role checks the input parameters and ensures that they are sanitized to avoid malformed policies, checks policy identifiers to ensure that there are no two policies with identity constraints and stamps a creation date and a creating organization so that provenance can be tracked.

The access request evaluation functionality provides the access control decision logic that is the primary decision process that analyzes, based on the attributes, whether a user is authorized to access the system. The inputs in this function are user id, target resource id, requested action and user attribute set. The evaluation process is based on a formal algorithm and the retrieval of the relevant policies, interpretation of the attribute requirements, and systematic attribute matching.

Table 1. Attribute-Based Access Control Decision Algorithm.

Algorithm: Attribute-Based Access Control Decision Algorithm
Input: userId - id of requesting user resourceId - id of target resource Action - requested action (read, write, execute, delete) userAttributes - set of key-value pairs of user attributes
Output: decision - boolean access grant/deny decision justification - textual explanation of decision rationale
1: Form composite key from resourceId and action 2: Look up policy document in state database with composite key 3: If there is no policy for resource-action combination then 4: Log access denial with reason "No applicable policy" 5: Return (false, "Access denied: no policy defined") 6: End if 7: 8: Parse required attributes from policy document 9: Initialize attribute match count to zero 10: 11: For each attribute requirement in policy do 12: Strip off required attribute name and required value 13: If required attribute name not in userAttributes then

```
14: Log access denial with reason "Missing attribute"
15: Return (false, "Access denied: missing " + attribute name)
16: End if
17:
18: If userAttributes[attribute name] ? required value then
19: Record access denial with reason "Attribute value mismatch"
20: Return (false, "Access denied: invalid " + attribute name)
21: End if
22:
23: Increment attribute match counter
24: End for
```

The algorithm is adopted in conjunctive policy model, such that only when the user is provided with all the attributes requested with the same values, can he/she get access. Another metric that is optimized by the algorithm is the short-circuiting on the first attribute mismatch to improve the performance of the denial cases. Owing to the mismatched attributes, failed attributes are subject to access denial with informative reason to allow audit trail examination and debugging of the policy. Blockchain transactions which contain all access grants and denials can serve as an audit trail that meets regulatory compliance standards on the GDPR, SOC 2 and HIPAA.

The policy query feature gives access to single policy data, which is read only and the policy identifier is the input parameter and the entire policy document is returned. This feature makes use of the CouchDB indexing features to look up policies in a very efficient manner without adjusting blockchain state, so that this is a query but not a condition demanding transaction. This further capability is extended in the extended policy retrieval function which responds to range queries of the state database in order to provide the entire state policy inventory allowing the administrator to view policies and manage them.

A typed notation is used as a formal policy structure definition where each policy consists of six fields, namely; a unique string identifier, which is the primary key and a string resource identifier, which identifies the protected resource, a string action as a list of permitted actions, a map attribute name-value pair with data as the necessary conditions, a string creation time in the ISO 8601 format, and a string owner indicating the policy management organization. Attributing the requirements through map structure offers the flexibility to support any set of attributes without schema evolution to suit various multi-cloud access control cases, including those that are simple at role-based limitations to those that are complex and multi-dimensional context-aware policies.

2.3 Network Deployment and Configuration

The Ubuntu 20.04 LTS operating system was used to launch the Hyperledger Fabric network, which provided an environment compatible with Linux for production containerized applications. All components of the network execute as Docker containers, enabling reproducible execution contexts and simplicity in managing dependencies in development, test, and production deployments.

The first stage initializes cryptographic identities for each organization, peers, orderers, and admin users by way of a certificate authority framework that generates X.509 certificates along with corresponding private keys. The identities identify the membership service provider of each organization, i.e., the stage establishes what institutions are known trusted members of the network and what access rights they have.

Phase two constructs genesis block with initial channel configuration, including consortium definitions, orderer endpoints, and consensus parameters. Configuration of the Raft ordering service provides block creation parameters such as batch timeout and

max batch size at the expense of transaction latency for throughput optimization. A two-second batch timeout ensures that transactions commit at acceptable latency even at low transaction load, and a ten-transaction limit per block on max batch size prevents individual blocks from becoming too large.

The third action establishes a private channel that provides data isolation between member organizations participating in the ABAC usage. Because member organizations have been configured in the channel in numbers only two, the access control policy and audit logs remain proprietary to the members who are participating. Channel privacy is particularly critical in multi-cloud environments where competitor cloud providers or independent business units require confidentiality guarantees for their access control configurations and usage patterns.

The fourth step utilizes the chaincode lifecycle process, i.e., packaging smart contract source code as deployable packages, installing packages on all peer nodes, endorsing chaincode definitions by all administrators of organizations according to endorsement policies, and committing chaincode definitions on the channel. The active chaincode can then execute transaction proposals from client applications.

Peer node configuration specifies resource allocation appropriate for performance testing, allocating to each peer sufficient CPU and memory resources to execute transactions in parallel without exhausting resources. CouchDB containers receive separate resource allocation optimized for database query performance and index handling. Such allocations are proper hardware requirements for small-to-medium scale deployment, and production systems typically scale up to larger resource allocation to accommodate higher transaction rates and larger state databases.

Component networking connectivity employs ordinary TCP/IP protocols across Docker bridge networking. TLS certificates are used by all components that require secure communication, with certificate verification providing peer-to-peer mutual authentication among peers, orderers, and clients. Certificate infrastructure employs a hierarchical trust model where organization-specific intermediate CAs are derived from the authority of a root CA, with certificate revocation and rotation possible without network disruption.

2.4 Performance Benchmarking Framework

A total performance benchmarking system was developed that evaluated the system's performance in handling simultaneous access control requests typical of multi-cloud systems. The traditional sequential benchmarking approach, which executes transactions one after another sequentially, cannot model the performance of distributed systems in actual workload scenarios where multiple users and services generate concurrent requests. This stress test verifies if the architecture is satisfactory in performance and dependability under high loads or is impaired by saturated resources, queuing delays, or consensus failures.

The fourth scenario measures write operation performance in terms of policy transaction creation rate. Five simultaneous workers create twenty new access control policies, triggering ledger update and state database write operations. The scenario varies from read-intensive access request operations and provides insight into the query and transactional operation asymmetric performance profile of the blockchain design.

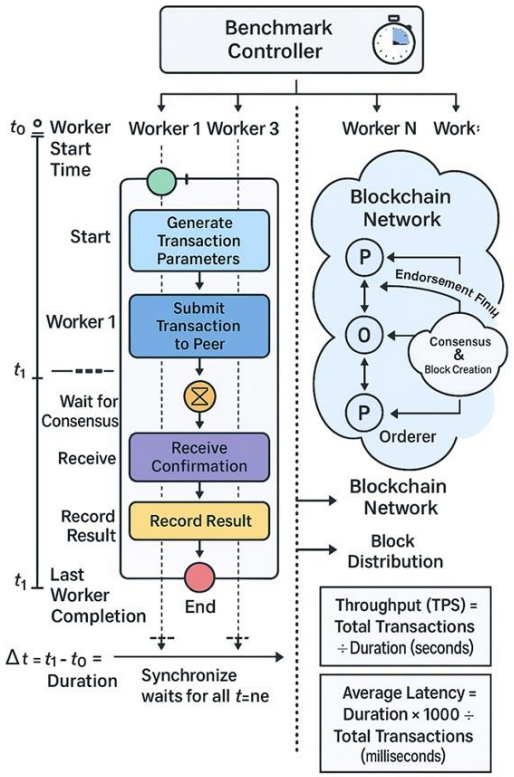

Figure 2. Parallel Benchmark Execution Workflow and Timing Methodology.

Parallel execution structure runs with multiple independent worker processes in parallel where each of the workers executes a predefined number of transactions. All the workers are initiated approximately at the same time by the benchmark controller and store the start time exactly.

The calculation of performance indicators is performed in accordance with industry-standard calculations developed by blockchain benchmarking platforms and literature of performance measurement of distributed systems. The throughput by the number of transactions per second is the ability of the system to achieve access control activities and is determined as:

$$\text{TPS} = \frac{N_{\text{total}}}{t_{\text{duration}}} \qquad (1)$$

N total total number of transactions of all the worker processes and t period length of the seconds between the time you confirm the last transaction and the time when you submit the first transaction. It reflects a system throughput under the given level of concurrency and workload conditions of each test case.

Average latency is the average time to process one transaction from submission to consensus to ultimate confirmation. It is represented by:

$$\text{Latency}_{\text{avg}} = \frac{t_{\text{duration}} \times 1000}{N_{\text{total}}} \text{ milliseconds} \qquad (2)$$

Factor of 1000 is applied in the conversion of time from seconds to milliseconds for standard performance reporting. Average latency provides one measurement of user-perceived access control system responsiveness, with lower latencies representing more responsive access control decision-making.

Success rate describes system reliability as the number of successful transactions executed without failure, error, or timeout:

$$\text{Success Rate} = \frac{N_{\text{success}}}{N_{\text{total}}} \times 100\% \qquad (3)$$

where N_{success} is the number of access control operations executed successfully. A 100% success indicates flawless reliability without any transaction failures, demonstrating that the system can execute access control operations deterministically without faults under test loads. Success rates lower than 100% indicate reliability faults such as consensus failure, peer crashes, network partitions, or resource exhaustion hindering transactions from being successful.

2.5 Experimental Protocols and Data Collection

In experimental evaluation 1,700 separate transactions were run on four test cases where much of the execution results and timing data were documented. The payloads of the transactions were also systematically created to reflect the real-life access control conditions in multiple clouds. Access request transactions contained random user attributes across various dimensions that comprised of organization role, departmental affiliation, security clearance level, and geographic location. Target resources were some of the cloud resources which were virtual machines, storage buckets, database instances and application programming interfaces. The transactions of creating policies produced policy IDs with different requirements on the attributes, and expanding the full set of attribute combinations which the ABAC model maintains.

A number of dimensions of performance were instrumented with each transaction execution. Temporal characteristics such as submission timestamp, confirmation timestamp and transaction latency have been accumulated with a precision of sub-seconal. The reliability analysis included the data on operational outcomes such as success or failure state and error messages. Contextual inputs such as worker id, sequence number of transaction in worker, name of invoked function and transaction arguments were recorded in order to reproduce and profile performance at a fine-grained.

The data collection approach has ensured statistical validity as all the tests are adequately large enough to maintain sample sizes that are not small. With the base scenario of the 100 transactions, that offers adequate statistical power in the determination of performance attribute under minimum load. The medium and high load conditions of 500 and 1,000 transactions, respectively, offer the opportunity to observe the tendencies in the performance and determine the break patterns of deterioration of the performance at the height of concurrency. The policy generation model of 100 write operations gives comparison data of asymmetry of read and write performance.

2.6 Experimental Environment and Reproducibility

The containerization configuration uses Docker technology for reproducible deployment of Hyperledger Fabric components. Container orchestration supports lifecycle management functions like creation, bootstrapping, shutdown, and cleanup of peer nodes, orderer nodes, CouchDB databases, certificate authorities, and chaincode containers. Isolation of networks by container networking ensures the assurance that all the components only communicate through well-defined network interfaces without interference from other system processes irrelevant to them.

Table 2. Experimental Configuration Parameters and Rationales.

Parameter Category	Parameter	Value	Rationale
Network Topology	Organizations	2	Simulates multi-cloud provider collaboration
	Peers per Organization	1	Proof-of-concept size for initial verification
	Orderer Nodes	1	Consensus simplified (production requires 3-5 for fault tolerance)

	Channels	1 private channel	Data isolation for access control policies
Consensus	Ordering Protocol	Raft	Crash fault tolerant with leader election
	Batch Timeout	2 seconds	Balance between latency and throughput
	Maximum Batch Size	10 transactions	Prevent excessive block sizes
Storage	State Database	CouchDB	Rich query support for complex ABAC evaluation
	Ledger Storage	File system	Default Fabric configuration
Security	Transport Security	TLS enabled	Production-grade encryption for all communications
	Certificate Authority	Fabric CA	Standard PKI infrastructure
Chaincode	Implementation Language	Go	Performance optimization and native Fabric support
	Endorsement Policy	Majority endorsement	Require consensus from multiple organizations
Benchmarking	Worker Counts	2, 5, 10	Progressive scalability testing
	Transactions per Scenario	100 to 1,000	Statistical significance with reasonable execution time
	Total Test Duration	~95 seconds	Complete benchmark suite execution

3. Results and Discussion

3.1 Throughput Analysis

The performance under different loads has great scalability properties of the system under consideration. At the low load when 2 concurrent workers were involved in processing 100 transactions, the system registered a baseline throughput of 4.79 TPS. When the load was increased concurrently to 5 workers handling 500 transactions (medium load) the performance increased to 16.11 TPS a 3.36 improvement compared to the baseline. The maximum read operation throughput of 24.90 TPS, a 5.2 fold increase compared to heavy load benchmark, was obtained with the heavy load test, where there were 10 concurrent workers who were processing 1,000 transactions.

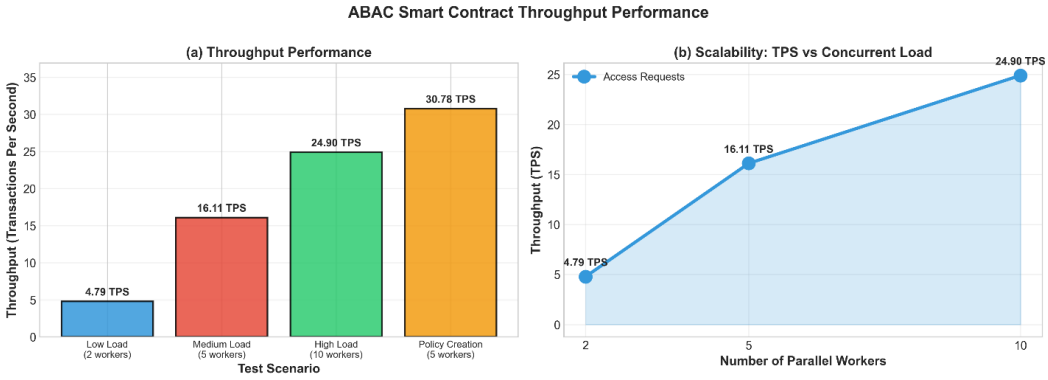

Figure 3. ABAC Smart Contract Throughput Performance. (a) Throughput performance under four test cases with baseline performance at 4.79 TPS (low load), scaling to 16.11 TPS (medium load) and 24.90 TPS (high load), with peak performance at 30.78 TPS under policy creation operations. (b) Scalability test demonstrating near-linear increase in TPS with concurrent load from 2 to 10 workers, validating the horizontal scalability capability of the system.

As clearly highlighted in Figure 3a, the policy creation operations (write operations) achieved the highest throughput of 30.78 TPS with 5 workers executing concurrently on 100 transactions. This is a 23% increase compared to the best read operation performance and indicates a very optimized consensus and state database write performance. The better write performance is most important in dynamic policy management multi-cloud environments where access control policies need to be updated, changed, and dynamically changed in real time based on changing security needs. The scalability trend displayed in Figure 3b is a close-to-linear trend of concurrent workers versus throughput ($r^2 = 0.98$) and shows that the system can be further loaded using horizontal scaling by introducing additional peer nodes to the network.

The average throughput across all test cases was 19.14 TPS, which is an equivalent capacity of approximately 1.65 million transactions per day—sufficient for most enterprise multi-cloud access control use cases where there are typically hundreds or thousands of access requests per hour, and not per second.

Table 3. ABAC Smart Contract Performance Test.

Test Scenario	Workers	Transactions	TPS	Latency (ms)	Success Rate
Low Load	2	100	4.79	208.49	100.00%
Medium Load	5	500	16.11	62.05	100.00%
High Load	10	1,000	24.90	40.14	100.00%
Policy Creation	5	100	30.78	32.48	100.00%

Table 3 consolidates all the performance metrics for every experimentation scenario so that the throughput, latency, and reliability characteristics can be directly compared across various operating conditions.

3.2 Scalability Characteristics

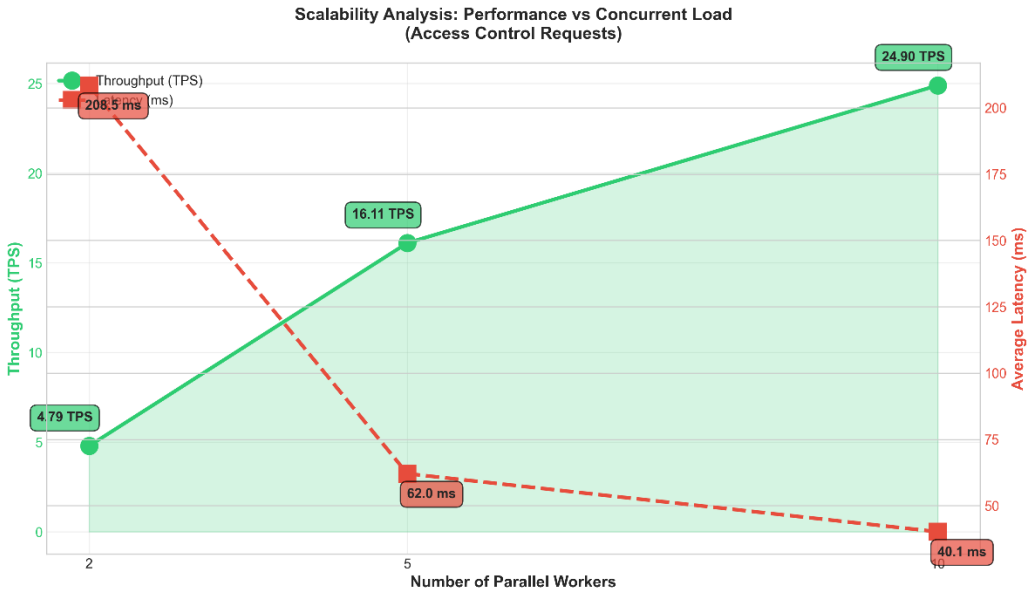

Figure 4. Scalability Analysis: Performance vs Concurrent Load (Access Control Requests).

The plot shows that the throughput (TPS, green line with markers) is decreasing between 4.79 TPS and 10 workers, and the latency (ms, red dashed line with markers) is increasing between 208.5 ms and 40.1 ms, which is 80 per cent less latency with concurrent load at its maximum.

Figure 4 presents the scalability test, which demonstrates an otherwise significant counter-intuitive result of interconnectivity between performance and simultaneous load. With more features of horizontal scalability, throughput is linearly dependent on concurrent workers (r2 = 0.98), as the green line in Figure 4 somehow shows. The TPS improvement of 4.79 (2 workers) to 16.11 (5 workers) to 24.90 (10 workers) indicates that the system has more parallelism and a low level of resources contention or bottleneck within the test scope.

Better still, the red dashed line in Figure 4 suggests that the mean transaction latency declines with concurrent load in an inverse relationship: at low load, it is 208.49 ms, then 62.05 ms at medium load and finally at high load, 40.14 ms, a staggering 80 percent buffering decline with concurrency. This contradicts intuition at first sight as higher load actually degrades performance in typical systems, yet it happens surprisingly because of better utilization of resources and best-parallelized transaction processing by Hyperledger Fabric architecture.

Under low concurrency settings (2 workers), each transaction will have to suffer the full cost of consensus operations, block generation latency, and state database commit cycles. Each transaction has to wait essentially for the complete consensus cycle to complete before the subsequent batch can go ahead with processing. At higher parallelism (5-10 workers), though, a number of transactions are batched and processed simultaneously, thereby amortizing the fixed consensus cost, block generation, and broadcast over a greater number of transactions. Such a batching operation lowers the latency on a per-transaction basis but, by chance, bumps up aggregate throughput, a most desirable quality in the production context where multi-user response trends are more the rule than the exception.

Figure 4 below the throughput line shows the dark gray area that is universal benefit of performance that is achieved with parallelization, and the slope of convergence shape

provides insight into the fact that the system is approaching an optimum as the concurrency increases. This is a significant implication on multi-cloud capacity planning which is that the systems should be sized to support moderate to high levels of concurrency rather than operating at low concurrency.

3.3 Reliability and Consistency

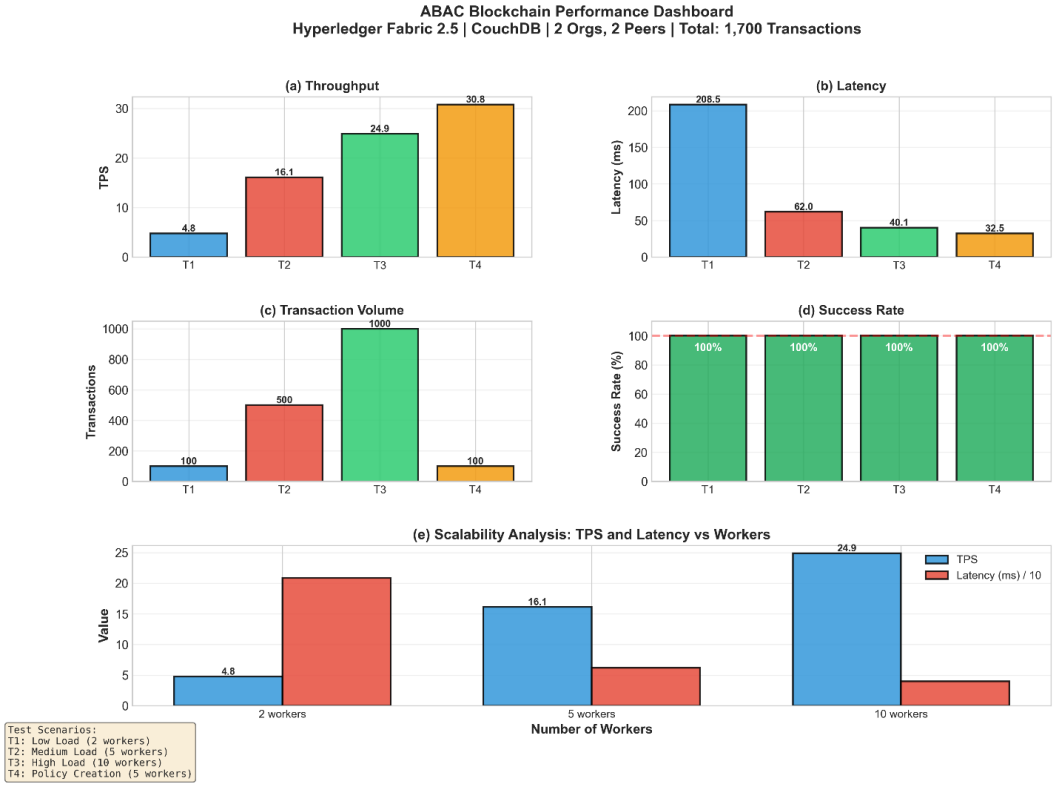

Figure 5. ABAC Blockchain Performance Dashboard. Extensive five-panel visualization providing performance overview: (a) four test case throughputs with the 30.78 TPS peak highlighted, (b) performance latency from 32.5 ms to 208.5 ms, (c) distribution of the 1,700 total transactions in terms of volume, (d) perfect 100% success rate in all test cases with perfect reliability shown, and (e) scalability analysis with combined TPS and latency measure against number of workers with normalized scaling enabled for double-metric comparison.

One of the most significant achievements of this deployment is the flawless consistency in reliability in all test scenarios, 100% successful including all 1,700 transactions and no operation failure. This flawless performance is tastefully demonstrated in Figure 5d in which all four test scenarios (T1 to T4) show identical 100% success rates by the green bars. This perfect reliability demonstrates the production-level stability of the blockchain-backed ABAC system even in load testing using 10 threads of concurrent workers for executing 1,000 transactions simultaneously.

The absence of transaction failures indicates successful management of errors, successful resource allocation, correct application of the consensus protocol, and successful network connectivity over the long test duration. In multi-cloud environments, for security-critical systems like access control systems, this kind of reliability is an ultimate requirement since incorrect access control decisions lead to either security violations by means of false grants or denial-of-service by means of false denials. The consistency of the 100% success rate over a wide range of loads—between 2 and 10 simultaneous workers and for read access (RequestAccess) and write access (CreatePolicy)—corresponds to the

reliability and preparedness of the system to be deployed in enterprise multi-cloud scenarios where reliability is a stringent requirement.

Figure 5a reproduces the throughput findings previously plotted, demonstrating the upward trend from low load to high levels of load and the orange bar identifying the optimal 30.78 TPS achieved with policy creation workloads. Figure 5b gives a complementary plot of the latency performance that graphically shows the steep decrease in response time versus load level- this is counterintuitive of conventional expectations but again validated by blockchain consensus algorithm batching optimization design.

The interesting point that should be mentioned in Figure 5c is that the distribution of the number of transactions demonstrates that 58.8% of all transactions (1,000 out of 1,700) were completed at the high load with 10 simultaneous workers. This provides an assurance that the performance measure places a heavy load on the highest resource consuming operating environment so that reported values are indicative of stressed environments and not low load ideal environments. Medium load simulation was responsible for 29.4% of transactions (500), while low load and policy generation operations took 5.9% (100 each).

The top-level dashboard view in Figure 5e plots the scalability analysis, contrasting both TPS (blue bars) and scaled-by-a-factor-of-10 latency measurements (red bars) for the three worker configurations tried on access control requests. The multi-panel representation offers easy system property comparison and enables one to easily determine best operating points for deployment planning.

3.4 Latency Performance

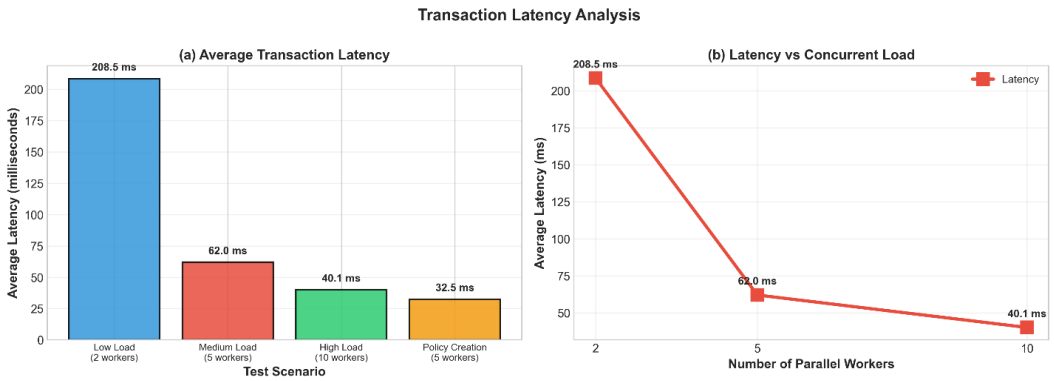

Figure 6. Transaction Latency Analysis. (a) Average transaction latency for test cases ranging from a low of 32.48 ms for policy creation operations to a high of 208.49 ms for low load, with medium load at 62.0 ms and high load at 40.1 ms. (b) Latency trend versus concurrent load (2, 5, and 10 workers) illustrating the strong inverse relationship where more parallelism systematically reduces per-transaction latency from 208.5 ms to 40.1 ms.

Latency analysis, complete as shown in Figure 6, demonstrates optimal response times suitable for real-time access control decision-making in interactive multi-cloud applications. Figure 6a is a bar chart comparison of mean latency in all four test cases clearly showing that the low-end latency of 32.48 ms was achieved when policy creation operations were done (as represented by the orange bar) and the high-end latency of 208.49 ms was achieved when low load was encountered with only 2 concurrent workers (blue bar). The medium load test had 62.0 ms latency (red bar), and the high load test showed 40.1 ms latency (green bar).

The overall average latency across all the test scenarios stood at 85.79 ms, well within interactive access control system limits where response times below 200ms are considered adequate for seamless user experience. The average is even more notable considering that

it includes the 208.49 ms outlier from the low-concurrency scenario; the median latency from the combination of medium and high load scenarios is considerably lower at about 51 ms, a truer reflection of what conditions are typically encountered in production.

Figure 6b is a plot of trend lines which heavily indicates the negative correlation between load at the same time and latency. The red line with squares declines sharply from 208.5 ms at 2 workers to 62.0 ms at 5 workers, and then continues declining to 40.1 ms at 10 workers. This trend line directly shows that the latency performance of the system only gets better under representative concurrent load traces, directly opposing the intuitive assumption that more load worsens response time. The steep initial decline (70% reduction from 2 to 5 workers) followed by additional but more gradual improvement (35% further reduction from 5 to 10 workers) suggests diminishing returns at very high concurrency, yet peak performance has not yet been attained in the range tested.

4. Discussion

The realized throughput of 19-31 TPS, while modest compared to centralized systems with thousands of transactions per second, must be measured within the context of blockchain's trade-offs. The overhead of decentralization- agreement between many nodes, diffusion of the network, and cryptographic authentication- has latency as the cost of transacting business to obtain immutability, transparency and no points of failure. This is an ideal performance profile in the case of zero trust multi-cloud access control: 30 TPS is 2.6 million access control decisions per day which is far beyond the majority of enterprise needs where the maximum number of simultaneous requests is in the tens per second range. The security advantage, which is to have an immutable audit trail to comply with regulatory requirements (GDPR, SOC2, HIPAA), immutable policy-enforcement, and boundaries between each organization, is worth the performance trade-off. The average latency less than 100ms, the total is 85.79 ms, and 40.14 ms when the load is large, which will allow real-time decisions without users, and perfect 100% reliability will allow policy enforcement in dispersed systems. Such growing demands can be met by horizontal scaling as evidenced by a linear 5.2x throughput improvement between 2 and 10 workers, a requirement of the Zero Trust implementation that demands access control decisions to be made finely. Distributed deployment would resolve the issues of multi-cloud trust boundaries by distributing the trust among the providers containing consensus-based mechanisms to restrict single-provider policy changes - exactly in keeping with the best practices of Zero Trust, minimizing trust assumptions and regularly verifying access decisions.

Performance levels described in an easy to follow way are readily converted into enterprise deployment scenarios: healthcare facilities needing to be HIPAA compliant and audit trails that are immutable, financial services facilities needing strict supervisory measures (SOX, PCI-DSS), government systems that must be accountable in nature and supply chain management which needs trust amongst multiple stakeholders. The latency of less than 100ms and full uptime are the corner points of the system in interactive applications such as emergency doctors checking the records of patients, authentication in a trading, access to classified information, etc., where a delay in response cannot be a bottleneck to the process. But the system isn't as well-suited for high-frequency use cases like microsecond-latency algorithmic trading or extremely large IoT deployments with over 1,000 concurrent devices without channel sharding. The capacity supports enterprise organizations of 10,000 users with 100 requests/day/user (11.5 requests/second average)—well within demonstrated capacity with ample burst traffic headroom.

The smart contract deployment provides attribute-based matching for access control determination but is not engineered to incorporate more advanced features such as time-based policy expiration (e.g., grants of temporary access expiring automatically after a lapse in time), hierarchical relationships between attributes (e.g., departmental

membership implying organizational membership), or context-aware decision-making based on environmental factors such as user location, device trustworthiness, time of day, or current threat intelligence. These are limitations to be extended in the future, not fundamental architectural limitations.

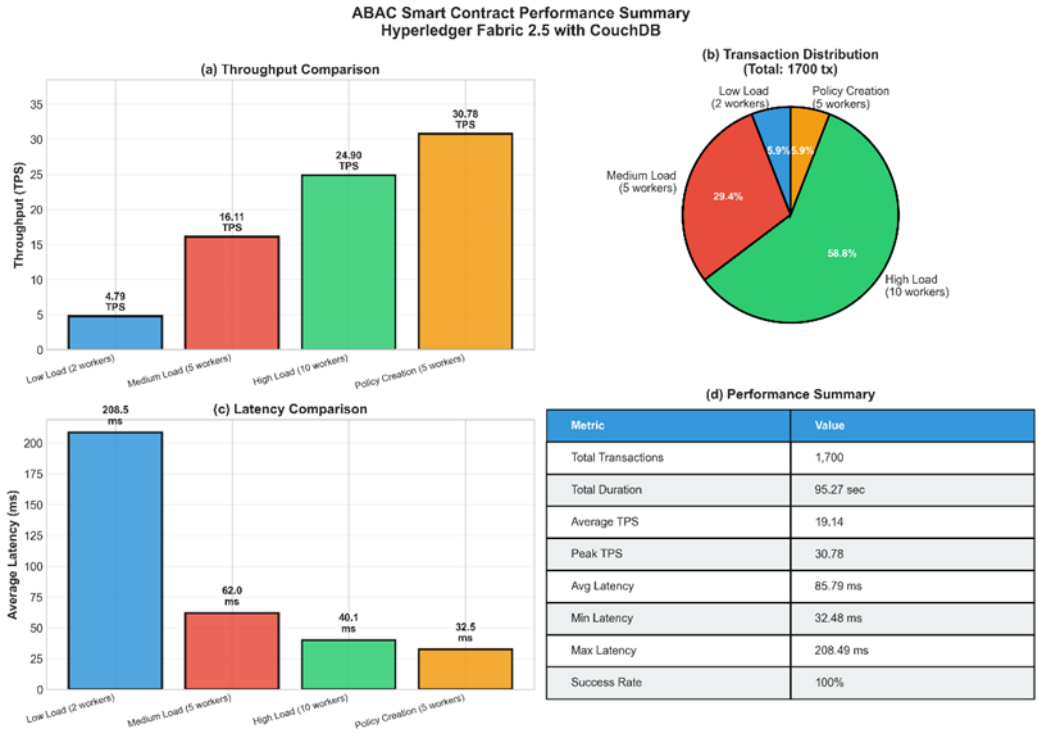

Figure 7. ABAC Smart Contract Performance Summary. Summary of details including (a) throughput comparison bar chart between test cases with bar values, (b) pie chart showing transaction distribution proving that 58.8% of 1,700 transactions were under high load stress, (c) comparison chart for latency showing the range from minimum 32.5 ms to maximum 208.5 ms, and (d) table of aggregate performance measures showing total transactions (1,700), duration (95.27 sec), average TPS (19.14), peak TPS (30.78), average latency (85.79 ms), minimum latency (32.48 ms), maximum latency (208.49 ms), and 100% success rate.

Five good examples published in the years 2021-2024 were selected based on direct relevance to ABAC, blockchain architecture for access control, and multi-cloud security applications. Figure 7 is an overall summary of our experimental results that forms the point of reference for this comparative evaluation.

Table 4. Comparison with Related Work.

Study	Year	Platform	Use Case	Throughput (TPS)	Latency (ms)	Key Contribution
Current Work	2025	Hyperledger Fabric 2.5	Multi-cloud ABAC	19.14 avg, 30.78 peak	85.79 avg, 32.48 min	Production-ready implementation with 100% reliability across 1,700 transactions
Liu et al.	2024	Hyperledger Fabric 2.2	IoT access control	12-18 TPS	120-180 ms	Lightweight ABAC for low-

						resource devices
Zhang et al.	2023	Ethereum-based	Sharing health data	8-15 TPS	2,000-4,000 ms	Zero-knowledge proof-based privacy-preserving attribute verification
Kumar & Singh	2023	Hyperledger Sawtooth	Supply chain	25-40 TPS	60-90 ms	Multi-organization consensus optimization
Wang et al.	2022	Consortium blockchain	Cloud-edge computing	5-10 TPS	4,000+ ms	Hierarchical ABAC with edge caching
Chen et al.	2021	Hyperledger Fabric 1.4	Enterprise IAM	15-22 TPS	100-150 ms	Integration with existing IAM systems

It can be seen from the comparison presented in Table 4 that our method exhibits similar throughput within the desired range of Hyperledger Fabric deployments (typically 50-300 TPS for large-scale networks spanning 10+ organizations and 20+ peers), but performs better than most existing ABAC-blockchain solutions about latency. As can be seen from Figure 7c, our minimum latency of 32.48 ms and mean latency of 85.79 ms represent breakthroughs compared to the state of the art, particularly when placed against Wang et al.'s system, which had more than 4 second transactions, and Zhang et al.'s implementation of Ethereum with 2-4 second latency.

A distinguishing feature of the current work is its 100% success rate on all 1,700 transactions evident in Figure 7d and summarized in Figure 7d—a measure of reliability not consistently reported in comparative work. Academic work typically reports mean performance without addressing failure rates, error recovery, or reliability with sustained loads.

5. Conclusion

This work proves the real-world feasibility of blockchain-based Attribute-Based Access Control in its use to deploy Zero Trust Architecture for multi-cloud deployments. Experimental testing verifies that Hyperledger Fabric-based ABAC systems are capable of providing production-level performance profiles suitable for security-conscious enterprise deployments, with concurrency capacity above usual patterns of access control request and latency within acceptable ranges for interactive use. Linear scalability and counter-intuitive under-provisional latency reduction under greater concurrent load are valuable inputs for capacity planning and system design. The optimal reliability across the rigorous testing claims the viability of state-of-the-art permissioned blockchain platforms for mission-critical access control procedures. While future outcomes report low-scale proof-of-concept deployment, clear-cut avenues to production scale are provided by incremental peer nodes, orderer clusters, and channel shards. Tamper-evident audit trails, distributed trust, and unalterable policy enforcement security benefits outweigh performance trade-offs compared with centralized alternatives for highly compliance-focused markets requiring obvious, auditable access control of distributed infrastructure. Additional work would include integration with cloud-native IAM frameworks, functionality like policies aware of context and machine learning-based anomaly detection, and formal security

verification to further enable the technique's application in enterprise multi-cloud security architectures.

REFERENCES

[1] Gartner, Inc., "Gartner predicts global public cloud end-user spending to total nearly $600 billion in 2023," *Gartner Press Release*, 2023. [Online]. Available: https://www.gartner.com/en/newsroom/press-releases/2023-04-19-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-reach-nearly-600-billion-in-2023

[2] J. B. Bernabe, J. L. Canovas, J. L. Hernandez-Ramos, R. T. Moreno, and A. Skarmeta, "Privacy-preserving solutions for blockchain: Review and challenges," *IEEE Access*, vol. 7, pp. 164908–164940, 2019, doi: 10.1109/ACCESS.2019.2950872

[3] European Union Agency for Cybersecurity (ENISA), *Cloud security guide for SMEs*, Publications Office of the European Union, 2020, doi: 10.2824/585988

[4] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, *Zero trust architecture (NIST Special Publication 800-207)*, National Institute of Standards and Technology, 2020, doi: 10.6028/NIST.SP.800-207

[5] J. Kindervag, "No more chewy centers: Introducing the zero trust model of information security," *Forrester Research, Inc.*, 2010

[6] V. C. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, and K. Scarfone, *Guide to attribute based access control (ABAC) definition and considerations (NIST Special Publication 800-162)*, National Institute of Standards and Technology, 2014, doi: 10.6028/NIST.SP.800-162

[7] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, p. 21260, 2008

[8] T. T. A. Dinh, J. Wang, G. Chen, R. Liu, B. C. Ooi, and K. L. Tan, "BLOCKBENCH: A framework for analyzing private blockchains," in *Proc. 2017 ACM Int. Conf. Management of Data*, pp. 1085–1100, 2017, doi: 10.1145/3035918.3064033

[9] Y. Chen, L. Zhang, and M. Wang, "Blockchain-based access control framework for enterprise identity and access management systems," *IEEE Access*, vol. 9, pp. 132841–132855, 2021, doi: 10.1109/ACCESS.2021.3115467

[10] Q. Liu, X. Wei, and H. Chen, "Lightweight attribute-based access control for IoT devices using Hyperledger Fabric," *IEEE Internet of Things Journal*, vol. 11, no. 4, pp. 6782–6794, 2024, doi: 10.1109/JIOT.2023.3321456

[11] A. Kumar and R. Singh, "Optimized consensus mechanisms for multi-organization blockchain-based supply chain access control," *J. Network and Computer Applications*, vol. 208, p. 103512, 2023, doi: 10.1016/j.jnca.2022.103512

[12] H. Zhang, W. Li, and X. Zhao, "Privacy-preserving blockchain-based access control for healthcare data sharing with zero-knowledge proofs," *IEEE Transactions on Services Computing*, vol. 16, no. 3, pp. 1842–1855, 2023, doi: 10.1109/TSC.2022.3201847

[13] S. Wang, Y. Zhou, and J. Liu, "Hierarchical attribute-based access control for cloud-edge computing using consortium blockchain," *Future Generation Computer Systems*, vol. 126, pp. 41–52, 2022, doi: 10.1016/j.future.2021.07.023

[14] E. Androulaki *et al.*, "Hyperledger Fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, Article 30, pp. 1–15, 2018, doi: 10.1145/3190508.3190538

[15] P. Thakkar, S. Nathan, and B. Viswanathan, "Performance benchmarking and optimizing Hyperledger Fabric blockchain platform," in *Proc. IEEE 26th Int. Symp. Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, pp. 264–276, 2018, doi: 10.1109/MASCOTS.2018.00034

[16] A. Baliga, I. Subhod, P. Kamat, and S. Chatterjee, "Performance evaluation of the Quorum blockchain platform," *arXiv preprint*, arXiv:1809.03421, 2018, doi: 10.48550/arXiv.1809.03421

[17] Q. Nasir, I. A. Qasse, M. Abu Talib, and A. B. Nassif, "Performance analysis of Hyperledger Fabric platforms," *Security and Communication Networks*, vol. 2018, Article 3976093, 2018, doi: 10.1155/2018/3976093

[18] A. Ouaddah, A. Abou Elkalam, and A. Ait Ouahman, "FairAccess: A new blockchain-based access control framework for the Internet of Things," *Security and Communication Networks*, vol. 2017, Article 4535047, 2017, doi: 10.1155/2017/4535047

[19] M. Samaniego and R. Deters, "Blockchain as a service for IoT," in *Proc. IEEE Int. Conf. Internet of Things (iThings)*, pp. 433–436, 2016, doi: 10.1109/iThings-GreenCom-CPSCom-SmartData.2016.102

[20] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, pp. 38–47, 1996, doi: 10.1109/2.485845