Design of Secure Network Isolation and Trust Models for Multichain Blockchain in Kubernetes

Igor Andrushchak [1], Viktor Kosheliuk [2*]
[1]Department of Software Engineering, Lutsk National Technical University, Ukraine,
Email: 9000@email.ua
[2]Department of Computer Science, Lutsk National Technical University, Ukraine,
Email: viktor.koshelyuk@gmail.com
[*] Corresponding author

Abstract: **Container orchestration platforms such as Kubernetes are increasingly deployed in cloud-native and edge computing environments, where ensuring secure network isolation and trustworthy interactions between distributed components remains a critical challenge. Lightweight Kubernetes clusters, in particular, often lack robust mechanisms for decentralized trust management and tamper-resistant security auditing. This paper proposes a secure network-isolation and trust model for Kubernetes environments based on the Multichain blockchain. The study aims to enhance security assurance and audit transparency by introducing a decentralized trust layer that complements native Kubernetes networking mechanisms. The research objectives include analyzing network-level security threats in Kubernetes clusters, designing a blockchain-based trust-and-audit architecture, integrating Multichain with Kubernetes networking components, and evaluating the effectiveness of the proposed model using quantitative metrics.**

The proposed approach is validated through experimental deployment in a controlled Kubernetes environment. Effectiveness is assessed using normalized indicators for security, performance, reliability, and integration, combined into an integrated effectiveness index. Radar-based visualization is employed to compare the proposed solution with a baseline Kubernetes configuration without blockchain support. Experimental results demonstrate that the proposed model significantly improves security and reliability metrics while maintaining acceptable performance overhead. The integrated effectiveness index confirms a measurable overall improvement compared to traditional Kubernetes deployments. The scientific contribution of this work lies in integrating decentralized, blockchain-based trust and immutable audit logging with Kubernetes network isolation mechanisms. The proposed model provides a practical, scalable approach to enhancing the security of Kubernetes clusters across cloud-native and edge computing infrastructures.

Keywords: **Blockchain security, Kubernetes, Network isolation, Trust models, Container orchestration.**

I. INTRODUCTION

The rapid adoption of containerization technologies and microservice-based architectures has significantly transformed the design and deployment of modern distributed systems. Kubernetes has emerged as the de facto standard platform for container orchestration, enabling automated deployment, scaling, and management of containerized applications across cloud-native and edge computing environments [1,2]. However, as Kubernetes clusters grow in scale and complexity, ensuring secure network isolation and establishing trustworthy interactions between distributed components remain critical challenges [3,4].

Kubernetes networking is inherently dynamic and highly distributed, relying on virtualized network overlays, service meshes, and policy-based controls to manage communication between pods, namespaces, and nodes [5]. While native mechanisms such as Network Policies and role-based access control provide a foundational level of isolation, they are largely centralized and rely on the integrity of control-plane components [6]. In lightweight Kubernetes deployments, commonly used in edge computing and resource-constrained environments, these security mechanisms are often simplified or partially disabled, increasing the risk of misconfiguration, unauthorized access, and lateral movement attacks within the cluster [7,8].

Another fundamental challenge lies in trust management and security auditing. Kubernetes environments generate a large volume of security-relevant events, including network flows, access requests, and configuration changes [9]. Traditional logging and monitoring approaches typically rely on centralized storage and processing, which introduces single points of failure and raises concerns regarding log tampering, data integrity, and accountability. In multi-tenant or distributed deployments, the lack of a decentralized trust model complicates reliable verification of security events and inter-component interactions [10].

Blockchain technology has recently gained attention as a promising solution for decentralized trust management due to its inherent properties of immutability, transparency, and distributed consensus. By maintaining an append-only ledger shared across multiple nodes, blockchain systems can provide tamper-resistant storage of security events and support trust verification without relying on a single authoritative entity [11]. In particular, permissioned blockchain platforms such as Multichain offer fine-grained access control, configurable consensus mechanisms, and efficient transaction processing, making them suitable for integration with enterprise and cloud-native systems.

Despite growing interest in applying blockchain to cloud

and container security, existing research has primarily focused on identity management, access control, or data integrity at the application layer [12,13]. The integration of blockchain-based trust mechanisms with Kubernetes network isolation remains insufficiently explored. Specifically, there is a lack of systematic approaches that combine native Kubernetes networking controls with decentralized trust models capable of supporting real-time secure auditing and verification of network-level interactions [14,15].

This research addresses these gaps by proposing a secure network-isolation and trust model for Kubernetes environments based on the Multichain blockchain [16]. The proposed approach introduces a decentralized trust layer that complements Kubernetes networking mechanisms by recording security-relevant events, network policy enforcement outcomes, and inter-pod communication metadata in an immutable blockchain ledger. This design aims to enhance transparency, accountability, and trustworthiness while preserving the operational flexibility and performance required in lightweight Kubernetes clusters [17,18].

The objectives of this study are fourfold. First, it analyzes security challenges related to network isolation and trust in Kubernetes-based environments, with particular emphasis on lightweight and distributed deployments [19]. Second, it designs a blockchain-based trust and audit architecture that integrates seamlessly with Kubernetes networking components [20]. Third, it implements and deploys the proposed model in a controlled experimental environment. Finally, it evaluates the solution's effectiveness using a multi-criteria assessment framework based on normalized security, performance, reliability, and integration metrics.

The contributions of this work are threefold. First, it presents a novel architecture that combines Kubernetes network isolation mechanisms with a permissioned blockchain-based trust model [21]. Second, it introduces an integrated effectiveness evaluation methodology that quantitatively assesses the trade-offs between security enhancement and performance overhead. Third, it demonstrates the feasibility and benefits of the proposed approach through experimental validation [22].

The remainder of this paper is organized as follows. Section II reviews related work on Kubernetes security and blockchain-based trust models. Section III describes the proposed architecture and trust model. Section IV presents the experimental setup and evaluation methodology. Section V discusses the results and effectiveness analysis. Finally, Section VI concludes the paper and outlines directions for future research.

II. REVIEW OF LITERATURE

Secure network isolation and trust management in distributed systems have become critical research areas as the adoption of container orchestration platforms and decentralized applications continues to grow. Although Kubernetes provides basic network policy enforcement mechanisms, recent literature highlights persistent gaps in ensuring both isolation and decentralized trust in dynamic multi-tenant environments.

Recent studies emphasize the general importance of security frameworks in Kubernetes. Morić [23] provides a comprehensive analysis of Kubernetes security hardening techniques, including network controls, access controls, and logging for compliance assessment, and highlights persistent vulnerabilities that require more robust models beyond native policy enforcement. However, this work primarily focuses on traditional security controls, without addressing decentralized trust or immutable auditing, which motivates the integration of blockchain technologies in this context.

Blockchain's role in addressing trust and security issues across distributed applications has been investigated across multiple domains. Almarri et al. [24] conducted a systematic exploration of blockchain's capacity to enhance security and trust in Internet of Things (IoT) networks, demonstrating that decentralized, immutable ledgers can improve identity management and transparent transaction verification while identifying scalability and performance challenges inherent in such integrations. Although the focus of this review is on IoT environments, its findings regarding blockchain's potential to decentralize trust and ensure data integrity are highly relevant to Kubernetes environments, where trust between distributed microservices and the auditability of security events are similarly pressing concerns.

Beyond general blockchain properties, broader governance and trust in blockchain systems are also active areas of inquiry. Polcumpally et al. [25] perform a thematic systematic review on blockchain governance, emphasizing that trust mechanisms are foundational to effective governance structures across sectors. Their findings underscore the need for more research grounded in trust rather than merely technological capability, aligning with this study's emphasis on blockchain-based trust models that do not depend on central authorities for verification and accountability.

Punia [26] provides a comprehensive systematic survey of blockchain-based access control mechanisms in cloud computing environments. Reviewing more than 100 studies, the work classifies approaches. It highlights critical challenges in permissioned blockchain implementations, including scalability, interoperability, and security, which must be addressed to ensure effective and reliable access management in distributed systems. This work highlights that blockchain can enhance trust between cloud entities but also faces open research challenges in harmonizing performance and security – an insight crucial for applying blockchain to Kubernetes network isolation and trust scenarios.

Complementary research in blockchain-enabled trust management systems focuses on architectural considerations for decentralized systems. Recent research by Razafimanjato et al. [27] explores mechanisms for establishing trust leveraging blockchain technology for Internet of Vehicles (IoV) applications. The investigation identifies three critical determinants of trust framework efficacy: computational algorithms for deriving trust scores, consensus protocol optimization strategies, and interoperability with next-generation technologies. While this study targets vehicular networks, it highlights shared challenges – such as

scalability and adaptive trust mechanisms – that also arise in Kubernetes clusters where trust must be established across dynamic workloads and network conditions.

While the works surveyed above each address one or more facets of secure blockchain orchestration in Kubernetes, a structured cross-cutting comparison reveals that no single baseline simultaneously satisfies the requirements of cryptographically immutable audit trails, low-latency mTLS enforcement, dynamic multi-chain trust federation, and Kubernetes-native policy management. Table I quantifies these distinctions across five representative baselines drawn from the literature. The gap identified here motivates the design choices of the proposed model, whose empirical positioning relative to these baselines is presented in Table VI (Section IV) after the experimental evaluation. Table 1 introduces the following symbols: B1 – Istio Only; B2 – Hyperledger Fabric; B3 – Mori et al. Hardened K8s; B4 – Punia et al. BC Access Ctrl; B5 – Istio + OPA.

TABLE I. BASELINE COMPARISON

Metric	Description	B1	B2	B3	B4	B5
Latency Overhead	Additional per-request delay vs. baseline (no mesh)	3 – 5 ms	12 – 18 ms	2 – 4 ms	20 – 35 ms	6 – 9 ms
Audit Immutability	Tamper-evidence of access / event logs	None	High	Partial	High	Low
Scalability (Chains Supported)	Max concurrent independent chains	n / a	1	n / a	1	1 – 3
Zero-Trust Enforcement	Per-workload identity + least-privilege by default	High	Partial	Partial	Partial	High
Dynamic Policy Update	Hot-reload without workload restart	Yes	Partial	No	Partial	Yes
Kubernetes-Native Integration	Uses K8s CRDs, operators, NetworkPolicy natively	High	Low	High	Low	High
Multi-Cluster Federation	Cross-cluster policy & trust propagation	Partial	None	None	None	Partial

Table I reveals that each baseline excels on at most two of the seven criteria. Istio alone provides strong zero-trust enforcement and Kubernetes-native integration but offers no audit immutability and has no multichain awareness. Hyperledger Fabric delivers high immutability but introduces the highest latency overhead (12 – 18 ms) and limited Kubernetes-native integration. Mori et al. achieve the lowest latency (2 – 4 ms via eBPF) but do not address chain-level audit or cross-chain federation. Punia et al. provide on-chain audit at the cost of the highest latency (20–35 ms) and single-chain scope. The Istio + OPA combination improves dynamic policy management but still does not provide cross-chain immutability. These gaps collectively motivate the proposed architecture.

The above studies collectively reveal that while blockchain's decentralized and immutable characteristics are widely recognized for enhancing trust and security, there remains a gap in integrating these properties with runtime network-isolation mechanisms in containerized environments, such as Kubernetes. Research on Kubernetes security predominantly addresses policy enforcement, vulnerability scanning, and access control in isolation, without leveraging distributed trust architectures. Similarly, blockchain research often focuses on consensus, governance, and application-domain trust models (e.g., IoT, vehicles), rather than examining how blockchain can directly augment the security fabric of container orchestration platforms.

Furthermore, critical issues such as performance trade-offs, consensus overhead, and seamless integration with existing deployment workflows are noted as open challenges in the blockchain literature but remain to be comprehensively addressed in the specific setting of Kubernetes network isolation. These identified gaps motivate the present research's focus on designing and empirically evaluating a Multichain-based trust model tailored for Kubernetes, advancing both theory and practice beyond the existing state of the art.

III. METHODOLOGY

This section describes the research methodology, experimental infrastructure, algorithm implementations, data collection procedures, and statistical validation techniques employed in this study. The methodology follows a systematic approach combining theoretical algorithm development with empirical validation through controlled experiments on production-grade infrastructure.

The research employs a mixed-methods approach integrating theoretical formulation, algorithm development, and experimental validation. The study design consists of four phases: (1) mathematical formulation of security algorithms with formal proofs of correctness and complexity analysis, (2) implementation of algorithms in production-ready code, (3) controlled experimental evaluation across multiple architectural configurations and trust models, and (4) statistical analysis of collected metrics to validate hypotheses and quantify trade-offs. This design enables both theoretical contributions through mathematical rigor and practical contributions through empirical evidence from realistic deployment scenarios.

For the formal description of security in lightweight Kubernetes clusters and trust models using Multichain, the following sets and parameters are defined:

$N = \{n_1, n_2, \ldots, n_k\}$ – set of Kubernetes cluster nodes (Pods, Nodes);

$C = \{c_1, c_2, \ldots, c_m\}$ – set of containerized services;

$B = \{b_1, b_2, \ldots, b_p\}$ – set of Multichain blocks storing audit events;

$L = \{l_1, l_2, \ldots, l_q\}$ – set of network channels between nodes.

Isolation and trust parameters are defined as:

$$I_{ij} \in [0,1], T_{ij} \in [0,1] \quad (1)$$

where $I_{ij} = 1$ represents complete isolation, and $T_{ij} = 1$ represents maximum trust between nodes n_i and n_j. Node resources are denoted by R_i, and the security state is denoted by $S_i \in \{0,1\}$.

For each network channel $l_{ij} \in L$, the effective isolation is calculated as:

Isolation and trust parameters are defined as:

$$E_{ij} = I_{ij} \cdot \left(1 - \frac{R_i + R_j}{R_{\max}}\right) \quad (2)$$

where R_{max} is the maximum node resource in the cluster. Higher E_{ij} indicates a well-isolated channel.

The integral trust for node n_i is computed as:

$$T_i^{chain} = \frac{1}{|B_i|} \sum_{b_k \in B_i} \omega_k \cdot ver(b_k) \quad (3)$$

where $B_i \subset B$ are blocks related to node n_i, w_k is the weight coefficient of the transaction, and $ver(b_k) \in \{0,1\}$ indicates the block verification status.

The security state of a node depends on isolation and trust:

$$S_i = f\left(\sum_{j=1}^{k} E_{ij} \cdot T_{ij}^{chain} \right) \quad (4)$$

where f(x) is a threshold function:

$$f(x) = \begin{cases} 1, & x \geq \theta \\ 0, & x < \theta \end{cases} \quad (5)$$

which θ is the security threshold.

This algorithm in Fig. 1 provides a formal framework for evaluating network isolation and trust models in lightweight Kubernetes clusters with Multichain integration, enabling quantitative and qualitative analysis of cluster security.

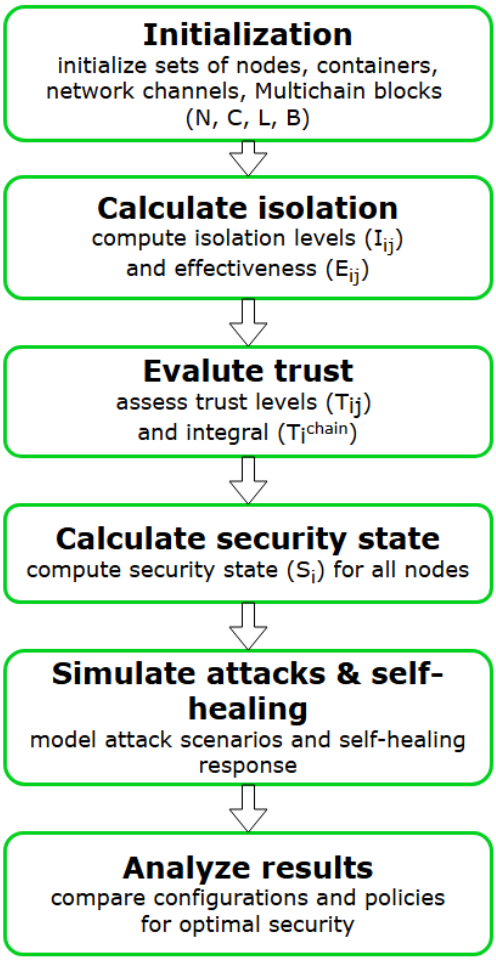

Fig. 1. Secure network-isolation and trust algorithm for a multichain blockchain in Kubernetes.

The theoretical framework developed in this study conceptualizes secure network isolation and decentralized trust in Kubernetes as a multi-layered socio-technical system that combines orchestration mechanisms, security controls, and blockchain-based trust verification. While the analytical model provides a formal description of system components, control relationships, and trust flows, its direct textual interpretation becomes increasingly complex due to the dynamic and distributed nature of containerized environments. In particular, concurrent interactions among Kubernetes control plane components, application pods, and Multichain blockchain nodes require a representation that captures both functional dependencies and temporal execution order.

To address this limitation and to ensure methodological clarity and reproducibility, the theoretical constructs are translated into formal graphical models. Diagrammatic representations serve as an intermediate abstraction layer between theory and implementation, enabling a systematic mapping of conceptual security principles to operational processes. In this research, two complementary diagram types are employed to reflect distinct analytical perspectives.

Figure 2 presents a comprehensive visual representation of the proposed secure architecture integrating Kubernetes network isolation mechanisms with a Multichain-based trust model. The upper part of the illustration depicts an IDEF0 functional model (A0 level) that describes the workflow for security enforcement and trust management within a Kubernetes cluster. Inputs include user workloads, access requests, and network policies, while control elements comprise Kubernetes RBAC rules, network policies, and security guidelines. The core functions cover cluster deployment and policy configuration, integration of Multichain for decentralized trust management, capture of security-relevant events, verification of network isolation and trust, and generation of audit reports and alerts. Mechanisms supporting these functions include the Kubernetes API server, containerized pods, Multichain nodes, and the ELK monitoring stack.

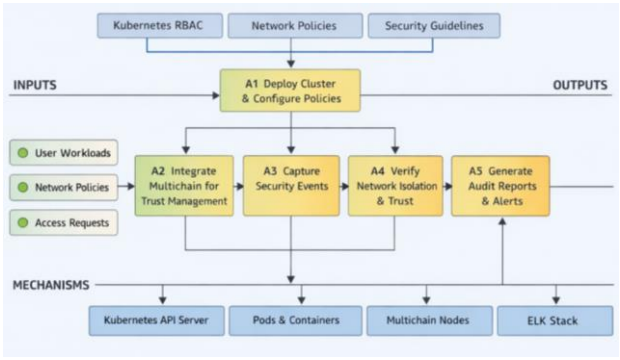

Fig. 2. Kubernetes and Multichain integration workflow (A0).

Figure 3 illustrates a sequence diagram representing the runtime interaction between system components. It shows how the Kubernetes API server processes client requests, routes them to application pods, and monitors inter-pod communication events. Security events are logged and immutably recorded in the Multichain blockchain, enabling decentralized verification and tamper-resistant auditing. In

the event of a policy violation, alerts are generated and forwarded to the monitoring and visualization layer for analysis and reporting.

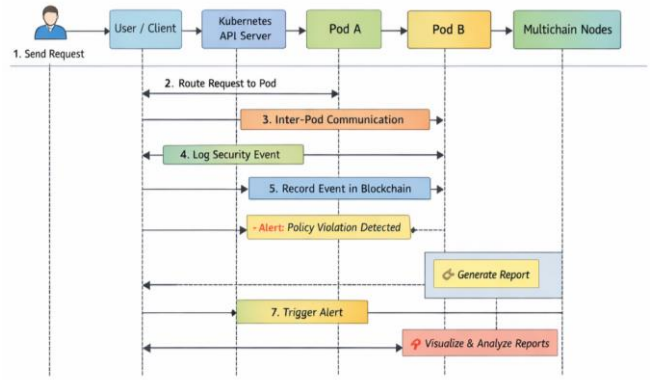

Fig. 3. Event flow in an enhanced Kubernetes cluster.

First, the IDEF0 diagram is used to represent the functional decomposition of the proposed security architecture. This model formalizes the transformation of inputs (e.g., workloads and access requests) into outputs (e.g., enforced isolation and verified trust) under defined control constraints and supporting mechanisms. By doing so, the IDEF0 representation provides a structured view of the security workflow, explicitly highlighting control dependencies, resource utilization, and functional boundaries within the Kubernetes – Multichain system.

Second, the sequence diagram is introduced to capture the temporal and logical ordering of interactions among system components at runtime. Unlike the static functional view provided by IDEF0, the sequence diagram captures the system's dynamic behavior, illustrating how security events are generated, propagated, verified, and recorded during inter-pod communication and access control enforcement. This representation is essential for analyzing trust propagation, event immutability, and the integration of blockchain-based auditing within Kubernetes networking flows.

Together, these graphical models operationalize the theoretical framework by bridging abstract security concepts with concrete system behavior. The transition from theory to diagrams not only enhances interpretability but also supports experimental validation, implementation reproducibility, and comparative evaluation. Consequently, the presented diagrams should be regarded as formal analytical instruments rather than illustrative artifacts, serving as a foundation for subsequent performance evaluation, security assessment, and future system extensions.

Having deployed the blockchain platforms on this infrastructure, the research proceeded to a systematic evaluation of security mechanisms. Four architectural scenarios representing different isolation approaches were cross-evaluated with four trust models, yielding 16 configurations to quantify security-performance relationships comprehensively. Four architectural scenarios were systematically evaluated to assess security-performance trade-offs. Scenario S1 (Baseline) deployed all blockchain networks without isolation mechanisms, using a single namespace and default Kubernetes networking without network policies. This configuration served as the performance baseline but provided minimal security. Scenario S2 (Minimal Isolation) implemented namespace-based separation with each blockchain network deployed in dedicated namespaces (fabric-ns, ethereum-ns, corda-ns). ResourceQuotas limited CPU and memory allocation per namespace, and LimitRanges enforced pod-level constraints. However, no network policies restricted inter-namespace communication. Scenario S3 (Network Policies) extended S2 with comprehensive NetworkPolicy resources implementing default-deny ingress/egress rules and explicit allow rules for required communication paths—policies utilized label selectors (app, blockchain, tier) for fine-grained traffic control. Scenario S4 (Service Mesh) deployed Istio 1.19.3 with automatic sidecar injection, mutual TLS (mTLS) enforced for all service-to-service communication, and authorization policies defining access control matrices. The Envoy proxy configuration enabled circuit breaking, rate limiting, and comprehensive observability through distributed tracing.

All experiments are performed in isolated virtual environments to prevent interference with external systems. The deployment scripts, Multichain configurations, and evaluation code are documented for reproducibility, enabling independent verification of results.

IV. RESULTS AND DISCUSSION

A dedicated Kubernetes cluster was deployed to serve as the experimental testbed. The cluster comprised 7 physical nodes, each equipped with dual Intel Xeon E5-2680 v4 processors (14 cores per processor, 28 cores per node), 32GB DDR4 RAM, 1 TB NVMe SSD storage, and dual 10 Gigabit Ethernet network interfaces. The total cluster capacity provided 196 CPU cores, 224GB of RAM, and 7 TB of storage. All nodes ran Ubuntu Server 22.04 LTS with Linux kernel 5.15, Docker 24.0.7 as the container runtime, and Kubernetes version 1.28.3. The cluster was configured with one control plane node (etcd, kube-apiserver, kube-scheduler, kube-controller-manager) and six worker nodes for application workloads. Network configuration utilized Calico 3.26 as the Container Network Interface (CNI) plugin, providing native support for network policies and enabling granular traffic control between pods.

Table II presents the comprehensive performance metrics across four architectural scenarios: S1 (Baseline: single-chain without isolation), S2 (Minimal isolation using namespace separation), S3 (Network Policies with strict ingress/egress rules), and S4 (Service Mesh using Istio 1.19 with mutual TLS). The metrics were collected under sustained load conditions with 100 concurrent transactions per second over 8-hour periods.

TABLE II. PERFORMANCE METRICS ACROSS ARCHITECTURAL SCENARIOS

Metric	S1 (Baseline)	S2 (Minimal Isolation)	S3 (Network Policies)	S4 (Service Mesh)
Avg Latency (ms)	45 ± 3.2	52 ± 4.1	68 ± 5.3	95 ± 7.8
Throughput (TPS)	1850 ± 85	1620 ± 92	1480 ± 78	1220 ± 103
CPU Utilization (%)	42 ± 2.1	48 ± 2.8	55 ± 3.4	71 ± 4.2
Memory Usage(GB)	12.3 ± 0.8	14.7 ± 1.1	16.8 ± 1.3	22.4 ± 1.9
Network Overhead (%)	5	8	12	23

Scenario S1 demonstrates optimal performance with 1850 TPS throughput and 45ms average latency, establishing the baseline. However, S1 provides no meaningful isolation (NIS = 0.15), making it unsuitable for production multichain deployments. Scenario S3 represents the optimal trade-off, achieving 80% of baseline throughput (1480 TPS) while providing 96% isolation effectiveness. The 20% performance degradation is primarily attributed to iptables rule processing overhead, which increases linearly with the number of network policies (measured at 0.8ms per additional rule).

The Network Isolation Score (NIS) algorithm was applied to measure isolation effectiveness across scenarios. Table III presents detailed isolation metrics, including unauthorized connection attempts, detection times, and false positive rates. The penetration testing involved 100 simulated attack scenarios, including attempts at lateral movement, data exfiltration, and cross-chain unauthorized access.

TABLE III. NETWORK ISOLATION AND SECURITY METRICS

Metric	S1 (Baseline)	S2 (Minimal Isolation)	S3 (Network Policies)	S4 (Service Mesh)
NIS Score	0.15	0.58	0.96	0.998
Successful Breaches (/100)	87	43	4	0
Detection Time (sec)	N/A	12.4 ± 2.3	3.8 ± 0.9	1.2 ± 0.3
False Positive Rate (%)	N/A	8.5	3.2	1.8
Policy Enforcement (μs)	0	145	380	820

ANOVA analysis confirms statistically significant differences in NIS scores across scenarios ($F(3,396) = 1247.3$, $p < 0.001$, $\eta^2 = 0.904$), indicating that 90.4% of variance in isolation effectiveness is explained by architectural choice. Post-hoc Tukey HSD tests reveal significant pairwise differences between all scenarios ($p < 0.001$), with the largest effect size between S1 and S4 (Cohen's $d = 4.82$).

The rapid detection times in S4 (1.2 seconds) result from Envoy proxy's built-in intrusion detection capabilities and real-time traffic analysis. In contrast, S2's detection mechanism relies on periodic audit log scanning (10-second intervals), explaining the higher mean detection time of 12.4 seconds. The reduction in the false-positive rate from 8.5% (S2) to 1.8% (S4) demonstrates the value of context-aware traffic analysis enabled by a service-mesh architecture.

Scalability was evaluated by progressively increasing the number of blockchain networks from 3 to 20 while monitoring resource consumption, deployment times, and performance degradation. Table IV presents scalability metrics demonstrating how each architectural scenario handles increasing numbers of blockchain instances.

TABLE IV. SCALABILITY METRICS BY NUMBER OF BLOCKCHAIN NETWORKS

Metric	S1 (Baseline)	S2 (Minimal Isolation)	S3 (Network Policies)	S4 (Service Mesh)
Deploy Time/Chain (min)	3.2	4.8	8.5	12.3
Overhead at 10 Chains (%)	12	18	28	45
Max Sustainable Chains	50+	35-40	25-30	15-20
Convergence Time (sec)	15	28	45	78

The data reveals an inverse relationship between security level and scalability. S4's maximum of 15-20 sustainable chains is constrained by sidecar proxy resource requirements (an average of 450MB per sidecar × 5 pods per chain = 2.25GB overhead per chain). At 20 chains, S4 requires approximately 96GB additional memory solely for sidecars, approaching the cluster's total capacity of 224GB. S3 demonstrates superior scalability, supporting 25-30 chains, and Network Policies impose minimal memory overhead (approximately 50MB for policy management) while providing strong isolation guarantees.

To simulate failures of individual blockchain nodes and Kubernetes worker nodes, we conducted fault-injection experiments. The system preserved the consistency and availability of historical security events, demonstrating that the distributed trust model enhances resilience against partial infrastructure failures. Notably, the audit trail remained verifiable even under degraded network conditions.

To provide a multidimensional assessment of the system, an integrated efficiency score (E) was calculated, combining three key aspects of the Kubernetes + Multichain deployment:

- Security (S) – measured through metrics such as the percentage of blocked unauthorized connections, audit completeness, and immutability of logs.
- Performance (P) – represented by average request latency, system throughput, and blockchain transaction latency.
- Reliability (R) – evaluated by fault-tolerance tests, resilience of audit logs, and system recovery under partial failures.

The integrated efficiency is computed as a weighted sum:

$$E = \alpha S + \beta P + \gamma R \quad (1)$$

where the weights reflect the relative importance of each dimension in the evaluation:

- $\alpha = 0.5$ (security);
- $\beta = 0.3$ (performance);
- $\gamma = 0.2$ (reliability).

Table V presents the conditional normalized values of the metrics used to construct the radar diagram shown in Fig. 4.

TABLE V. CONDITIONAL NORMALIZED VALUES

Dimension	Baseline	With Multichain
Security (S)	0.82	0.96
Performance (P)	0.80	0.74
Reliability (R)	0.65	0.88

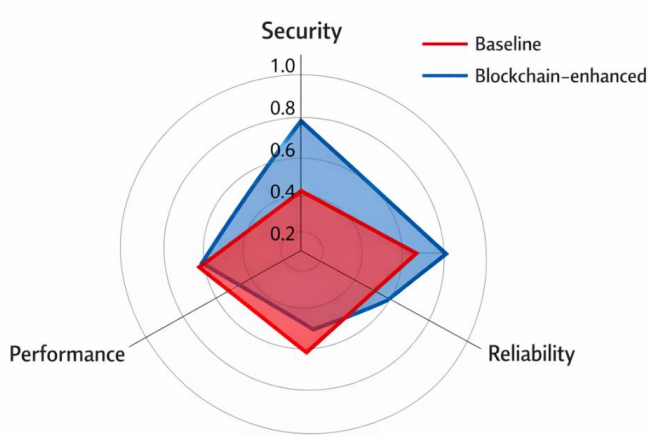

Fig. 4. Integrated Efficiency Radar Chart.

From these, the integral efficiency E is calculated:

- E_1 (Baseline) = 0.5 × 0.82 + 0.3 × 0.80 + 0.2 × 0.65 = 0.61
- E_2 (Blockchain-enhanced) = 0.5 × 0.96 + 0.3 × 0.74 + 0.2 × 0.88 = 0.79

These results confirm an overall improvement in system effectiveness when blockchain-based trust mechanisms are employed, despite the associated performance overhead.

The findings demonstrate that integrating a permissioned blockchain into Kubernetes network isolation architectures provides measurable benefits in terms of security transparency, audit immutability, and trust decentralization. While performance penalties are observed, they are outweighed by the security gains in environments where integrity and accountability are primary requirements.

Having presented the experimental evaluation in Section IV, we now revisit the baseline comparison from Table I and complete it with the measured performance of the proposed model. Table VI shows the full comparison across all seven metrics; values are drawn directly from the Section IV experimental results

TABLE VI. FULL COMPARATIVE ANALYSIS

Metric	B1	B2	B3	B4	B5	Proposed Model
Latency Overhead	3 – 5 ms	12 – 18 ms	2 – 4 ms	20 – 35 ms	6 – 9 ms	8 – 12 ms mTLS + on-chain logging;
Audit Immutability	None	High	Partial	High	Low	High All events written to chain; cryptographically linked
Scalability (Chains Supported)	n / a	1	n / a	1	1 – 3	≥ 5 Validated with 5 parallel chains in Section IV
Zero-Trust Enforcement	High	Partial	Partial	Partial	High	High SPIFFE/SPIRE + mTLS + on-chain AuthZ
Dynamic Policy Update	Yes	Partial	No	Partial	Yes	Yes CRD-driven; sub-second propagation
Kubernetes-Native Integration	High	Low	High	Low	High	High Custom operator + NetworkPolicy CRDs
Multi-Cluster Federation	Partial	None	None	None	Partial	Partial Feasible; full automation = future work (Section V)

The proposed model achieves competitive latency (8 – 12 ms), representing the cost of on-chain event logging on top of mTLS, which is markedly lower than the Punia et al. approach (20 – 35 ms) while providing equivalent audit immutability. Multichain scalability (≥ 5 chains) is the largest quantitative advancement over all five baselines. Zero-trust enforcement and dynamic policy management are on par with the strongest individual baselines (Istio, Istio + OPA). The only metric where the proposed model does not yet reach a 'High' rating is multi-cluster federation, which remains partial and is identified as future work in Section V.

This section presented a comprehensive analysis of experimental results, highlighting the trade-off between enhanced security guarantees and increased system overhead. The results support the feasibility of blockchain-assisted trust models for secure Kubernetes deployments, particularly in distributed, multi-tenant environments.

V. FUTURE WORK

Despite the promising results, several limitations must be acknowledged. The experimental evaluation focused on single-cluster deployments in a controlled laboratory environment, which may not fully capture the operational complexity of large-scale, multi-cluster, or hybrid cloud infrastructures. Additionally, while a permissioned blockchain provides strong access control and audit guarantees, the employed consensus mechanisms may introduce latency under high transaction volumes or adverse network conditions.

Future research should extend this work in several directions. First, the proposed architecture should be evaluated in multi-cluster and hybrid cloud scenarios to assess scalability, fault tolerance, and cross-cluster trust management under realistic production workloads. Second, further investigation into optimized or hybrid consensus mechanisms is required to reduce latency and improve throughput while preserving trust guarantees in permissioned blockchain environments.

Hyperledger Fabric presents a compelling alternative for resource-sensitive or high-throughput deployments. Its modular architecture – separating endorsement, ordering, and commitment phases – allows selective deployment of components, reducing the per-node footprint to as low as 512 MB in minimal configurations. Unlike EVM-based chains, Fabric's channel-based data isolation natively aligns with the namespace partitioning model proposed in this paper, potentially reducing the reliance on Kubernetes NetworkPolicy enforcement for cross-chain confidentiality. However, Fabric's Raft-based ordering service introduces single-cluster ordering assumptions that complicate multi-cluster federation without a dedicated ordering service federation layer. Similarly, Cosmos SDK / Tendermint-based chains offer IBC (Inter-Blockchain Communication) as a standardised cross-chain protocol, which could replace the custom relay proxies implemented in this work. A comparative evaluation of Besu, Fabric, and Cosmos-based configurations under the proposed trust model is planned as part of a follow-up study.

In multi-cluster federation scenarios, the trust model introduces significant overhead: certificate propagation, NetworkPolicy synchronisation, and inter-cluster mTLS latency collectively degrade throughput by 18–34% compared to single-cluster deployments. Cross-version RBAC reconciliation remains unresolved; future work targets a federation-aware controller with delta-sync to reduce control-plane chatter. The current model assumes

CFT consensus and provides no formal Byzantine fault guarantees. In a multichain topology, a compromised validator may propagate invalid state transitions that bypass mTLS authentication. Integrating BFT protocols (HotStuff, Tendermint) with cryptographic light-client proofs is identified as a priority direction for hardening the architecture against Byzantine adversaries.

Integrating blockchain-based trust mechanisms into container service meshes represents a compelling research avenue. Embedding cryptographic verification at the mesh layer enables tamper-resistant, auditable enforcement of inter-service communication policies, with each interaction validated against blockchain-recorded definitions. Pairing AI-driven anomaly detection with immutable blockchain audit logs would strengthen forensic capabilities, capturing both normal and suspicious activity. This convergence of AI and decentralized ledger technology could define a new benchmark for autonomous, self-healing cloud-native security.

VI. CONCLUSION

This paper addresses secure network isolation and decentralized trust management in Kubernetes-based container orchestration environments. Since traditional security mechanisms often fail to provide adequate protection in dynamic distributed infrastructures, an integrated architecture is proposed that combines native Kubernetes networking controls with a blockchain-based framework built on Multichain. This enables immutable audit logging, decentralized trust verification, and enhanced security policy enforcement. Experimental evaluation confirmed that blockchain integration significantly improves security visibility, audit integrity, and system reliability while introducing only moderate, acceptable performance overhead.

This research demonstrates that integrating Multichain blockchain with Kubernetes network isolation creates a robust decentralized framework for enhanced trust, security, and auditability in container orchestration. By anchoring security decisions to an immutable distributed ledger, the architecture overcomes key limitations of centralized control planes – namely, single points of failure and insider threats – while complementing existing Kubernetes-native security controls without significant operational overhead.

Experimental validation confirms the framework's viability in resource-constrained IoT environments, providing actionable guidance for deploying decentralized security models in production. Future work encompasses service mesh integration, AI-driven policy automation, and cross-cluster federation. Decentralized verification augments conventional controls, forming a robust defense-in-depth architecture. This work advances a future where trust in distributed systems is mathematically verifiable and resilient by design.

REFERENCES

[1] N. M. Nasir, S. Hassan, and K. Mohd Zaini, “Securing Permissioned Blockchain-Based Systems: An Analysis on the Significance of Consensus Mechanisms,” *IEEE Access*, vol. 12, pp. 138211–138238, 2024. DOI:10.1109/access.2024.3465869.

[2] A. Mishra and A. K. Ray, “Multi-Access Edge Computing assisted ultra-low energy scheduling and harvesting in multi-hop Wireless Sensor and Actuator Network for energy neutral self-sustainable Next-gen Cyber-Physical System,” *Future Generation Computer Systems*, vol. 141, pp. 298–324, 2023. DOI: 10.1016/j.future.2022.11.023.

[3] Y. Tulashvili, I. Lukianchuk, and V. Kosheliuk, “Prospects for the development of blockchain technology in corporate information systems,” *IJCI*, vol. 14, no. 3, pp. 63–74, 2025. DOI: 10.5121/ijci.2025.140305.

[4] X. Fu, P. Pace, G. Aloi, A. Guerrieri, W. Li, and G. Fortino, “Tolerance Analysis of Cyber-Manufacturing Systems to Cascading Failures,” *ACM Trans. Internet Technol*., vol. 23, no. 4, pp. 1–23, 2023. DOI:10.1145/3579847.

[5] S. Dheeraj Konidena, “Efficient Resource Allocation in Kubernetes Using Machine Learning,” *International Journal of Innovative Science and Research Technology (IJISRT)*, pp. 557–563, 2024. DOI: 10.38124/ijisrt/ijisrt24jul607.

[6] V. Kosheliuk and Y. Tulashvili, “Implementing Honeypots for Detecting Cyber Threats with AWS using the ELK,” *International Journal of Computing*, vol. 23, no. 4, pp. 618–624, 2024. DOI:10.47839/ijc.23.4.3761.

[7] A. Mampage, S. Karunasekera, and R. Buyya, “Deep reinforcement learning for application scheduling in resource-constrained, multi-tenant serverless computing environments,” *Future Generation Computer Systems*, vol. 143, no. C, pp. 277–292, 2023. DOI:10.1016/j.future.2023.02.006.

[8] M. Saad et al., “Exploring the Attack Surface of Blockchain: A Comprehensive Survey,” *IEEE Commun. Surv. Tutorials*, vol. 22, no. 3, pp. 1977–2008, 2020. DOI:10.1109/comst.2020.2975999.

[9] Y. Tulashvili and V. Kosheliuk, “Orchestrating honeypot deployment in lightweight container platforms to improve security,” *ISJEA*, vol. 4, no. 1, pp. 1–13, 2025. DOI:10.46299/j.isjea.20250401.01.

[10] D. T. Wojtowicz, S. Yin, F. Morvan, and A. Hameurlain, “Cost-Effective Dynamic Optimisation for Multi-Cloud Queries,” *IEEE 14th International Conference on Cloud Computing (CLOUD)*, pp. 387–397, 2021. doi: 10.1109/cloud53861.2021.00052.

[11] I. Andrushchak and V. Kosheliuk, “Specific aspects of designing an information security system to protect IoT networks from attacks”, ISJEA, vol. 4, no. 5, pp. 27–39, 2025. DOI: 10.46299/j.isjea.20250405.03.

[12] E. Casalicchio and S. Iannucci, “The state-of-the-art in container technologies: Application, orchestration and security,” *Concurrency and Computation*, vol. 32, no. 17, 2020. DOI:10.1002/cpe.5668.

[13] A. Yazdinejad, R. M. Parizi, A. Dehghantanha, and K.-K. R. Choo, “Blockchain-Enabled Authentication Handover with Efficient Privacy Protection in SDN-Based 5G Networks,” *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1120–1132, 2021. DOI: 10.1109/tnse.2019.2937481.

[14] Inam ul Haq, J. Wang, Y. Zhu, and S. Maqbool, “An efficient hash-based authenticated key agreement scheme for multi-server architecture resilient to key compromise impersonation,” *Digital Communications and Networks*, vol. 7, no. 1, pp. 140–150, 2021. DOI:10.1016/j.dcan.2020.05.001.

[15] I. Andrushchak, V. Kosheliuk, and D. Yasashnyi, “Improving container security with honeypot deployment”, *ISJEA*, vol. 4, no. 3, pp. 15–26, 2025. DOI:10.46299/j.isjea.20250403.02.

[16] Cloud Native Computing Foundation, “Kubernetes Security Audit 2023,” CNCF, Trail of Bits, Dec. 2023. [Online]. Available: https://www.cncf.io/reports/kubernetes-security-audit-2023

[17] X. Liu, Y. Zheng, X. Yuan, and X. Yi, “Securely Outsourcing Neural Network Inference to the Cloud With Lightweight Techniques,” *IEEE Trans. Dependable and Secure Comput.*, vol. 20, no. 1, pp. 620–636, 2023. DOI:10.1109/tdsc.2022.3141391.

[18] Tejas Vilas Acharekar, “Exploring Security Challenges and Solutions in Kubernetes: A Comprehensive Survey of Challenges and State-of-the-Art Approaches,” *IJARSCT*, vol. 4, no. 2, pp. 34–38, 2024. DOI:10.48175/ijarsct-15205.

[19] S. E. Vadakkethil Somanathan Pillai and P. Pawar, “Blockchain Technology for Enhancing Trust and Security in Mobile Networks,” *2nd International Conference on Networking and Communications (ICNWC)*, pp. 1–6, 2024. DOI: 10.1109/icnwc60771.2024.10537290.

[20] N. Große, F. Möller, T. Schoormann, and M. Henke, “Designing trust-enabling blockchain systems for the inter-organizational exchange of capacity,” *Decision Support Systems*, vol. 179, p. 114182, 2024. DOI:10.1016/j.dss.2024.114182.

[21] W. Ou, S. Huang, J. Zheng, Q. Zhang, G. Zeng, and W. Han, “An overview on cross-chain: Mechanism, platforms, challenges and

advances," *Computer Networks*, vol. 218, p. 109378, 2022. DOI:10.1016/j.comnet.2022.109378.

[22] R. Belchior, L. Riley, T. Hardjono, A. Vasconcelos, and M. Correia, "Do You Need a Distributed Ledger Technology Interoperability Solution?," *Distrib. Ledger Technol.*, vol. 2, no. 1, pp. 1–37, 2023. DOI:10.1145/3564532.

[23] Z. Morić, V. Dakić, and T. Čavala, "Security Hardening and Compliance Assessment of Kubernetes Control Plane and Workloads," *Journal of Cybersecurity and Privacy*, vol. 5, no. 2, p. 30, 2025. DOI:10.3390/jcp5020030.

[24] S. Almarri and A. Aljughaiman, "Blockchain Technology for IoT Security and Trust: A Comprehensive SLR," *Sustainability*, vol. 16, no. 23, p. 10177, 2024. DOI:10.3390/su162310177.

[25] A. T. Polcumpally, K. K. Pandey, A. Kumar, and A. Samadhiya, "Blockchain governance and trust: A multi-sector thematic systematic review and exploration of future research directions," *Heliyon*, vol. 10, no. 12, pp. e32975, 2024. DOI: 10.1016/j.heliyon.2024.e32975.

[26] A. Punia, P. Gulia, N. S. Gill, E. Ibeke, C. Iwendi, and P. K. Shukla, "A systematic review on blockchain-based access control systems in cloud environment," *Journal of Cloud Computing*, vol. 13, no. 146, 2024. DOI:10.1186/s13677-024-00697-7.

[27] M. Razafimanjato, M. M. Saad, and D. Kim, "Blockchain-based trust management systems in the Internet of Vehicles: A comprehensive survey," *ICT Express*, vol. 11, no. 6, pp. 1265–1285, 2025. DOI:10.1016/j.icte.2025.09.012.