Embedding Security Awareness into a Blockchain-Based Dynamic Access Control Framework for the Zero Trust Model in Distributed Systems

Avoy Mohajan and Sharmin Jahan *

Posted Date: 7 February 2025

doi: 10.20944/preprints202502.0498.v1

Keywords: zero trust; dynamic access control; blockchain; security awareness

 doi:10.20944/preprints202502.0498.v1

Article

Embedding Security Awareness into a Blockchain-Based Dynamic Access Control Framework for the Zero Trust Model in Distributed Systems

Avoy Mohajan [1] **and Sharmin Jahan** [2,*]

[1] Affiliation 1
[2] Affiliation 2
* sharmin.jahan@okstate.edu

Abstract: The Zero Trust (ZT) model is pivotal in enhancing the security of distributed systems by emphasizing rigorous identity verification, granular access control (AC), and continuous monitoring. To address the complexity and scalability challenges of modern distributed systems, we propose a blockchain-based dynamic access control scheme (DACS) as a practical solution for implementing ZT principles. This framework dynamically manages access control lists (ACLs) and enforces policies through smart contracts. In the DACS framework, each blockchain node maintains an object list specifying access permissions within its ACL and incorporates a minimum trust metric (TM) threshold to evaluate access requests. The TM assigned to each node reflects its trustworthiness. To further enhance security, the framework includes security awareness, enabling the dynamic assessment of the risk factor (RF), which reflects the operational risk level. The TM of access-requesting nodes is updated at runtime based on their behavior, with penalties imposed for malicious actions according to the prevailing RF. Access control policies are dynamically adjusted, mitigating risks posed by potentially untrustworthy users with valid credentials. Implemented and tested on the Ethereum blockchain, the proposed DACS framework demonstrates its efficiency and effectiveness in securing distributed systems.

Keywords: zero trust; dynamic access control; blockchain; security awareness

1. Introduction

The proliferation of distributed systems has transformed modern computing, introducing scalable and decentralized architectures that enhance performance, resilience, and availability [1]. However, as the number of interconnected nodes and services grows, security challenges have become increasingly complex. Traditional perimeter-based security models are no longer sufficient to safeguard these intricate environments [2,3]. This shift has driven the adoption of the Zero Trust (ZT) security model, which operates on the principle of "never trust, always verify" [2,4]. In the ZT model, every user, device, and service must continuously authenticate its identity and access rights, irrespective of its location or position within the network.

Implementing ZT in distributed environments requires a context-aware approach to security, making dynamic access control schemes (DACS) [5] essential. Unlike static policies, DACS adaptively evaluates various factors such as an actor's (e.g., user, device, or service) identity, service request patterns, and real-time threat intelligence to make in-formed, real-time access decisions [6,7]. This continuous evaluation ensures that access rights can be granted or revoked as conditions evolve [7,8], providing a flexible and robust solution for securing distributed systems under the ZT model [6].

 doi:10.20944/preprints202502.0498.v1

DACS represents a paradigm shift from rigid, predefined policies to flexible, adaptive access management strategies. Effective DACS implementation depends on robust policy management [6,9,10], which includes key components such as: Policy Enforcement Point (PEP) that intercepts and evaluates access requests, Policy Administration Point (PAP), which defines, updates, and enforces access rules in alignment with organizational requirements, compliance standards, and security contexts using Access Control Lists (ACLs), and Policy Decision Point (PDP) is for evaluating access requests based on ACLs and contextual in-formation to determine appropriate actions to adjust the access policy. Well-designed DACS policy management ensures seamless coordination between ACL policies and re-al-time security events [6,7], significantly enhancing system adaptability and resilience against emerging threats. With the increased interaction among the components in distributed system, DACS policy management needs to be autonomous and includes security awareness to reason over the access policy adjustment as per the risk level in operational context [6,7]. Security awareness is a form of self-awareness [11,12], defined as the knowledge that enables the ability to investigate a system's or interacting actor's behavior to evaluate system's security state, detect and assess changes in security states, and reason about potential adjustments needed to maintain a secure state [13,14].

Continuous monitoring and risk management are critical components to embed security awareness in DACS for implementing the Zero Trust (ZT) model [4]. Continuous monitoring allows for the early detection of emerging risks, such as compromised devices or insider threats, as they occur. Risk management incorporates a proactive risk assessment mechanism to evaluate potential threats and provide actionable insights for mitigation, thereby strengthening the overall security posture. Security awareness in DACS enables the ability to evaluating access control decisions in real time, adjusting access per-mission based on the latest risk indicators [6–8,15]. This security aware approach minimizes the attack surface and protects sensitive resources by ensuring that access privileges remain appropriate and do not escalate into security threats. Traditional centralized policy management systems, however, often struggle to balance the dynamic, scalable, and context-aware requirements of the ZT model [16]. Decentralized policy management using blockchain technology addresses these challenges by storing policies and access logs on an immutable ledger [7,17]. This tamper-proof system ensures that policies remain transparent and auditable, creating an unalterable record of access control decisions. By integrating blockchain with access control systems, organizations can establish a ZT framework where smart contracts govern access decisions, continuously validated through consensus mechanisms [17]. This setup provides a secure, resilient, and scalable solution for managing access in distributed Zero Trust environments.

This paper introduces a DACS framework with embedded security awareness for implementing the ZT model in distributed systems by leveraging blockchain technology. The framework's novelty lies in enhancing smart contract functionality to enable continuous risk assessment and runtime policy adjustments, thereby eliminating the need for specific trusted nodes to act as policy management units or conduct risk assessments for other nodes. In [7,18,19], authors have employed blockchain technology to enforce DACS; however, they rely on some distinguished trusted nodes deployed in the network to monitor access requests and manage policies. This reliance renders the trusted nodes attractive targets for attackers. Moreover, these approaches lack a mechanism to verify the actions of trusted nodes, which contradicts the core principles of the ZT model.

In our proposed framework, each blockchain node is equipped with policy management capabilities for its own resources (referred to as objects) through smart contracts that reference the ACL, specifying which nodes in the blockchain have access permissions and the permitted operations for each object. ACL also includes impact levels associated with those operations, where higher impact levels signify greater potential harm to the system in cases of unauthorized access. Organizations define these impact levels based on the potential damage unauthorized operations could cause [20], aligning them with their business security requirements. Researchers in [7,21] introduced trust values to quantify the trustworthiness of actors during access decisions, they did not consider the impact levels of unauthorized access attempts. Our framework extends smart contract

 doi:10.20944/preprints202502.0498.v1

capabilities to perform ongoing risk assessments for every access request, dynamically adjusting access policies based on behavior analysis and contextual risk. The current contextual risk for a node is evaluated by analyzing incoming access requests within an organization-defined time window. Risk increases as the number of unauthorized access requests grows, indicating potential broken access control attacks [22], where attackers exploit legitimate nodes or infiltrate the network as insiders to probe for vulnerabilities.

To quantify risk, we introduce a metric called the Risk Factor (RF) and its probability estimation. Each blockchain node is assigned a Trust Metric (TM), which reflects its trustworthiness based on its operational context and behavioral actions. A penalty enforcement mechanism is triggered for anomalous behaviors, considering both the RF and the impact levels of unauthorized access attempts. Prior research [6,7,17] has explored penalty enforcement mechanisms, but these typically rely on predefined penalties that fail to account for the dynamism of contextual risk. Such approaches are insufficient to address the uncertainty posed by evolving attack surfaces and the dynamic interactions in distributed systems. Organizations also define a threshold TM for each operation, aligning the required trust level with the operation's impact level. The policy management mechanism evaluates each access request by referencing both the ACL and the requester's TM. Access permission is granted or denied based on a comparison with the threshold value assigned to the requested operation, enabling dynamic policy adjustments in real time. This penalty enforcement and risk-aware policy adjustment approach aligns with the ZT principle, delivering an adaptive access control solution tailored to the organization's security and business requirements. To ensure transparency and resilience, we adopt the Proof of Stake (PoS) consensus mechanism [23] to validate each transaction, including policy adjustments and TM updates, supporting the ZT model. To evaluate the framework, we implemented a testbed on the Ethereum blockchain platform. Smart contracts were developed using the Remix IDE, demonstrating the framework's effectiveness in providing resilient and adaptable access control.

2. Background

Distributed systems, composed of diverse components, applications, and services operating across varied environments, present unique and pressing security challenges [1]. Traditional security models, rooted in the outdated concept of a trusted perimeter, were once effective but now fail to meet the demands of increasingly decentralized infra-structures [2]. Their reliance on implicit trust and static access controls creates critical vulnerabilities, leaving systems exposed to insider threats, unauthorized access, and operational inefficiencies [3]. Addressing these challenges requires a paradigm shift toward innovative, adaptable, and scalable security frameworks [4]. The Zero Trust (ZT) security model has emerged as a transformative approach to securing distributed systems [4,17]. Emphasizing the principles of "never trust, always verify," ZT validates user identities, continuously monitors behavior, and enforces fine-grained access controls across all network resources. ZT principles have been successfully implemented across various domains, including healthcare [24], supply chains [25], and communication networks [26,27].

Despite its promise, implementing ZT in distributed systems introduces complexities, particularly regarding scalability and reliance on centralized security administration. The advancement of blockchain technology addresses these challenges by providing a decentralized, tamper-proof foundation for managing security policies [26,27]. Block-chain-based ZT solutions enable secure access to resources while dynamically adjusting access control to account for changing operational contexts.

One critical enhancement to ZT is the integration of Dynamic Access Control Scheme (DACS), which adapts access permissions in real-time based on actors' behavior, attack patterns, and evolving network or system architecture [6–8]. DACS incorporate security mechanisms that extend traditional ACLs with contextual attributes, enabling flexible and adaptive access control. Ali et al. [29] developed the d-CAP framework, an ML-based dynamic ACL system for Software-Defined Networks (SDNs), which optimizes access con-troll rules in real-time, reducing latency and

 doi:10.20944/preprints202502.0498.v1

processing overheads. Similarly, Jung et al. [30] proposed PortCatcher, a scalable architecture that enhances ACL rule management using a TCAM-SRAM hybrid design, maximizing space efficiency while minimizing latency. To effectively implement DACS, robust policy management is essential. This includes components like the Policy Enforcement Point (PEP), which intercepts and evaluates access requests; the Policy Administration Point (PAP), which defines and updates access rules in alignment with security contexts and compliance standards; and the Policy Decision Point (PDP), which dynamically evaluates access requests to adjust policies based on real-time contextual information [6,9,10]. DACS must also embed security awareness, enabling systems to continuously evaluate their security state, detect threats, and adapt policies accordingly [11,12,14].

Blockchain-based distributed DACS further addresses key limitations of centralized systems, such as single points of failure and lack of auditability. Sun et al. [32] introduced a Blockchain-enabled Provenance-based Dynamic Access Control (BPDAC) scheme, which uses smart contracts to automate access-related decision-making while maintaining decentralized governance. Nakamura et al. [33] demonstrated an Ethereum-based Capability-Based Access Control (CapBAC) system that manages permissions with granular control, offering flexibility and security in hierarchical organizations. Gong et al. [5] proposed SDACS, a blockchain-powered architecture for IoT systems based on Hyperledger Fabric and IPFS, leveraging Attribute-Based Access Control (ABAC) to ensure fine-grained and decentralized access management. The combination of blockchain and DACS offers a promising future for access control in distributed systems. By eliminating reliance on trusted third parties and central authorities, these technologies empower organizations to implement decentralized, scalable, and context-aware security policies. Research has shown that blockchain can revolutionize access control by enabling secure policy enforcement under complex and dynamic conditions [34,35]. The integration of ZT principles, DACS, and blockchain technologies marks a critical evolution in distributed systems security.

3. Approach

This paper presents a DACS framework with embedded security awareness designed to implement a ZT model in distributed systems, utilizing blockchain technology for enhanced security and reliability. This section provides a detailed explanation of the core components of the framework, including blockchain nodes and smart contract functionalities, which are integral to the system's operations. The framework leverages blockchain nodes to enable decentralized and tamper resistant logging, ensuring that all access attempts are continuously monitored and recorded. Smart contracts facilitate real-time detection of unauthorized access attempts and perform runtime risk assessments. These assessments evaluate the RF of the operational context by analyzing various parameters, such as the node's trustworthiness, access history, and behavioral patterns.

The policy management functionality within the framework is a cornerstone of its ZT implementation. It includes mechanisms to process and enforce access requests based on an ACL and the TM associated with the participating nodes. This functionality ensures that access decisions are dynamically informed by the most current contextual and trust-related data. Additionally, the policy management module incorporates dynamic adaptability to evolving security contexts. It evaluates and updates the TM of the requesting nodes, reflecting changes in their behavior or operational environment. Furthermore, the system dynamically modifies access policies in response to these changes, ensuring a robust and responsive access control mechanism that aligns with ZT principles. By combining continuous monitoring, runtime risk assessment, and adaptive policy management, the proposed DACS framework provides a comprehensive solution for securing distributed systems while maintaining operational flexibility and resilience against emerging threats.

3.1. Define Node in Proposed Blockchain-Based DACS Framework

In our blockchain infrastructure, any active actor in distributed system such as devices, components, services, or user accounts that collaborate to maintain operations is represented as a node. Each node plays a vital role in maintaining, validating, and, at times, broadcasting transactions

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 February 2025 doi:10.20944/preprints202502.0498.v1

and blocks within the network. The blockchain network comprises a set of such nodes, collectively referred to as *AllNodes*. We define a node, N as

$$N = (basicInfo_N, hierarchyInfo_N, O_N, ACL_N, TM_N)$$

Here, $basicInfo_N$ includes the fundamental information required to uniquely identify and instantiate the node, N within the blockchain network. As shown in Figure 1, *nodeName* is a unique identifier of a node.

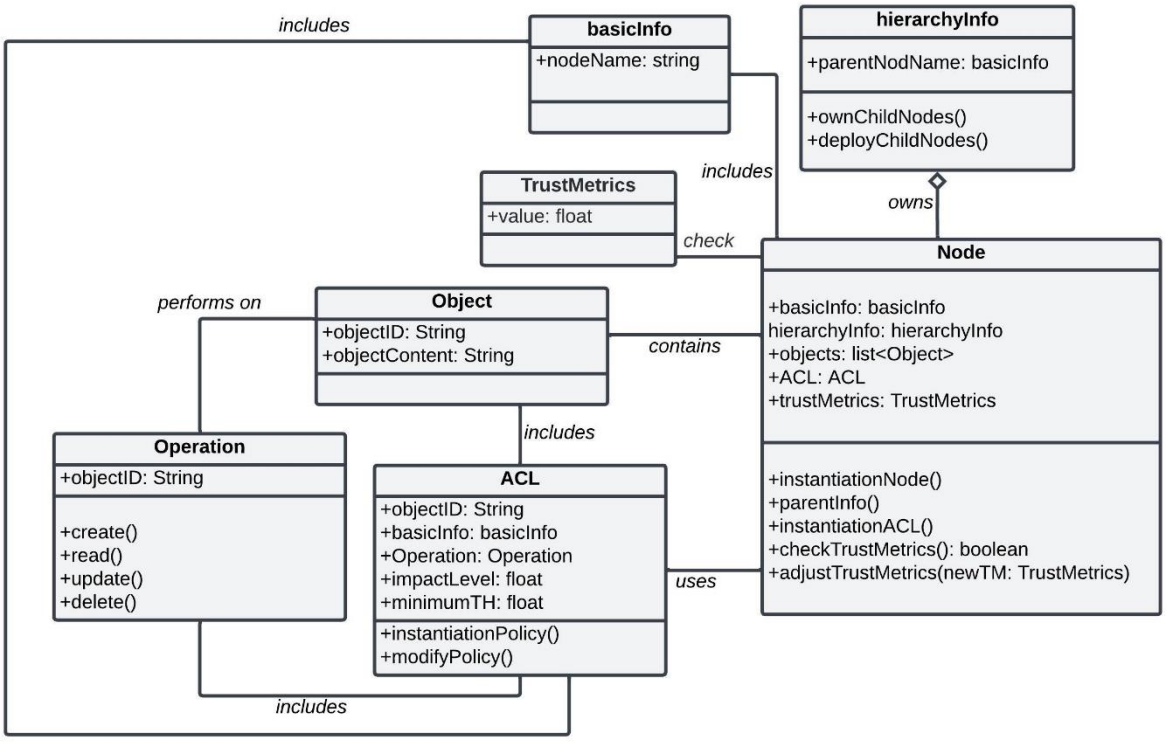

Figure 1. Node Architecture.

$hierarchyInfo_N$ represents the hierarchical relationships within the blockchain network, if applicable. It includes the information of the node's parent, which is another active node in the network. Such hierarchical relationships model interdependencies among system components, reflecting organizational or functional structures.

O_N denotes the set of objects owned by the node.

For each object $o_i \in O_N$, the node maintains an ACL:

$$ACL_N(o_i) = \bigcup_{n \in AllNodes} \{(basicInfo_n, (Opp, I, minTH)\}$$

Each entry in ACL_N defines the permitted operation ($Opp \in OP$) that a node n can perform on the object, o_i. Each operation has an organization defined impact level, I and $minTH$ specifying the minimum required trustworthiness of node n to perform the operation.

OP is the set of all possible operations. In this paper, we are dealing with create (C), read (R), update(U), delete(D) operations. So, $OP = \{C, R, U, D\}$

TM_N represents the trust metric (TM) assigned to the node N. The trust metric reflects the reliability and security posture of the node, dynamically adjusted based on its actions and behavior within the network.

A node encompasses a range of critical functionalities that enable its role within the blockchain infrastructure as shown in Figure 1. It includes the ability to instantiate ACLs for its owned objects, either by creating new policies or modifying existing ones, ensuring that access policies are enforced in accordance with organizational requirements. Additionally, the node adjusts its TM based on computations confirmed by other nodes regarding its actions and behavior within the system, reflecting its evolving trustworthiness. The node is also responsible for performing operations on its objects, such as create, read, update, and delete (CRUD), based on the decisions made by the access

 doi:10.20944/preprints202502.0498.v1

control mechanism. These operations ensure that policies are enforced consistently across the network.

Beyond these access control and policy enforcement functionalities, the node provides essential blockchain infrastructure services. These include block creation, transaction processing, and ledger maintenance, which collectively enable the decentralized, secure, and tamper-resistant operation of the blockchain network.

3.2. Enhanced Smart Contract for DACS with Embedded Security Awareness

The ZT model necessitates three core functionalities: continuous monitoring, risk assessment, and trust evaluation [4]. These components form the backbone of DACS, which rely on a robust policy management mechanism to ensure secure access to objects [6,7]. Continuous Monitoring functionality provides real-time surveillance of every access request. It scrutinizes request patterns, identifies requesters, and analyzes their behaviors. This process ensures that any deviation from expected patterns or anomalous activities can be promptly detected. Continuous monitoring creates a detailed behavioral profile for each requester, offering a granular view of access dynamics over time. The risk assessment procedure evaluates the data collected from continuous monitoring to quantify the risk associated with each access request. By analyzing factors such as the sensitivity of the requested resource, the requester's historical behavior, and the current access context, this step assigns a RF to every access permission. The RF acts as a crucial input for decision-making, enabling proactive responses to potential threats. Trust evaluation complements risk assessment by determining the trustworthiness of the requester. It leverages the insights from continuous monitoring, focusing on the requester's behavioral consistency, compliance with security policies, and alignment with expected access patterns. This process results in a TM that reflects the reliability of the requester under the given access context. To implement DACS, we enhanced smart contract functionality by embedding security awareness, enabling the system to extract insights by aggregating the outcomes of continuous monitoring, risk assessment, and trust evaluation. Using predefined ACLs along with the calculated RF and TM, the embedded security awareness mechanism dynamically evaluates and adjusts access permissions. This adaptive approach ensures that access is granted only when the requester meets the required security and trust thresholds, aligning with ZT model principles to mitigate risks and uphold security in real-time.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 7 February 2025 doi:10.20944/preprints202502.0498.v1

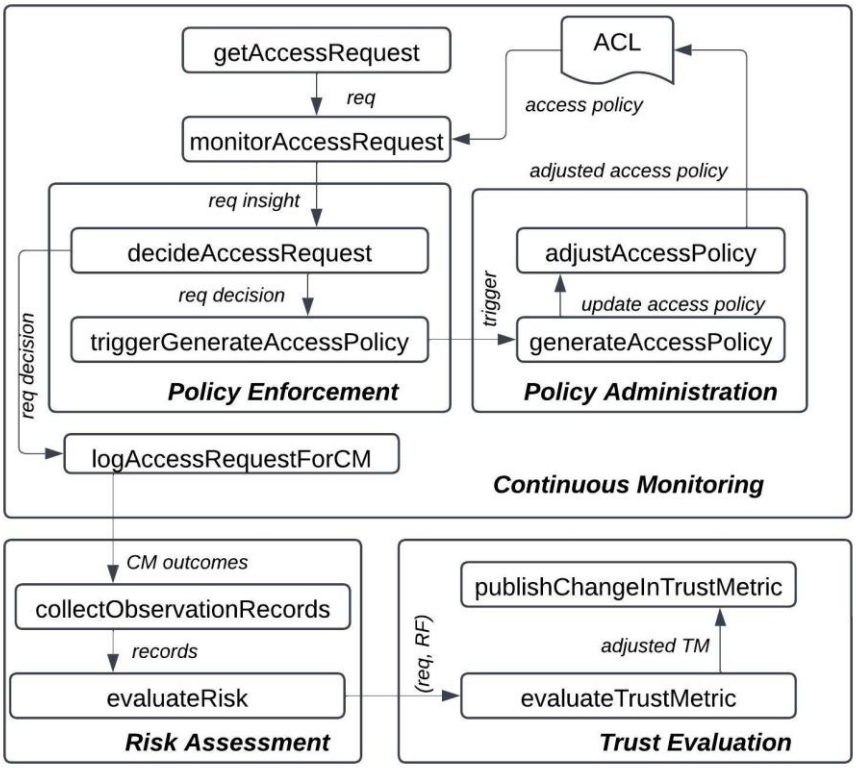

Figure 2. Enhanced smart contract functionalities for embedding security awareness in DACS to implement ZT model.

We enhance the smart contract functionality within our blockchain infrastructure to serve as a decentralized policy management mechanism for DACS, as illustrated in Figure 2. This integration enables secure, automated, and distributed management of access requests. When a blockchain node receives a new access request, the *getAccessRequest* function within the smart contract associated with that node is triggered. This function initiates the monitoring process by passing the access request to the *monitorAccessRequest* function. The monitoring process evaluates the access request against the predefined ACLs and generates actionable insights based on the request's characteristics and context. Let, Req_N represent the set of access requests directed to node N. Each individual access request, req is defined as:

$$req = (\ basicInfo_{\bar{N}}, o_N, Opp, TM_{\bar{N}})$$

It specifies which node $\bar{N} \in AllNodes \setminus \{N\}$ requests access to the object o_N to perform the operation Opp, as well as the TM of the node $\bar{N}$. This $TM_{\bar{N}}$ is a critical decision factor for policy enforcement.

The Policy Enforcement Mechanism includes *decideAccessRequest*, which evaluates and decides on access requests based on insights from *monitorAccessRequest* in real time. Based on this evaluation, there are three possible outcomes:

1. **Access Granted**: If the requesting node, $\bar{N}$ has the necessary access permissions as per the ACL and maintains a TM above the minimum threshold, the operation is allowed, and the node retains its current permissions.
2. **Access Denied – Insufficient Permissions**: If the requesting node, $\bar{N}$ does not have the required access permissions in the ACL, the operation is denied outright.
3. **Access Denied – Low Trust Metric**: If the requesting node, $\bar{N}$ has the required access permissions but its TM falls below the minimum threshold, the operation is denied due to insufficient trustworthiness.

 doi:10.20944/preprints202502.0498.v1

In the third scenario, where access is denied because the node, $\overline{N}$ maintains a TM below the acceptable threshold despite having access permissions. In this case, *decideAccessRequest* triggers a critical follow-up process that invokes Policy Administration mechanism:

Policy Generation: The *generateAccessPolicy* function, part of the Policy Administrator, is activated. This function generates a revised access policy to revoke the access permissions of node , $\overline{N}$ due to its low TM.

Policy Adjustment: The *adjustAccessPolicy* function then updates the ACL to reflect the revoked permissions. This adjustment ensures that the node's access rights are aligned with the current trust evaluation.

ACL Update: Finally, an updated ACL is instantiated for node N that includes the adjusted policy for node, $\overline{N}$, ensuring that the latest trust and access policies are enforced across the system.

This dynamic trust evaluation and policy adjustment ensure that access permissions are not only granted based on predefined rules but also adjusted in response to behavioral and contextual changes, thereby maintaining a robust security posture. Moreover, the continuous monitoring functionality, denoted as *CMFunc*, encompasses a crucial component called *logAccessRequestForCM*. This component is responsible for systematically logging each access request alongside the corresponding access decision. This functionality is integral to ensuring traceability and enabling comprehensive analysis of access control activities within the system. For every access request $req \in Req_N$ the *CMFunc* generates a detailed outcome encapsulating essential information. The outcome can be expressed as:

$$\forall req \in Req_N, CMFunc_N: req \rightarrow (basicInfo_{\overline{N}}, o_N, (Opp, I, minTH), decision, TM_{\overline{N}})$$

The outcome includes $basicInfo_{\overline{N}}, o_N, Opp, TM_{\overline{N}}$ derived from the request, req entry, while I and $minTH$ are obtained from the ACL for the associated Opp on the corresponding o_N. $decision$ represents the final access control outcome, specifying whether the request is approved or denied.

The risk assessment functionality in smart contract enables dynamic risk assessment according to organization defined observation window. The observation window determines the number of recent access requests that must be analyzed to accurately assess and contextualize the current security posture. To support this, the *collectObservationRecords* method retrieves a batch of the most recent access requests from the outcomes of $CMFunc_N$ as specified by the observation window. This approach ensures that the system continuously monitors and evaluates access patterns in real time. A significant number of unauthorized access requests within the observation window serves as an indicator that the node is being targeted, potentially signaling an elevated security risk. In this context, risk in dynamic access control is defined as the combination of two factors: the likelihood of unauthorized access occurring and the impact such access could have on the system's operations or sensitive data. We formulate the probability of a single request to a specific node, N being unauthorized as:

$$p_N\ (unauthorized\ access\ request) = \frac{|(O_N \times OP \times Allnodes)| - |ACL_N|}{|(O_N \times OP \times Allnodes)|}$$

The RF increases as the number of unauthorized access requests rises, indicating that the node is being targeted by malicious actors. The *evaluateRisk* estimates the likelihood, L_N of an unauthorized access request based on the records within the observation window, as derived from the outcomes of $CMFunc_N$. The likelihood is computed using the formula:

$$L_N(unauthorized\ access\ request) = \binom{m}{k} \times {p_N}^k \times (1 - p_N)^{(m-k)}$$

Here, m is total number of access requests within the observation window from $CMFunc_N$ outcomes.

k is number of access requests identified as unauthorized within the observation window from $CMFunc_N$ outcomes. The RF for a specific request, req to access the node, N is defined:

$$RF_N(req) = L_N(unauthorized\ access\ request) \times I \times \begin{cases} 0, if\ decision = granted \\ 1, otherwise \end{cases}$$

The Trust Evaluation functionality within the smart contract is designed to evaluate the TM for a node, $\overline{N}$, that initiates an access request. To enhance security, a penalty enforcement mechanism is

 doi:10.20944/preprints202502.0498.v1

incorporated, which is triggered when an unauthorized access request is detected. The TM of the node, $\overline{N}$ is dynamically adjusted based on the RF at the specific moment of evaluation. The penalty enforcement and trust metric adjustment are formulated as follows:

$$TM_{\overline{N_{adjusted}}} = TM_{\bar{N}} - TM_{\bar{N}} \times RF_N$$

With high RF_N, penalty will be high and dynamically estimated. The adjusted TM will be new TM for the node, $\bar{N}$. *PublishchangeInTrustMetric* allows node, N to instantiate a transaction to publish the changes so that the affected node, $\bar{N}$ of which the TM has adjusted can validate the actions and update its current TM. The smart contract associated with each node repeats the process continuously on every access request to include DACS for implementing ZT model. Our trust evaluation mechanism does not reward a node for behaving as expected. In other words, the TM cannot increase dynamically. The rationale behind this is that persistent attackers may wait for a certain period before attempting unauthorized access to evade detection. In such cases, if a node's TM falls below the threshold, the system administrator can investigate the actor represented by the node and manually reassign the TM based on their findings.

4. Experiment

To evaluate our approach, we designed a blockchain network using the Ethereum blockchain, a decentralized, open-source platform that facilitates the creation and deployment of smart contracts and decentralized applications (DApps). Ethereum's robust infrastructure and support for programmable contracts make it an ideal platform for implementing our solution. We enhanced the functionality of smart contracts using Remix IDE, a powerful web-based development environment tailored for Ethereum blockchain development. Remix IDE is widely recognized for its capabilities in writing, deploying, testing, and debugging smart contracts. It supports Solidity, the most commonly used programming language for Ethereum smart contracts and provides a suite of tools to interact seamlessly with the Ethereum network. One notable advantage of Remix IDE is its interactive interface, which allows developers to test both public and internal functions of smart contracts directly. This feature was instrumental in validating the logic and functionality of our enhanced smart contracts before deployment.

After successfully implementing and deploying the defined node functionalities and enhanced smart contracts, we designed a blockchain network comprising 10 interconnected nodes (S_A to S_J), as outlined in Table 1. This network simulates a decentralized environment, enabling us to rigorously test and analyze the performance and security of our proposed system in a realistic and scalable setup. By leveraging the Ethereum blockchain and its associated tools, we ensured that our experimental setup aligns with industry standards, providing a robust and flexible foundation for evaluating our approach.

For simplicity, we assume that each node is associated with a single object, resulting in a total of 10 objects in the entire application, as outlined in Table 1. CRUD (Create, Read, Update, and Delete) operations can be performed on these objects, and each operation is assigned an impact level based on its potential severity to the application if performed without proper authorization. The impact levels are categorized as follows:

High (H): Impact score of 0.9

Moderate (M): Impact score of 0.5

Low (L): Impact score of 0.2

Table 1. Access Control Lists (ACLs) Matrix: Node-Wise Permissions (Rows) vs. Object-Wise Operations (Columns) for Create (C), Read (R), Update (U), and Delete (D).

	O_A (C=H, R=H, U=H, D=H)	O_B (C=H, R=M, U=H, D=H)	O_C (C=H, R=M, U=H, D=H)	O_D (C=M, R=L, U=L, D=M)	O_E (C=M, R=L, U=L, D=M)	O_F (C=M, R=L, U=L, D=M)	O_G (C=M, R=L, U=L, D=M)	O_H (C=L, R=L, U=M, D=L)	O_I (C=L, R=L, U=L, D=L)	O_J (C=L, R=L, U=M, D=L)

 doi:10.20944/preprints202502.0498.v1

S_A	C, R, U, D	C, R, U, D	C, R, U, D	C, R, U, D	C, R, U, D	C, R, U, D	C, R, U, D	C, R, U, D	C, R, U, D	C, R, U, D
S_B	C	C, R, U, D	C, R	C, R, U, D	C, R, U, D	R	R	R, U	R, U	R, U
S_C	C	C, R	C, R, U, D	R	R	C, R, U, D	C, R, U, D	R	R	R
S_D		C		C, R, U, D	C, R	C, R	C, R	C, R, U, D	C, R, U, D	R
S_E		C		C, R	C, R, U, D	C, R	C, R	R	R	C, R, U, D
S_F			C	C, R	C, R	C, R, U, D	C, R	R	R	R
S_G			C	C, R	C, R	C, R	C, R, U, D	R	R	R
S_H				C				C, R, U, D	C, R	C, R
S_I				C				C, R	C, R, U, D	C, R
S_J					C			C, R	C, R	C, R, U, D

Each object and its associated operations are assigned a minimum threshold value for TM, as detailed in Table 2. The methodology for determining the impact levels and minimum threshold values for TM is beyond the scope of this paper. For our experiment, we assume these values are provided by domain experts, guided by organizational policies and risk assessments.

Table 2. Minimum required trust metric for perform CRUD operations on objects.

	O_A	O_B	O_C	O_D	O_E	O_F	O_G	O_H	O_I	O_J
Create (C)	0.95	0.8	0.8	0.65	0.65	0.65	0.65	0.55	0.55	0.55
Read (R)	0.95	0.75	0.75	0.6	0.6	0.6	0.6	0.55	0.55	0.55
Update (U)	0.95	0.8	0.8	0.6	0.6	0.6	0.6	0.6	0.6	0.6
Delete (D)	0.95	0.8	0.8	0.65	0.65	0.65	0.65	0.55	0.55	0.55

Each node's ACL contains entries only for its own objects. For instance, node S_F owns the object, O_F. So, the ACL for node S_F for object, O_F would be:

$$ACL_F(o_F) = \{(basicInfo_{S_A}, (C, M, 0.65), (basicInfo_{S_A}, (R, L, 0.6), (basicInfo_{S_A}, (U, L, 0.6), (basicInfo_{S_A}, (D, M, 0.65), (basicInfo_{S_B}, (R, L, 0.6), (basicInfo_{S_C}, (C, M, 0.65), (basicInfo_{S_C}, (R, L, 0.6), (basicInfo_{S_C}, (U, L, 0.6), (basicInfo_{S_C}, (D, M, 0.65), (basicInfo_{S_D}, (C, M, 0.65), (basicInfo_{S_D}, (R, L, 0.6), (basicInfo_{S_E}, (C, M, 0.65), (basicInfo_{S_E}, (R, L, 0.6), (basicInfo_{S_F}, (C, M, 0.65), (basicInfo_{S_F}, (R, L, 0.6), (basicInfo_{S_F}, (U, L, 0.6), (basicInfo_{S_F}, (D, M, 0.65), (basicInfo_{S_G}, (C, M, 0.65), (basicInfo_{S_G}, (R, L, \}$$

After designing the blockchain network and successfully instantiating the nodes, assigning their ACLs, and specifying the corresponding TM and minimum TM thresholds for each operation, we conducted experiments on various scenarios. In these scenarios, a node receives access requests—both authorized and unauthorized—for its objects. The node dynamically determines the RF, evaluates the access request based on the ACL policy, RF, and the current TM of the requesting node, and decides whether to grant or deny access.

 doi:10.20944/preprints202502.0498.v1

Below, we describe three scenarios in detail:

Scenario 1: Access Request Granted

Our first scenario is that node, S_F receives an access request from node, S_B to perform a read (R) operation to object O_F as below:

$$req_{sc1} = (\ basicInfo_{S_B}, o_F, R, 1)$$

The request includes the current TM of node S_B, denoted as, TM_B as 100%. Upon receiving the access request, the smart contract associated with node, S_F triggers to evaluate the request. The smart contract checks the access policy defined ACL of node, S_F for object, O_F, denoted as $ACL_F(O_F)$.

According to $ACL_F(O_F)$, the read operation is permitted (as shown in Table 1), and the minimum required TM threshold is 60% (as shown in Table 2). Since the current TM_B is 100%, which exceeds the required threshold, the access request is "granted." The decision is logged for continuous monitoring and appended to the outcome of the Continuous Monitoring Function ($CMFunc_F$) as follows:

$$CMFunc_F: req_{sc1} \rightarrow (basicInfo_{S_B}, o_F, (R, 0.2, 0.6), "granted", 1)$$

Since, the request is "granted", no risk assessment and trust evaluation have been performed.

Scenario 2: Access Request Denied Due to No Permission

In the second scenario, node S_F receives an access request from node S_G to perform an update (U) operation on object O_F, with TM_G currently at 100%. The access request is defined as:

$req_{sc2} = (\ basicInfo_{S_G}, o_B, U, 1)$

The smart contract evaluates the access request based on $ACL_F(O_F)$ and TM_G. The decision is to deny the request, as node, S_G lacks update operation permission for object, O_F (as referenced in Table 1). The outcome generated by $CMFunc_B$ includes the decision and relevant information as follows:

$$CMFunc_B: req_{sc2} \rightarrow (basicInfo_{S_G}, o_F, (U, 0.2, 0.6), "denied", 1)$$

Risk Assessment and Trust Metric Adjustment

Risk Assessment mechanism is triggered to evaluate RF_{S_F} (the Risk Factor for node, S_F) and adjust TM_G to penalize node, S_G. Any unauthorized access request is considered a potential security compromise attempt.

The risk assessment uses an observation window containing a specified number of recent records to determine the frequency of denied access requests. For demonstration, we consider variations in the observation window size and the resulting RF_{S_F} and adjusted TM_G. The Risk Factor and adjusted TM vary depending on the observation window size and the number of unauthorized access requests within the window. These variations, along with their calculated to RF and TM adjustment, are summarized in Table 3.

The probability of a single request to node, S_F being unauthorized is calculated as below:

$$p_{S_F}(unauthorized\ access\ request) = \frac{(1 \times 4 \times 10) - 19}{(1 \times 4 \times 10)} = 0.525$$

In our experimenting observation window from $CMFunc_B$ outcomes, 25 recent records are considered, with 3 recorded as unauthorized access requests. The likelihood of an unauthorized access request (L_B) is computed as:

$$L_{S_F}(unauthorized\ access\ request) = \binom{25}{3} \times 0.525^3 \times (1 - 0.525)^{22} = 2.57 \times 10^{-5}$$

Risk Factor and Adjusted Trust Metric

The RF_{S_F} calculated after the denied access request, req_{sc2} is:

$$RF_{S_F}(req_{sc2}) = 2.57 \times 10^{-5} \times 0.2 \times 1 = 0.00000514$$

And the adjusted. TM_G would be:

$$TM_{\overline{G_{adjusted}}} = 1 - 1 \times 0.00000514 = 0.99999486$$

The *PublishchangeInTrustMetric* function allows node, S_F to initiate a transaction to notify node, S_G of its adjusted TM_G, which will be validated by node Sg and established in the network.

 doi:10.20944/preprints202502.0498.v1

Table 3. Dynamic RF and adjusted TM estimation with variation in records across different observation windows (considering that we perform the calculation when node S_F receives an access request from node S_G to perform an update (U) operation on object O_F, and the request has been "denied").

Observation window size	Number of unauthorized access requests	RF	Current TM	Adjusted TM
25	3	0.00000514	1	0.99999486
50	7	0.00000236	1	0.99999764
25	3	0.00000514	0.7	0.699996402
50	7	0.00000236	0.7	0.699998348

Scenario 3: Access Request Denied Due to Insufficient Trust Metric

In the third scenario, node, S_F receives an access request from node, S_C to perform a read (R) operation on object O_F. However, TM_C (current Trust Metric of node, S_C) is only 50%, as specified below:

$req_{sc3} = (\ basicInfo_{S_C}, o_F, R, 0.5)$

The access request is denied, even though node, S_C has read operation permission in $ACL_F(O_F)$, (see Table 1) This denial occurs because the minimum required trust metric to perform read operation on O_F is 60% (as referenced in Table 2), which node , S_C fails to meet.

Triggering Policy Adjustment

This decision triggers the *generateAccessPolicy* mechanism to create a new access policy. The updated policy revokes previously granted permission to node, S_C to perform the read operation on object, O_F, reflecting its reduced trustworthiness.

The *adjustAccessPolicy* function then updates $ACL_F(O_F)$ to reflect the changes, creating a new instance of the Access Control List.

5. Performance Analysis

To evaluate the scalability of the proposed DACS framework, we measured key performance metrics, such as transaction validation time for processing access requests, as the number of participating nodes and access requests increased. Additionally, to assess the operational overhead introduced by the dynamic RF and TM adjustments in our approach, we compared the average transaction validation time with a basic block-chain-based access control management approach. The latter simply allows nodes to decide on access requests based on a predefined ACL for the node's objects. The comparison results are presented in Table 4. We conducted the experiments on a personal computer running Linux (csx2 5.15.0-130-generic #140-Ubuntu SMP x86_64 x86_64 x86_64 GNU/Linux). The results indicate that while the additional processes of risk evaluation and TM adjustment at individual nodes—validated by the PoS consensus mechanism—introduce some operational overhead, it remains within a tolerable range. In future work, we will explore optimizations to enhance the performance of our approach.

Table 4. Performance comparison of our approach with a basic blockchain-based access control management approach by varying the number of nodes in the network.

Number of Nodes	Average Transaction Validation Time (Our Approach) in Seconds	Average Transaction Validation Time (Basic Approach) in Seconds
15	0.08523	0.002839
50	0.261504	0.002834
100	0.468003	0.002819
500	2.31576	0.002814

 doi:10.20944/preprints202502.0498.v1

1000	4.602902	0.002920

6. Conclusions

The rapid evolution of distributed systems has introduced complex security challenges, necessitating the implementation of the Zero Trust (ZT) model to address security threats effectively. Dynamic Access Control Scheme (DACS) are critical in realizing ZT principles, offering adaptive and context-aware security measures. Embedding security awareness into DACS enhances the ability to dynamically adjust access policies based on real-time contextual analysis. This paper presents an innovative DACS framework with embedded security awareness and leverages blockchain technology to eliminate the reliance on centralized trusted nodes, thereby improving security and scalability. By enhancing smart contract functionalities, the framework supports continuous risk assessment, real-time policy adjustments, and penalty enforcement that respond to evolving threats. Metrics such as Risk Factor (RF) and Trust Metric (TM) ensure granular and dynamic access control, aligning with organizational security objectives and the core principles of the ZT model. The proposed framework adopts a Proof of Stake (PoS) consensus mechanism to validate transactions related to access control policies and TM adjustments. However, the study acknowledges two limitations: (1) the absence of a procedural approach to determine the minimum required TM for operations based on organizational impact levels, and (2) the lack of a communication protocol to coordinate TM adjustments across multiple nodes. Future work will focus on addressing these gaps. An evaluation of our approach on the Ethereum blockchain demonstrates the framework's effectiveness in delivering resilient, adaptable, and decentralized access control. This solution not only strengthens the security posture of distributed systems but also lays the foundation for future innovations in secure, scalable, and context-aware access management in dynamic and open environments.

Author Contributions: "Conceptualization", Avoy Mohajan and Sharmin Jahan, "Methodology", Avoy Mohajan and Sharmin Jahan, "Validation" , Avoy Mohajan, "Writing-original draft", Avoy Mohajan and Sharmin Jahan, "Investigation", Sharmin Jahan, "Writing-Reviewing and editing", Sharmin Jahan, "Supervision", Sharmin Jahan.

Funding: This research received no external funding.

References

1. Van Steen, M.; Tanenbaum, A. S. *Distributed systems*, 3rd ed.; Leiden, The Netherlands: Maarten van Steen, 2017.
2. Patil, A. et al. Design and implementation of a consensus algorithm to build zero trust model. 2020 IEEE 17th India Council International Conference (INDICON). IEEE, 2020, doi: 10.1109/INDICON49873.2020.9342207.
3. Sengupta, B.; Anantharaman, L. Distritrust: Distributed and low-latency access validation in zero-trust architecture. Journal of Information Security and Applications, 2021, doi: https://doi.org/10.1016/j.jisa.2021.103023.
4. Stafford, V. Zero trust architecture. NIST special publication 800-207, 2020, doi: https://doi.org/10.6028/NIST.SP.800-207.
5. Gong, Q. et al. SDACS: Blockchain-Based Secure and Dynamic Access Control Scheme for Internet of Things. Sensors 24.7: 2267, 2024, doi: https://doi.org/10.3390/s24072267.
6. Alboqmi, R. Jahan, S. Gamble, R. F. A Risk Adaptive Access Control Model for the Service Mesh Architecture, 2024 IEEE 3rd International Conference on Computing and Machine Intelligence (ICMI), 2024, pp. 1-6, doi: 10.1109/ICMI60790.2024.10585800.
7. Wang, P. et al. Dynamic access control and trust management for blockchain-empowered IoT. IEEE Internet of Things Journal 9.15, 2021, pp. 12997-13009, doi: 10.1109/JIOT.2021.3125091.
8. Hwang, D. Y.; Jung, Y. C.; Ki-Hyung K. Dynamic access control scheme for iot devices using blockchain. 2018 international conference on information and communication technology convergence (ICTC). IEEE, 2018, doi: 10.1109/ICTC.2018.8539659.
9. Alevizos, L.; Vinh T. T.; Max H. E. Augmenting zero trust architecture to endpoints using blockchain: A state-of-the-art review. Security and privacy 5.1: e191, 2021, doi: https://doi.org/10.1002/spy2.191.

10. Alkhresheh, A.; Khalid E.; Hossam S. H. DACIoT: Dynamic access control framework for IoT deployments. IEEE Internet of Things Journal 7.12, 2020, pp. 11401-11419, doi: 10.1109/JIOT.2020.3002709.
11. Dutt, N. et al. Self-awareness for autonomous systems. Proceedings of the IEEE 108.7, 2020, pp. 971-975, doi: 10.1109/JPROC.2020.2990784.
12. Petrovska, A. Self-Awareness as a Prerequisite for Self-Adaptivity in Computing Systems. 2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C). IEEE, 2021, doi: 10.1109/ACSOS-C52956.2021.00039.
13. Jahan, S.; Gamble, R. F. Applying Security-Awareness to Service-Based Systems. 2021 IEEE International Conference on Autonomic Computing and Self-Organizing Systems Companion (ACSOS-C). IEEE, 2021, doi: 10.1109/ACSOS-C52956.2021.00041.
14. Jahan, S. *An adaptation assessment framework for runtime security assurance case evolution*, Diss. The University of Tulsa, 2021.
15. Vanickis, R. et al. Access control policy enforcement for zero-trust-networking. 2018 29th Irish Signals and Systems Conference (ISSC). IEEE, 2018, doi: 10.1109/ISSC.2018.8585365.
16. Gai, K. et al. A blockchain-based access control scheme for zero trust cross-organizational data sharing. ACM Transactions on Internet Technology 23.3, 2023, pp. 1-25, doi: https://doi.org/10.1145/3511899.
17. Whyte, S. T.; Omoyiola, B. O.; Okoni, B. Use of Blockchain Technology in Data Integrity Assurance. SSRN, 2022
18. Zhang, Y. et al. Smart Contract-Based Access Control for the Internet of Things, in IEEE Internet of Things Journal, vol. 6, no. 2, 2019, pp. 1594-1605, doi: 10.1109/JIOT.2018.2847705.
19. Rahman, M.; Barbara, G.; Fabrizio, B. Blockchain-based access control management for decentralized online social networks. Journal of Parallel and Distributed Computing 144, 2020, pp. 41-54, doi: https://doi.org/10.1016/j.jpdc.2020.05.011.
20. Radack, S. M. Federal information processing standard (FIPS) 199, standards for security. 2004.
21. Wang, J. et al. Trust and attribute-based dynamic access control model for Internet of Things. 2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC). IEEE, 2017, doi: 10.1109/CyberC.2017.47.
22. Ahmed, A. et al. BACAD: AI-based framework for detecting vertical broken access control attacks. Egyptian Informatics Journal 28: 100571, 2024, doi: https://doi.org/10.1016/j.eij.2024.100571.
23. Nguyen, C. T. et al. Proof-of-stake consensus mechanisms for future blockchain networks: fundamentals, applications and opportunities. IEEE access 7, 2019, pp. 85727-85745, doi: 10.1109/ACCESS.2019.2925010.
24. Peepliwal, A. K et al. A prototype model of zero trust architecture blockchain with EigenTrust-based practical Byzantine fault tolerance protocol to manage decentralized clinical trials. Blockchain: Research and Applications 5.4: 100232, 2024, doi: https://doi.org/10.1016/j.bcra.2024.100232.
25. Kulkarni, A.; Hazari, N. A.; Niamat, M. A Zero Trust-based framework employed by Blockchain Technology and Ring Oscillator Physical Unclonable Functions for security of Field Programmable Gate Array Supply Chain. IEEE Access, 2024, doi: 10.1109/ACCESS.2024.3418572.
26. Elmadani, S.; Hariri, S.; Shao, S. Blockchain based methodology for zero trust modeling and quantification for 5g networks. 2022 IEEE/ACS 19th International Conference on Computer Systems and Applications (AICCSA). IEEE, 2022, doi: 10.1109/AICCSA56895.2022.10017914.
27. Feng, Y. et al. "Blockchain enabled zero trust based authentication scheme for railway communication networks." Journal of Cloud Computing 12.: 62, 2023.
28. Jin, Q.; Liming, W. Zero-trust based distributed collaborative dynamic access control scheme with deep multi-agent reinforcement learning. EAI Endorsed Transactions on Security and Safety 8.27, 2020, doi: http://dx.doi.org/10.4108/eai.25-6-2021.170246.
29. Ali, F.S. et al. Dynamic acl policy implementation in software defined networks. 2022 International Conference on IT and Industrial Technologies (ICIT). IEEE, 2022, doi: 10.1109/ICIT56493.2022.9989241.
30. Jung, C. et al. A scalable and dynamic acl system for in-network defense. Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. 2022, doi: https://doi.org/10.1145/3548606.3560606.
31. You, H. et al. Dynamic access control method for SDP-based network environments. EURASIP Journal on Wireless Communications and Networking 2023.1: 94, 2023.
32. Sun, L. et al. BPDAC: A Blockchain Based and Provenance Enabled Dynamic Access Control Scheme. IEEE Access, 2023, doi: 10.1109/ACCESS.2023.3340887.
33. Nakamura, Y. et al. Exploiting smart contracts for capability-based access control in the internet of things. Sensors 20.6: 1793, 2020, doi: https://doi.org/10.3390/s20061793.
34. Rouhani, S.; Deters, R. Blockchain based access control systems: State of the art and challenges. IEEE/WIC/ACM International Conference on Web Intelligence. 2019, doi: https://doi.org/10.1145/3350546.335256.
35. Punia, A. et al. A systematic review on blockchain-based access control systems in cloud environment. Journal of Cloud Computing 13.1: 146, 2024.

 doi:10.20944/preprints202502.0498.v1