ClaimGuard: A Blockchain-Backed Access Control Gateway for Privacy-Preservation in Auto-Insurance Claims

Anthony Uchenna Eneh [1], Love Allen Chijioke Ahakonye [2], Jae Min Lee [1], Dong-Seong Kim [1] *
[1] IT-Convergence Engineering, *Kumoh National Institute of Technology*, Gumi, South Korea
* NSLab Co. Ltd., Gumi, South Korea, *Kumoh National Institute of Technology*, Gumi, South Korea
[2] ICT Convergence Research Center, *Kumoh National of Technology*, Gumi, South Korea
(anthony, loveahakonye, dskim, ljmpaul)@kumoh.ac.kr

***Abstract*—Modern auto-insurance workflows require sharing heterogeneous digital evidence across multiple organizations. Yet, current cloud-based role-based access control mechanisms remain coarse-grained and poorly suited for expressing time, purpose, and case-specific constraints. This study presents *ClaimGuard*, which addresses these limitations by placing a blockchain-backed attribute-based access control gateway in front of existing evidence stores, enforcing fine-grained on-chain policies, and issuing short-lived capability tokens for authorized access. Implemented as a REST gateway with PureChain smart contracts, ClaimGuard is evaluated using realistic workloads involving up to 200 subjects and 1000 evidence resources. Experiments on a local Ethereum network shows sub $\sim$70ms tail latency, throughput exceeding $\sim$1000 requests/s, rapid policy updates, and zero false accepts, demonstrating the practicality of decentralized, auditable access control for privacy-preserving claims evidence sharing.**

***Index Terms*—ABAC, access control, Blockchain, capability tokens, Ethereum, insurance, privacy, RBAC**

I. Introduction

The increasing reliance on heterogeneous digital evidence, ranging from high-resolution accident videos to telematics traces and medical documentation, has transformed modern auto-insurance claim workflows into data-intensive, multi-party environments [1], [2]. Yet, prevailing deployments continue to rely on coarse-grained, cloud-resident role-based access control (RBAC) configurations that grant broad privileges, offer limited expressiveness for workflow-specific constraints, and disperse auditability across opaque infrastructure layers [3]. These limitations create structural risks, such as over-privileged roles, weak accountability across organizational boundaries, and persistent exposure to insider access at the storage layer [3].

Blockchain technologies offer a decentralized and tamper-evident execution environment that serves as a robust substrate for access control in multi-party evidence workflows [4]. By encoding fine-grained, attribute-based policies in smart contracts, blockchain systems overcome the expressiveness and rigidity of traditional RBAC, enabling case-specific and temporally bounded rules [5]. Their immutable ledgers consolidate auditability across organizations, and verifiable on-chain decisions reduce reliance on trusted cloud operators, thereby limiting insider privileges and strengthening system-wide accountability [6].

The progressive dependence of Auto-insurance claims on heterogeneous digital evidence, such as dashcam footage, telematics, Controller Area Network (CAN) bus logs, repair invoices, and medical reports, that must be exchanged among insurers, garages, emergency responders, police, and courts, all operating under distinct legal and data-minimization requirements. In practice, this exchange is typically governed by coarse-grained RBAC policies embedded in cloud object stores or databases [7]–[9], where roles such as *INSURER*, *GARAGE*, and *POLICE* are provisioned with broad, long-lived privileges [10]–[12]. This model introduces well-known limitations: it cannot express case-specific constraints, it fragments auditability across heterogeneous infrastructure, and it leaves cloud-layer administrators inherently over-trusted, enabling potential misuse or unauthorized access to evidence.

To address these issues, we design *ClaimGuard*, a blockchain-backed attribute-based access control (ABAC) gateway for claims evidence. Instead of granting actors direct access to the evidence store, all requests pass through a gateway that (i) evaluates ABAC policies on an Ethereum-compatible chain, and (ii) issues short-lived capability tokens that authorize reads from the evidence store. The smart contracts encode subject attributes, resource metadata, workflow context, and fine-grained policy rules; every decision is logged to the ledger, and the object store only serves content when presented with a valid token. This study focuses on the system design and the evaluation of the blockchain-backed ABAC path itself. A broader comparison with legacy RBAC deployments and experiments on real-world claims datasets is left to follow-up journal work.

Finally, risk-adaptive authorization and anomaly detection are out of scope for this version: while ClaimGuard's on-chain decision log and gateway telemetry are amenable to ML-based risk scoring and abnormal-access detection, we defer their design and evaluation to extended work.

This paper makes the following contributions.

- We formulate the access-control and auditability requirements of multi-party auto-insurance claims sharing, and

derive a set of design requirements for a blockchain-backed gateway.
- We design *ClaimGuard*, an architecture that combines on-chain ABAC evaluation with off-chain capability tokens and a REST gateway, while remaining compatible with existing cloud object stores.
- We implement ClaimGuard on a local Ethereum network using Solidity contracts and a Node.js/TypeScript gateway, and develop an experimental harness that exercises 200 subjects and 1000 evidence resources under configurable concurrent workloads.
- We evaluate latency, throughput, and policy-update overhead, and discuss the security and correctness properties of the design.

II. Related Work

A. Blockchain for Insurance and Data Sharing

A growing body of work applies distributed ledgers to insurance processes, including parametric insurance, peer-to-peer risk pools, and claims automation [13]–[17]. Several proposals store claims metadata or audit logs on-chain while keeping raw evidence off-chain to preserve privacy and control storage costs. Similarly, prior work on blockchain-based data marketplaces and healthcare records uses smart contracts to encode access policies and log disclosures for later audit [18], [19]. ClaimGuard adopts this pattern but targets the concrete requirements of auto-insurance evidence sharing, including integration with existing cloud object stores and support for workflow-centric policies.

B. Attribute-Based Access Control on Blockchains

Attribute-based access control (ABAC) provides a flexible way to express policies over subject, resource, and environmental attributes [20]. Recent work has explored embedding ABAC into smart contracts to decentralize decision-making [21], [22]. Typical designs push attribute assertions and policy rules on-chain and evaluate them in a policy decision contract. However, many of these systems either assume that resources themselves are on-chain or omit the practical gateway design that mediates access to legacy storage systems. ClaimGuard instead implements a pragmatic ABAC *gateway* that uses on-chain decisions to drive token issuance for an external object store.

C. Capability Tokens and Secure Gateways

Capability-based access control associates rights with unforgeable tokens rather than identities [23]. In web settings, signed bearer tokens (e.g., JWTs) are widely used to represent delegated rights [24]. Our design follows this tradition: once an on-chain ABAC policy authorizes access, the ClaimGuard gateway issues a short-lived capability token that is required to fetch the underlying evidence object. Similar gateway patterns have been studied in cloud storage security and multi-tenant architectures [25]; our contribution is to bind these capabilities to a tamper-evident on-chain ABAC decision path.

III. System Design and Methodology

A. Threat Model and Requirements

We consider an insurer that stores digital claims evidence in a cloud object store (e.g., S3, MinIO, or IPFS-backed storage). The insurer interacts with external organizations such as garages, police, and courts, who must be granted controlled access to evidence for specific claims. We assume that the cloud provider and some internal operator accounts may be malicious or compromised, and that external actors may attempt replay, token forgery, or role-spoofing attacks.

From this setting, we derive the following requirements:

- **R1 – Fine-grained policies:** Decisions must depend on claim state, subject role, purpose, and resource sensitivity.
- **R2 – Strong auditability:** Every access decision and policy update must be recorded in a tamper-evident log.
- **R3 – Cloud-interface insider resistance:** The evidence store should not serve objects to any accessor, including cloud administrators, unless the request carries a valid, short-lived capability token generated after an on-chain ABAC evaluation; preventing raw disk inspection by the storage operator is out of scope.
- **R4 – Practical performance:** End-to-end latency should remain within a few hundred milliseconds, and throughput should support bursty workloads.

Evaluation assumptions and external validity. Our experiments run on a single-host local Ethereum-compatible network without injected network faults. Since end-to-end latency and cost are sensitive to consensus configuration, block time, and node placement; in consortium or public deployments, additional variability is expected. Quantifying these effects is left to extended evaluation.

B. Architecture Overview

Fig. 1 sketches the ClaimGuard architecture. The core components are:

- **ABAC contracts on PureChain:** Smart contracts maintain subject attributes (role, organization, jurisdiction), resource metadata (type, sensitivity, case identifier), and policy rules.
- **ClaimGuard gateway:** A stateless REST API fronting the evidence store. It handles access requests, calls the on-chain policy decision point, issues tokens for successful decisions, and logs results.
- **Evidence store adapter:** A thin layer that validates tokens and proxies allowed GET operations to the underlying object store.
- **Policy admin API:** An authenticated API for insurers to add, update, and revoke policy rules.

Subjects interact only with the gateway. When a subject requests access to a resource (e.g., a video for case #81), the gateway constructs an access query that includes the subject address, the resource identifier, and the requested action. It then calls a `checkAccess` function in the on-chain Access Policy Manager contract. If the decision is positive, the gateway issues a capability token bound to the subject, resource,

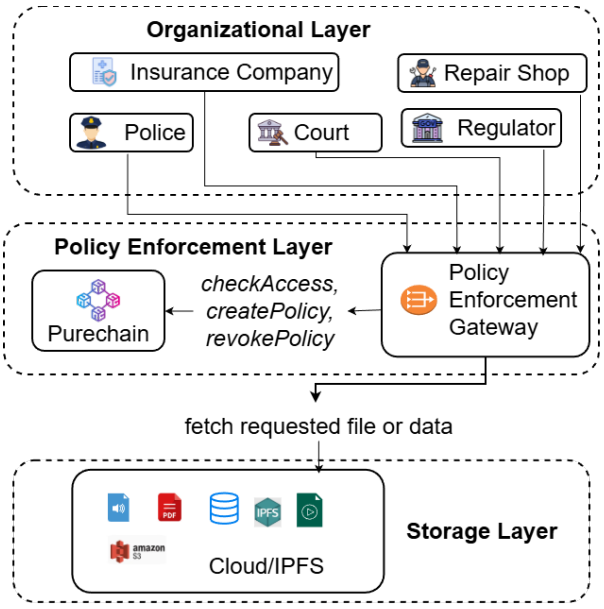

Fig. 1. ClaimGuard architecture. All evidence access passes through a REST gateway that consults on-chain ABAC contracts before issuing short-lived capability tokens for the object store.

action, and an expiry time; the resource server validates this token before returning the actual object.

C. On-Chain ABAC Model

The on-chain ABAC state is split across three contracts:

- **SubjectRegistry** stores public keys and attributes such as `role` (`INSURER`, `GARAGE`, `POLICE`, `COURT`), `orgId`, and jurisdiction.
- **EvidenceRegistry** maintains metadata for each resource: case identifier, type (`VIDEO`, `TELEMATICS`, `MEDICAL_REPORT`, etc.), sensitivity level, and a content hash and URI.
- **AccessPolicyManager** encodes policy rules as tuples over subject attributes, resource attributes, workflow stage, and action (`READ`, `APPEND`, `DELETE`). Each rule can be toggled or updated on-chain via admin transactions.

The `checkAccess` function takes a subject address, resource identifier, and action. It looks up the subject and resource attributes, derives the current workflow stage from policy metadata, and evaluates the policy rules using a fixed-order rule engine. The result is returned as a Boolean together with an event that logs the decision to the chain.

D. Implementation and Experimental Setup

We implement the contracts in Solidity and deploy them to a local Ethereum network using Hardhat. The ClaimGuard gateway is built in TypeScript on top of Express and the Viem client library. The gateway exposes two key endpoints:

- `POST /access`: evaluates an access request via `checkAccess`, issues a capability token on success, and records the decision.
- `POST /policy`: adds or updates a policy rule on-chain and returns the transaction hash and gas usage.

To drive experiments, we implement Python load generators:

- `access_test.py` generates access requests with configurable concurrency, sampling subjects and resources from pre-seeded JSON files and actions from a small set (here we focus on `READ`). It records per-request latency and HTTP status to CSV.
- `policy_update_test.py` repeatedly calls the policy admin endpoint to add or modify rules, logging block confirmation latency, gas usage, and block numbers.

We seeded the system with 200 subjects and 1000 evidence resources, spanning roles, resource types, and sensitivity levels. Policy rules are configured such that a mix of requests is allowed or denied based on role, case identifier, and sensitivity. Experiments are run with 1,000 access requests per setting and concurrency levels of 10, 50, 100, and 200 clients. All components run on a single development machine, which approximates the latency and throughput of a tightly coupled microservice deployment; our focus is on the relative overhead of on-chain ABAC and policy updates.

IV. Results and Evaluation

A. Access Latency

We first examine end-to-end access latency from the subject's perspective, measured at the gateway from the time the `/access` request is received to the time an HTTP response is returned. Table I summarizes the median (P50), 90th percentile (P90), and 99th percentile (P99) latencies for `READ` requests across the tested concurrency levels.

TABLE I
End-to-End Access Latency for READ Requests

Concurrency	P50	P90	P99
10	30.807 ms	42.834 ms	45.192 ms
50	31.713 ms	40.959 ms	63.176 ms
100	30.748 ms	33.507 ms	39.965 ms
200	31.201 ms	34.028 ms	40.993 ms

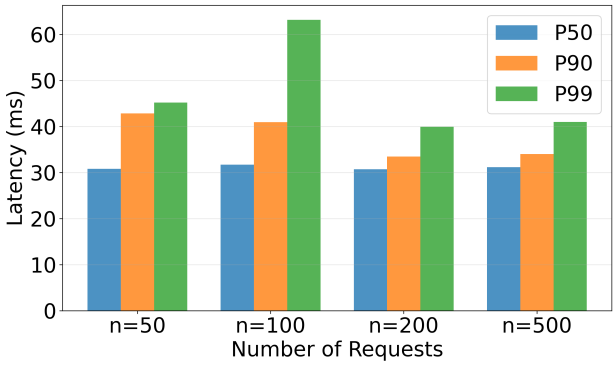

Fig. 2. Latency Chart for Access Requests

Across all workloads, median latency stays well below 65 ms, and even at the 99th percentile, it remains within

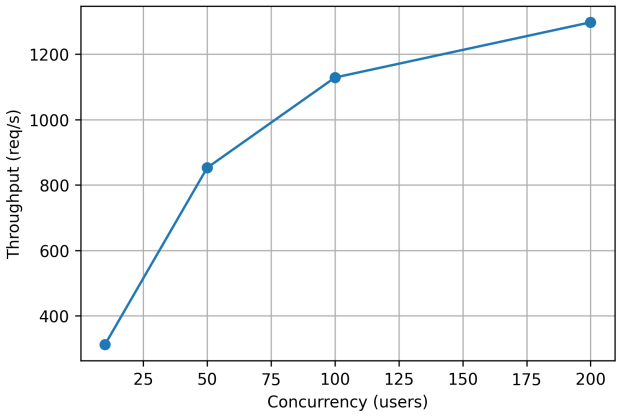

Fig. 3. PureChain ABAC throughput for `READ` requests as concurrency increases.

a few tens of milliseconds. The dominant contributors are the on-chain `checkAccess` call and the gateway-side token generation. Even with occasional outliers as seen in Fig. 2 for 100 concurrent requests, the system remains quite scalable. These results indicate that decentralized policy evaluation on an Ethereum-compatible chain can satisfy the responsiveness requirements of interactive claims workflows.

External validity. Typical insurer traffic involves at most tens of evidence reads per second per organization. In permissioned deployments, WAN placement and block times may add tens of milliseconds to end-to-end latency. The qualitative result, namely that on-chain ABAC combined with short-lived capabilities remains interactive, should hold under such conditions.

B. Throughput

Fig. 3 plots gateway throughput (successful `/access` responses per second) as concurrency increases. Throughput grows with concurrency up to about 100–200 clients, reaching over $\sim$1,000 requests/s on our testbed. Beyond 100 clients, the curve begins to flatten, reflecting the limits of the single-node gateway and local blockchain node rather than any fundamental protocol bottleneck. Even so, the observed throughput is sufficient for typical insurer workloads, which involve at most tens of evidence reads per second per organization.

C. Policy Update Overhead

Next, we evaluate the on-chain cost of modifying ABAC rules. In this experiment, the load generator submits 10 consecutive policy updates via the `/policy` endpoint. It measures the time from submission to block confirmation, along with gas usage and the number of blocks. Table II reports the average metrics across these updates. In practice, this means that an insurer can roll out new workflow rules or revoke subject permissions and see the effect take place within a few hundred milliseconds, while the on-chain gas cost remains modest for an administrative operation.

TABLE II
AVERAGE POLICY UPDATE COSTS ON PURECHAIN

Metric	Range	Average
Latency	13.72–34.2 ms	16.584 ms
Gas used	114k–134k	128k

D. Authorization Correctness

While our current experiments do not yet compare ClaimGuard against a deployed RBAC baseline, we can still assess the internal consistency of its authorization decisions. The correctness experiment tags each synthetic request with an "intended" authorization outcome derived from the same policy rules, and compares it with the on-chain decision. We compute false-accept and false-reject rates by aggregating outcomes over 1,000 requests per setting across four concurrency levels; across all workloads, we observe zero false accepts (unauthorized requests incorrectly allowed) and a small number of false rejects attributable to time-window edge cases in the synthetic generator. The absence of false accepts is a direct consequence of driving all decisions through the deterministic on-chain rule engine.

E. Discussion

The experimentation evaluation shows that ClaimGuard satisfies the requirements outlined in Section III. On-chain ABAC evaluation and capability-token issuance add modest latency overhead while preserving high throughput. Policy updates can be completed quickly enough to support emergency revocation scenarios, and the fully mediated token path eliminates classes of cloud insider attacks and ad hoc access granting. The present experiments rely on structurally realistic but synthetic data and focus solely on the blockchain-backed ABAC path; a complete comparison against existing RBAC deployments and experiments with real claims datasets will be presented in a subsequent journal article.

Security considerations beyond authorization. The ClaimGuard gateway is a potential DoS target; practical mitigations include per-subject/organization rate limiting (e.g., token buckets), autoscaling behind a WAF/CDN, and short-circuiting denials locally to avoid unnecessary on-chain calls. Smart-contract upgradeability also introduces risk; we recommend governance-guarded proxies with timelocks, explicit pause/rollback procedures, and disabling capability issuance during pauses to prevent inconsistent states.

V. CONCLUSION AND FUTURE WORK

This paper presents ClaimGuard, a blockchain-backed ABAC gateway for privacy-preserving auto-insurance claim evidence sharing. By moving policy evaluation to a PureChain smart-contract suite and issuing short-lived capability tokens for an external object store, ClaimGuard decouples fine-grained authorization from the cloud provider and produces a tamper-evident audit trail of all decisions. Our implementation on a local Ethereum network demonstrates that such a design can deliver sub-100 ms tail latencies and $\sim$1,000 requests/s throughput while supporting rapid policy updates and strong authorization correctness.

Future work will extend this prototype in three directions. First, we plan to integrate a conventional RBAC-based cloud deployment and systematically compare its performance, expressiveness, and operational complexity with those of other approaches. Second, we aim to evaluate ClaimGuard on real-world auto-claims datasets and investigate privacy-preserving attribute and policy encoding techniques. Third, we intend to explore multi-chain deployments and interoperability with consortium blockchains such as Hyperledger Fabric to support cross-jurisdictional claims workflows.

In addition, an extended study will (i) compare against a cloud-only RBAC baseline, (ii) evaluate consortium-chain variability and cost/fault conditions, and (iii) explore ML-assisted risk scoring and anomaly detection over access logs.

ACKNOWLEDGMENT

This work was partly supported by Innovative Human Resource Development for Local Intellectualization program through the IITP grant funded by the Korea government (MSIT) (IITP-2025-RS-2020-II201612, 33%) and by Priority Research Centers Program through the NRF funded by the MEST (2018R1A6A1A03024003, 33%) and by the MSIT, Korea, under the ITRC support program (IITP-2025-RS-2024-00438430, 34%).

REFERENCES

[1] J. Oliveira e Sá, C. Kaldeich, and M. J. Silva, "Digital transformation: A case study in the context of insurance companies," *Procedia Computer Science*, vol. 239, pp. 1165–1172, 2024, cENTERIS – International Conference on ENTERprise Information Systems / ProjMAN - International Conference on Project MANagement / HCist - International Conference on Health and Social Care Information Systems and Technologies 2023.

[2] S. Ahmad, R. Karim, N. Sultana, and R. P. Lima, "InsurTech: Digital Transformation of the Insurance Industry," in *Financial Landscape Transformation: Technological Disruptions*. Emerald Publishing Limited, 03 2025.

[3] Y. Li, Z. Du, Y. Fu, and L. Liu, "Role-Based Access Control Model for Inter-System Cross-Domain in Multi-Domain Environment," *Applied Sciences*, vol. 12, no. 24, 2022.

[4] L. A. C. Ahakonye, C. I. Nwakanma, and D.-S. Kim, "Tides of Blockchain in IoT Cybersecurity," *Sensors*, vol. 24, no. 10, p. 3111, 2024.

[5] R. Hu, Z. Ma, L. Li, P. Zuo, X. Li, J. Wei, and S. Liu, "An Access Control Scheme Based on Blockchain and Ciphertext Policy-Attribute Based Encryption," *Sensors*, vol. 23, no. 19, 2023.

[6] 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. 1, p. 146, 2024.

[7] M. Raikwar, S. Mazumdar, S. Ruj, S. Sen Gupta, A. Chattopadhyay, and K.-Y. Lam, "A Blockchain Framework for Insurance Processes," in *2018 9th IFIP International Conference on New Technologies, Mobility and Security (NTMS)*, 2018, pp. 1–4.

[8] K. Tekale and N. Rahul, "Blockchain and Smart Contracts in Claims Settlement," *International Journal of Emerging Trends in Computer Science and Information Technology*, vol. 1, no. 4, pp. 1–8, 2023.

[9] M. Kumar Thukral, "Security and Efficiency in Vehicle Insurance: A Blockchain-Based Solution," in *2023 International Conference on Self Sustainable Artificial Intelligence Systems (ICSSAS)*, 2023, pp. 1129–1136.

[10] H. Archana, M. Priti, and R. H. Bhagappa, "A blockchain framework for secured vehicle insurance claim application process," *Journal of Computational Analysis and Applications (JoCAAA)*, vol. 33, no. 07, p. 1524–1531, 2024.

[11] U. Raghav, S. Singh, S. Setia, A. Anand, and N. Singh, "Optimizing Vehicle Insurance Claims: Streamlining Settlement Processes with Blockchain Technology," in *2024 First International Conference on Technological Innovations and Advance Computing (TIACOMP)*, 2024, pp. 11–18.

[12] S. Alwis and T. M. K. K. Jinasena, "A Blockchain-Based Decentralized Insurance Platform," in *2022 International Research Conference on Smart Computing and Systems Engineering (SCSE)*, vol. 5, 2022, pp. 137–142.

[13] P.-O. Goffard and S. Loisel, "Collaborative and parametric insurance on the ethereum blockchain," 2024. [Online]. Available: https://arxiv.org/abs/2412.05321

[14] Nexus Mutual Team, "Nexus mutual documentation," 2023. [Online]. Available: https://docs.nexusmutual.io

[15] A. Paperno, V. Kravchuk, and I. Porubaev, "Teambrella: A peer-to-peer insurance system," 2017. [Online]. Available: https://teambrella.com/whitepaper.pdf

[16] A. Shetty, D. Shetty, D. Kadir *et al.*, "Blockchain application in insurance services," *SAGE Open*, vol. 12, no. 1, p. 21582440221079877, 2022.

[17] A. U. Eneh, L. A. C. Ahakonye, J. M. Lee, and D.-S. Kim, "PureAjo: A P2P Blockchain-Based Insurance Platform," in *2025 International Conference on Information and Communication Technology Convergence (ICTC)*, 2025.

[18] A. Ekblaw and A. Azaria, "A case study for blockchain in healthcare: "medrec" prototype for electronic health records and medical research data," MIT Media Lab, Tech. Rep., 2016. [Online]. Available: https://dci.mit.edu/research/blockchain

[19] Y. Xiao, B. Xu, W. Jiang *et al.*, "The healthchain blockchain for electronic health records: Development study," *Journal of Medical Internet Research*, vol. 23, no. 1, p. e13556, 2021. [Online]. Available: https://www.jmir.org/2021/1/e13556

[20] V. C. Hu, D. F. Ferraiolo, R. Kuhn, and A. Schnitzer, "Guide to attribute based access control (abac) definition and considerations (nist sp 800-162)," National Institute of Standards and Technology, Tech. Rep., 2014. [Online]. Available: https://nvlpubs.nist.gov/nistpubs/specialpublications/nist.sp.800-162.pdf

[21] S. Sajid Ullah *et al.*, "A survey on blockchain envisioned attribute based access control," *SSRN Electronic Journal*, 2023. [Online]. Available: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=4463404

[22] A. Muniswamy *et al.*, "Trust-based consensus and abac for blockchain using tdcb technology and fully homomorphic encryption," *International Journal of Computer Applications*, 2025. [Online]. Available: https://www.tandfonline.com/doi/full/10.1080/08839514.2025.2459461

[23] H. M. Levy, *Capability-Based Computer Systems*. Digital Press, 1984. [Online]. Available: http://www.bitsavers.org/pdf/dec/_Books/_Digital_Press/Levy_Capability-Based_Computer_Systems_1984.pdf

[24] N. Fotiou, V. A. Siris, and G. C. Polyzos, "Capability-based access control for multi-tenant systems using oauth 2.0 and verifiable credentials," *arXiv preprint arXiv:2104.11515*, 2021. [Online]. Available: https://arxiv.org/abs/2104.11515

[25] S. Rajasekaran, C. Saar, and K. Halachmi, "Secure isolation of tenant resources in a multi-tenant storage system using a security gateway," Patent, 2013, uS Patent 9411973. [Online]. Available: https://patents.google.com/patent/US9411973/en