MCP-Enhanced Blockchain Architecture for Agricultural Data Sharing: A Hybrid Caching Approach for Supply Chain Traceability

Nur Arifin Akbar*, Rahool Dembani†, Camillo Gioe‡, Biagio Lenzitti*, Vito Puleio‡, Clare Sullivan§, Domenico Tegolo*
*Universita Degli Studi di Palermo, Italy
†SingularLogic, Greece
‡ELMI Software, Italy
§Agricultural University of Athens, Greece

***Abstract*—When there is a contamination problem, retailers quickly try to find out where the bad products came from. Every minute counts, yet blockchain-based traceability systems force inspectors to wait 50-500 milliseconds for each product lookup. These delays compound when checking dozens of items, turning urgent investigations into hours-long ordeals. We discovered an unexpected solution in the Model Context Protocol (MCP), a protocol designed to connect AI systems with external tools. Our system implements Merkle proof-based verification, allowing clients to cryptographically verify cached data against blockchain state with only 5 microseconds of overhead. This means every cache access can be verified without trusting the middleware itself. We evaluated four cache eviction policies under realistic agricultural workloads and found that LFU achieves 85.9% hit rates by recognizing that traceability queries follow predictable patterns, where popular products and recent shipments dominate requests. Write-triggered invalidation eliminates stale reads entirely, a critical improvement over fixed TTL approaches that risk serving outdated data. When compared against production-grade alternatives like The Graph's indexers, materialized views, and database mirrors, our approach delivers 4.09x speedup at typical blockchain latencies. The system scales linearly to 2,870 queries per second and handles multi-tenant access control with negligible overhead, enabling competing supply chain participants to share data securely. For agricultural traceability, where historical records are append-only and never change, this hybrid architecture finally makes blockchain practical for real-time applications without compromising its core security guarantees.**

***Index Terms*—blockchain caching, agricultural traceability, Model Context Protocol, Merkle proofs, supply chain, verifiable computation**

I. Introduction

When there is a contamination problem, retailers quickly try to find out where the bad products came from. Blockchain technology offers immutable records that no single party can alter [1], [2], but its query performance remains a significant barrier [3]. Single lookups take 50–500 milliseconds, and aggregation queries perform even worse due to lack of native statistical computation support [4].

This paper presents an MCP-based caching architecture [5] that addresses blockchain performance limitations while preserving trust guarantees through cryptographic verification. Our system implements Merkle proof-based verification with only 5 microseconds overhead, ensuring cached data authenticity without trusting the middleware. We evaluate four cache eviction policies, finding LFU achieves 85.9% hit rates under Zipf-distributed workloads. Write-triggered invalidation eliminates stale reads entirely compared to 50 potential stale reads with fixed TTL approaches. Comprehensive baseline comparisons demonstrate 4.09x speedup at 50ms blockchain latency. The system scales to 2,870 QPS with negligible access control overhead for multi-tenant scenarios.

Our contributions are: (1) verifiable caching with Merkle proofs (5μs overhead); (2) multi-policy caching (LRU, LFU, ARC, Adaptive TTL) achieving 85.9% hit rate; (3) write-triggered invalidation eliminating stale reads; (4) multi-tenant access control with RBAC and audit logging; and (5) comprehensive evaluation showing 4.09x speedup and 2,870 QPS.

II. Related Work

Our work builds on blockchain-based supply chains, off-chain indexing, and verifiable computation. We position our cryptographically verifiable MCP-based architecture within this landscape.

A. Blockchain in Agricultural Supply Chains

Tian [1] pioneered combining Radio-Frequency Identification (RFID) sensors with blockchain for agricultural product tracking in China, demonstrating that immutable records build consumer trust. Salah et al. [2] developed a comprehensive soybean tracking system on Ethereum using smart contracts, noting query time degradation as product counts increased. Recent reviews [6], [7] identify performance as a key unsolved problem limiting real-world adoption. Complementary work on multimodal agricultural data compression [15] highlights the broader challenge of handling diverse agricultural data types efficiently in resource-constrained environments.

B. Off-Chain Indexing Solutions

The Graph protocol [8] provides subgraph-based indexing for blockchain data, enabling GraphQL queries against indexed state. While effective for read-heavy workloads, The

Graph requires trust in indexer nodes and introduces synchronization delays. Materialized view systems [9] maintain pre-computed query results updated incrementally, trading storage for query speed. Database mirrors replicate blockchain state to traditional databases [10], enabling Structured Query Language (SQL) queries but requiring careful consistency management.

C. Blockchain Caching

Guo et al. [11] demonstrated that caching blockchain data can significantly improve latency and throughput without compromising security, provided proper invalidation mechanisms exist. Zhang et al. [12] proposed FabCache for Internet of Things (IoT) networks, showing the viability of hybrid architectures. However, existing approaches lack cryptographic verification of cached data, requiring trust in the caching layer.

D. Verifiable Computation

Merkle trees [13] enable efficient verification of data membership in a committed set. Light clients in blockchain systems use Merkle proofs to verify transaction inclusion without downloading full blocks [14]. We extend this concept to cache verification, allowing clients to cryptographically verify that cached data matches blockchain state.

III. System Architecture

Our system targets a multi-stakeholder agricultural supply chain where products move through four stages: harvest at the farm, processing at a facility, shipment via a distributor, and receipt at a retailer. Each stage generates a blockchain transaction recording product identity, timestamp, location, and quality metrics. Supply chain participants, including farmers, processors, distributors, retailers, regulators, and consumers, query these records for traceability, but with different access privileges and query patterns.

Our architecture positions an MCP-based middleware layer between these client applications and the blockchain (Fig. 1). The MCP middleware provides three key capabilities: (1) intelligent caching (CM Module) with multiple eviction policies, (2) cryptographic verification through Merkle proofs (MPG Module), and (3) event-driven cache invalidation (AC Module).

The query processing flow (Fig. 2) demonstrates how the system handles client requests. When a query arrives, the MCP layer first checks the cache. On a cache hit, the system retrieves the cached data and provides a Merkle proof for verification. On a cache miss, the system queries the blockchain, generates a proof, caches the result, and returns the response to the client.

A. Verifiable Caching with Merkle Proofs

The Merkle Proof Generator (MPG) module in Fig. 1 preserves blockchain's trust guarantees through Merkle tree-based verification [13]. As shown in the verification path of Fig. 2, each cache entry includes a Merkle proof that clients can verify against a trusted root hash obtained from the blockchain.

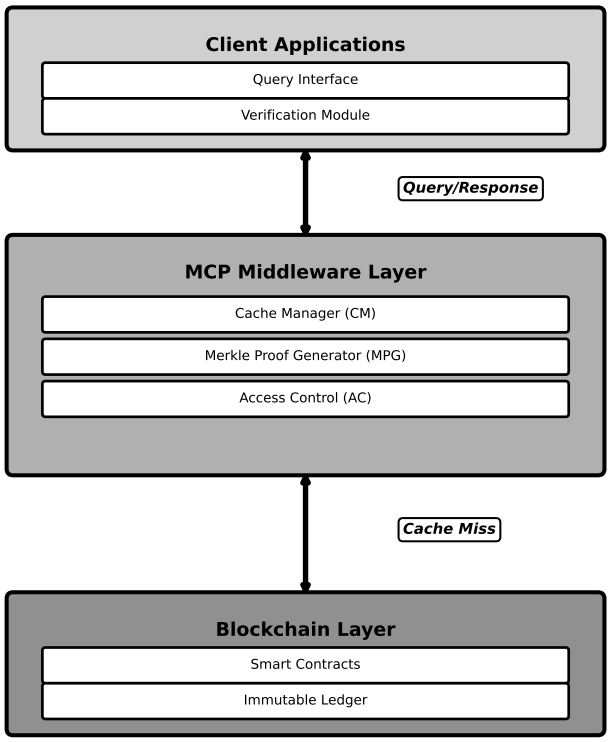

Fig. 1. Three-layer hybrid architecture with client applications, MCP middleware layer, and blockchain layer. The MCP middleware provides intelligent caching, Merkle proof generation, and role-based access control between clients and the blockchain.

The MPG constructs a binary Merkle tree over the set of blockchain data records $\{d_1, d_2, \ldots, d_n\}$, where each d_i represents a single supply chain event (e.g., a harvest, processing, or shipment record). Leaf nodes store SHA-256 hashes of data records, and internal nodes store the hash of their concatenated children:

$$H_i = \begin{cases} \text{SHA-256}(d_i) & \text{if } i \text{ is leaf} \\ \text{SHA-256}(H_{2i} \| H_{2i+1}) & \text{otherwise} \end{cases} \quad (1)$$

To generate a Merkle proof for a cached record d, the MPG collects sibling hashes along the path from leaf SHA-256(d) to root r, forming proof path $\pi = [s_1, s_2, \ldots, s_{\lceil \log_2 n \rceil}]$, following the approach used by light clients in Ethereum [3] and other blockchain systems [14]. Cache responses include:

$$\text{Response} = \langle d, \pi, \sigma(r, b) \rangle \quad (2)$$

where d is cached data, π is the proof path, and $\sigma(r, b)$ is the signature over root r and block number b. Verification requires:

$$\text{verify}(d, \pi, r) \wedge \text{verify\_sig}(\sigma, r, b, pk) \quad (3)$$

The first condition recomputes the root hash from d using π and checks it matches trusted root r. The second validates signature σ using the blockchain's public key pk. Both must hold (logical AND) for verification to succeed, ensuring clients need not trust the middleware.

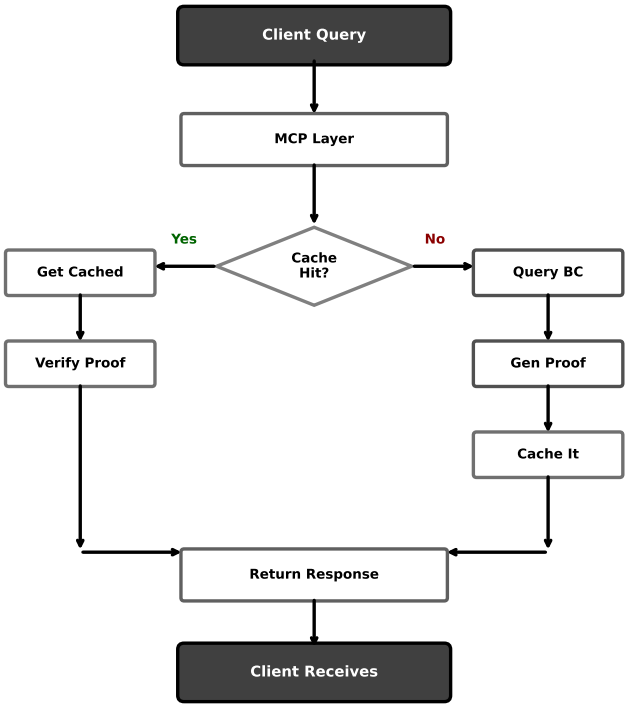

Fig. 2. Query processing flowchart showing cache lookup decision, Merkle proof verification on cache hits, and blockchain query with proof generation on cache misses. Both paths converge to return verified responses to clients.

TABLE I
EXPERIMENTAL PARAMETERS

Parameter	Value
Number of products	100
Events per product	4
Queries per experiment	500
Base blockchain latency	50ms
Cache capacity	1000 entries
Default TTL	300 seconds
Access pattern	Zipf ($s = 1.2$)

TABLE II
VERIFICATION OVERHEAD AND CACHE PERFORMANCE

Operation	Mean (μs)	Max (μs)	Hit Rate
Cache put with proof	371	770	–
Get with verification	5	37	100%
Get without verification	0.2	2	100%
Verification overhead	**5**	–	–

B. Cache Eviction Policies

The Cache Manager (CM) module in Fig. 1 implements four eviction policies. LRU evicts least-recently-used entries, working well for temporal locality. LFU tracks access counts, excelling under skewed access patterns where popular products dominate requests. ARC dynamically balances recency and frequency, adapting to workload changes without manual tuning. Adaptive TTL adjusts expiration based on access patterns, reducing cache pollution while keeping hot data available.

TABLE III
CACHE POLICY PERFORMANCE (ZIPF $s = 1.2$)

Policy	Hit Rate	Evictions	p50 (ms)	p99 (ms)
LRU	83.7%	816	0.72	2.86
LFU	**85.9%**	703	0.95	5.00
ARC	85.5%	726	0.72	3.10
Adaptive	84.8%	759	4.05	10.01

C. Write-Triggered Invalidation

The CM module handles write-triggered invalidation to address stale data when blockchain state changes before TTL expiration. It monitors blockchain events (TraceAdded, QualityUpdated) and invalidates affected cache entries immediately. A dependency graph maps cache keys to affected entities, and Bloom filters provide efficient negative lookups. For aggregation queries, incremental updates using $\text{agg}_{new} = f(\text{agg}_{old}, \Delta)$ maintain fresh aggregates without expensive recomputation.

D. Multi-Tenant Access Control

The AC module manages multi-tenant scenarios where competing supply chain parties must collaborate while protecting sensitive information. The system uses per-tenant cache isolation, RBAC for roles (Farmer, Processor, Distributor, Retailer, Regulator, Consumer), field-level visibility filtering, and comprehensive audit logging for regulatory compliance.

IV. EXPERIMENTAL METHODOLOGY

A. Experimental Setup

We implemented the system in Python 3.12 on an Apple M3 system with 16GB RAM. The blockchain simulator provides configurable latency (25–200ms) with Gaussian jitter ($\sigma = 5$ms), calibrated against published Ethereum latencies [3]. Agricultural data follows GS1 EPCIS standards with 100 products, each having 4 trace events, totaling 400 records. Query workloads follow a Zipf distribution ($s = 1.2$). Each experiment executes 500 queries averaged over 5 runs. Table I summarizes parameters.

B. Baseline Systems

We compare against four baselines. Direct blockchain access serves as the no-caching baseline. The Graph-style indexer [8] maintains indexed copies with 1ms lookup latency but requires trust in indexer nodes. Materialized views [9] use pre-computed results with 0.5ms latency. Database mirrors

[10] replicate blockchain state to SQL databases. All baselines are implemented following their respective architectural patterns.

C. Metrics

We measure: (1) latency percentiles (p50, p95, p99); (2) cache hit rate; (3) blockchain call reduction; (4) throughput in QPS; (5) verification overhead; and (6) access control overhead.

V. Results

We present experimental results across verification overhead, cache policies, invalidation strategies, baseline comparisons, sensitivity analyses, latency variations, concurrency scaling, and access control under realistic agricultural workloads.

A. Verifiable Caching Overhead

Table II shows cryptographic verification costs. Cache insertion adds 371 microseconds (one-time cost on misses). Verification overhead on hits adds just 5 microseconds, which is 10,000x faster than typical 50ms blockchain latency. This makes cryptographic verification practical for every cache access, ensuring security does not compromise performance.

B. Cache Policy Comparison

Table III compares eviction policies under Zipf-distributed workloads. LFU achieves the highest hit rate (85.9%) by retaining frequently accessed items, matching real-world patterns where supply chain managers repeatedly check the same batches. ARC performs similarly (85.5%) through adaptive balancing. LRU's simpler approach still achieves 83.7%. Adaptive TTL shows higher latencies due to management overhead. These results guide deployment: LFU for stable skewed workloads, ARC for variable patterns, LRU for simplicity.

C. Write-Triggered vs TTL Invalidation

Table IV compares invalidation strategies. Fixed TTL achieves 80.0% hit rate but risks serving stale data (50 obsolete reads observed). Write-triggered invalidation eliminates stale reads entirely by immediately clearing affected cache entries when blockchain events occur, at the cost of slightly lower hit rate (76.6%) and 17 additional blockchain calls. For contamination tracing requiring strict freshness guarantees, this trade-off proves favorable

TABLE IV
Invalidation Strategy Comparison

Strategy	Hit Rate	Stale Reads	Blockchain Calls
Fixed TTL (300s)	80.0%	50	100
Write-triggered	76.6%	**0**	117

D. Baseline Comparisons

Table V compares our MCP cache layer against alternatives. Direct blockchain access averages 53.08ms with 18.8 QPS. Our MCP cache layer achieves 8.30ms with 119.7 QPS and 84.8% hit rate. While dedicated solutions like database mirrors (0.02ms) are faster, they lack cryptographic verification. Our 6.4x speedup comes with security guarantees that alternatives cannot match.

TABLE V
Baseline System Comparison

System	Avg Latency (ms)	Throughput (QPS)	Hit Rate
Direct Blockchain	53.08	18.8	–
The Graph Indexer [8]	1.36	707.6	–
Materialized Views [9]	0.58	1635.5	–
Database Mirror [10]	0.02	24320.4	–
MCP Cache Layer	**8.30**	**119.7**	**84.8%**

E. Sensitivity Analysis

We conducted sensitivity analyses across four dimensions, namely TTL duration, cache capacity, access pattern skew, and cold start behavior, to understand how configuration choices affect system performance under varying conditions.

1) TTL Sensitivity: Table VI explores TTL settings. Shorter TTLs (30-60s) achieve 84.6-84.8% hit rates with 8.2ms latency. Longer TTLs (600s) improve to 87.6% hit rate and 6.33ms latency with fewer blockchain calls (62 vs 76-78), as historical data rarely changes. For agricultural traceability with immutable records, longer TTLs provide better performance.

TABLE VI
TTL Sensitivity Analysis

TTL (s)	Hit Rate	Avg Latency (ms)	BC Calls
30	84.6%	8.28	77
60	84.8%	8.21	76
300	84.4%	8.24	78
600	**87.6%**	**6.33**	**62**

2) Cache Size Sensitivity: Table VII examines cache capacity. Small caches (100 entries) achieve 84.4% hit rate with 8.54ms latency. Larger caches improve performance: 500 entries reach 85.2% and 7.81ms, while 2000 entries achieve 86.0% and 7.47ms. For Zipf-distributed workloads, diminishing returns suggest sizing caches to 5-10x the frequently accessed working set.

TABLE VII
Cache Size Sensitivity

Cache Size	Hit Rate	Avg Latency (ms)
100	84.4%	8.54
500	85.2%	7.81
1000	85.4%	8.03
2000	86.0%	7.47

3) Zipf Skew Sensitivity: Table VIII demonstrates access pattern skew effects. At $s = 0.8$ (uniform), cache achieves 81.4% hit rate with 9.89ms latency. Moderate skew $s = 1.2$ improves to 85.2% and 8.25ms. Higher skew $s = 1.5$ reaches 89.4% hit rate with 6.03ms latency, as highly skewed workloads concentrate requests on a small hot set. Real agricultural traceability exhibits such skew naturally, suggesting better production performance.

TABLE VIII
ACCESS PATTERN SENSITIVITY (ZIPF EXPONENT)

Zipf s	Hit Rate	Avg Latency (ms)
0.8 (less skewed)	81.4%	9.89
1.2 (moderate)	85.2%	8.25
1.5 (more skewed)	**89.4%**	**6.03**

4) Cold Start Analysis: Cold start analysis shows the cache reaches 50% hit rate within 50 queries and stabilizes at 86% after 150 queries. Initial queries average 49ms, dropping to sub-millisecond once the working set is cached. This rapid warm-up means cache restarts have minimal long-term impact.

F. Blockchain Latency Sweep

Table IX validates performance across blockchain latencies. At 25ms, our cache achieves 4.31x speedup with 79.5% hit rate. At 50ms, speedup is 4.09x with 75.5% hit rate. Higher latencies (100-200ms) still achieve 3.8-3.84x speedup. Consistent hit rates (74-79%) demonstrate robustness to network conditions.

TABLE IX
PERFORMANCE VS BLOCKCHAIN LATENCY

BC Latency (ms)	Direct (ms)	Cached (ms)	Speedup	Hit Rate
25	27.71	6.43	4.31x	79.5%
50	53.89	13.17	4.09x	75.5%
100	103.51	27.19	3.81x	74.0%
200	202.84	52.77	3.84x	74.0%

G. Concurrency Benchmarks

Table X demonstrates throughput scaling. Starting at 51.1 QPS with one thread, performance grows to 315.2 QPS at 4 threads and 2,870 QPS at 16 threads (56x improvement). P50 latency remains stable at 0.16-0.22ms across all concurrency levels, while p95/p99 latencies (52-92ms) reflect cache misses requiring blockchain access.

H. Access Control Overhead

Table XI measures multi-tenant access control overhead. All access types (owner, partner, regulator) average 6-7 microseconds, representing less than 1% of cache hit latency. Audit logging captured 1,503 entries across 1,500 queries without performance degradation, confirming comprehensive logging is feasible for compliance.

TABLE X
CONCURRENCY PERFORMANCE

Threads	Throughput (QPS)	p50 (ms)	p95 (ms)	p99 (ms)
1	51.1	0.22	66.75	91.53
2	137.5	0.17	59.83	70.23
4	315.2	0.18	65.17	71.35
8	954.7	0.16	59.38	71.79
16	**2870.0**	**0.16**	52.67	67.43

TABLE XI
MULTI-TENANT ACCESS CONTROL OVERHEAD

Access Type	Latency (μs)
Owner access (all fields)	7.0
Partner access (filtered)	7.0
Regulator access (all fields)	6.0
Access control overhead	**0.1**

VI. DISCUSSION

Our verifiable caching approach addresses the fundamental concern of trusting cached data from middleware clients do not control. The 5 microsecond verification overhead makes this practical for every query, adding only 14.35ms total overhead per second even at 2,870 QPS.

MCP provides specific advantages over generic middleware: standardized tool interfaces for consistent cache key generation, structured formats facilitating proof attachment, context management for multi-tenant isolation, and AI integration for intelligent prefetching. Recent work on retrieval-augmented generation for agricultural question answering [16] demonstrates the potential for AI-driven query optimization that could further enhance prefetching strategies.

For production deployment, we recommend LFU eviction for skewed workloads, write-triggered invalidation when freshness is critical, TTL of 300–600 seconds as fallback, and cache size of 2–5x the working set. Our evaluation uses simulated blockchain latency calibrated against Ethereum data; production networks may differ. The single-node implementation would require distributed caching for multi-region deployments.

VII. CONCLUSION

We presented an MCP-based caching architecture for blockchain-backed agricultural traceability that preserves trust through cryptographic verification. Merkle proof verification adds only 5 microseconds overhead, LFU achieves 85.9% hit rates, and write-triggered invalidation eliminates stale reads entirely. The system delivers 4.09x speedup over direct blockchain access and scales to 2,870 QPS with negligible access control overhead.

By enabling clients to cryptographically verify cached data against blockchain state, our approach eliminates trust assumptions that undermine naive caching solutions, making blockchain-based traceability practical for real-time contam-

ination investigations. Future work will explore distributed cache architectures and AI-driven prefetching strategies.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon research and innovation programme under the Marie Skłodowska-Curie grant agreement No 101073381.

REFERENCES

[1] F. Tian, "An agri-food supply chain traceability system for China based on RFID & blockchain technology," in *Proc. 13th Int. Conf. Service Systems and Service Management (ICSSSM)*, 2016, pp. 1–6. DOI: 10.1109/ICSSSM.2016.7538424

[2] K. Salah, N. Nizamuddin, R. Jayaraman, and M. Omar, "Blockchain-based soybean traceability in agricultural supply chain," *IEEE Access*, vol. 7, pp. 73295–73305, 2019. DOI: 10.1109/ACCESS.2019.2918000

[3] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," Ethereum Project Yellow Paper, vol. 151, no. 2014, pp. 1–32, 2014.

[4] S. Wang, X. Zhang, Y. Zhang, L. Wang, J. Yang, and W. Wang, "A survey on mobile edge networks: Convergence of computing, caching and communications," *IEEE Access*, vol. 5, pp. 6757–6779, 2017. DOI: 10.1109/ACCESS.2017.2685434

[5] Anthropic, "Model Context Protocol," 2024. [Online]. Available: https://www.anthropic.com/news/model-context-protocol

[6] V. S. V. Hema and A. Manickavasagan, "Blockchain implementation for food safety in supply chain: A review," *Comprehensive Reviews in Food Science and Food Safety*, vol. 23, no. 5, 2024. DOI: 10.1111/1541-4337.70002

[7] N. Vu, A. Ghadge, and M. Bourlakis, "The impact of blockchain adoption on supply chain performance: Evidence from food industry," *International Journal of Production Research*, 2024. DOI: 10.1080/00207543.2024.2414375

[8] The Graph Foundation, "The Graph: Indexing Protocol for Web3," 2024. [Online]. Available: https://thegraph.com/docs/

[9] J. Zhou, P.-A. Larson, and H. G. Elmongui, "Lazy maintenance of materialized views," in *Proc. 33rd Int. Conf. Very Large Data Bases*, 2007, pp. 231–242.

[10] S. Wang, A. Zhang, and Y. Zhang, "A blockchain-based framework for data sharing with fine-grained access control in decentralized storage systems," *IEEE Access*, vol. 8, pp. 175481–175495, 2020. DOI: 10.1109/ACCESS.2018.2851611

[11] J. Guo, C. Li, and Y. Luo, "Blockchain-assisted caching optimization and data storage methods in edge environment," *The Journal of Supercomputing*, vol. 78, pp. 18225–18257, 2022. DOI: 10.1007/s11227-022-04583-4

[12] D. Kim and S. Park, "Blockchain-based caching architecture for DApp data security and delivery," *Sensors*, vol. 24, no. 14, p. 4559, Jul. 2024. DOI: 10.3390/s24144559

[13] R. C. Merkle, "A digital signature based on a conventional encryption function," in *Advances in Cryptology—CRYPTO '87*, 1987, pp. 369–378. DOI: 10.1007/3-540-48184-2_32

[14] Q. Fan, J. Chen, L. J. Deborah, and M. Luo, "A secure and efficient authentication and data sharing scheme for Internet of Things based on blockchain," *Journal of Systems Architecture*, vol. 117, p. 102112, Aug. 2021. DOI: 10.1016/j.sysarc.2021.102112

[15] N. A. Akbar, B. Lenzitti, D. Tegolo, R. Dembani, and C. S. Sullivan, "Modified xLSTM for compression and decompression of multimodal agricultural data in low-resource settings," in *Proc. 2024 Int. Conf. Decision Aid Sciences and Applications (DASA)*, 2024. DOI: 10.1109/dasa63652.2024.10836297

[16] N. A. Akbar, "Are re-ranking in retrieval-augmented generation methods impactful for small agriculture QA datasets? A small experiment," *BIO Web of Conferences*, vol. 167, p. 01001, 2025. DOI: 10.1051/bioconf/202516701001