

Zero-Trust Blockchain-Based Tamper-Resistant Logging Framework

Prajwal Rajesh Mane*, Sagar Vyavahare**

*(Computer Science, CKT ACS College, and New Panvel (Autonomous)
Email: prajwal.mane.cs@gmail.com)

** (Computer Science, CKT ACS College, and New Panvel (Autonomous)
Email: gns.sagar@gmail.com)

Abstract:

Secure log management is critical for ensuring forensic reliability, regulatory compliance, and effective incident response in modern cybersecurity infrastructures. Conventional centralized logging systems are vulnerable to insider manipulation, unauthorized deletion, and single points of failure. To address these limitations, this paper proposes a Zero-Trust Blockchain-Based Tamper-Resistant Logging Framework that integrates SHA-256 cryptographic hashing, Merkle Tree-based batch verification, and blockchain anchoring to ensure log immutability and integrity. Each log entry is hashed and incorporated into a Merkle structure, with the Merkle Root committed to a blockchain ledger to provide decentralized, timestamped proof of integrity.

The framework enforces Zero-Trust Architecture (ZTA) principles in accordance with NIST SP 800-207 [3], incorporating strict authentication, role-based access control, and continuous verification mechanisms. Experimental evaluation demonstrates efficient processing, scalable verification complexity, and reliable tamper detection. The proposed approach eliminates centralized trust dependencies and provides a secure, transparent, and scalable logging solution for enterprise environments.

Keywords — Zero-Trust Architecture, Blockchain, Tamper-Resistant Logging, Merkle Tree, SHA-256, Log Integrity, Distributed Ledger Technology, Cybersecurity.

I. INTRODUCTION

In modern distributed computing environments, logs serve as critical evidence for cybersecurity monitoring, compliance validation, and forensic investigation. However, conventional logging architectures rely heavily on centralized storage models, making them vulnerable to unauthorized modification, deletion, and insider threats. Attackers with privileged access can manipulate logs to conceal malicious activity, thereby undermining accountability and incident response effectiveness.

Zero-Trust Architecture (ZTA) introduces a security paradigm based on the principle “Never Trust, Always Verify,” where no entity is inherently trusted, even within organizational boundaries [3]. Every

access request must be authenticated, authorized, and continuously validated.

Blockchain technology, introduced by Nakamoto [1], provides a decentralized and immutable ledger system in which data integrity is ensured through cryptographic chaining of blocks. Additionally, Merkle Trees enable efficient verification of large datasets by hierarchically combining cryptographic hashes [2].

This research integrates Zero-Trust principles with blockchain-backed integrity verification to develop a tamper-resistant logging framework that ensures:

- Cryptographic integrity
- Immutable storage
- Decentralized verification

- Transparent auditability
- Immediate tamper detection.

II. IMPLEMENTED TECHNIQUES

A) Log Collection and Normalization

Logs are collected from distributed sources including application servers, system services, authentication systems, and network devices. Each log entry is normalized into a structured format containing timestamp, source identifier, event description, and user identity. Input validation mechanisms prevent malformed data injection and ensure structured log integrity.

B) Cryptographic Hashing Using SHA-256

Each log entry LLL is processed using the SHA-256 hashing algorithm:

$$H=SHA256(L)$$

SHA-256 generates a 256-bit fixed-length digest. Due to the avalanche effect, even minor modifications to the log data produce significantly different hash outputs, ensuring strong tamper evidence [5].

C) Merkle Tree Construction

Hashed log entries are arranged into a binary Merkle Tree structure. Adjacent hashes are concatenated and re-hashed recursively until a single Merkle Root (MR) is generated:

$$MR=SHA256(H1||H2)$$

The Merkle Root uniquely represents the entire batch of logs. Verification requires only $O(\log n)$ operations, making it scalable for large datasets [2].

D) Blockchain Anchoring

The generated Merkle Root is committed to a blockchain ledger as a transaction. Each block contains:

- Previous block hash
- Timestamp
- Merkle Root
- Block hash

Because each block is cryptographically linked to the previous block, altering historical data requires recomputing all subsequent blocks, which is computationally infeasible in distributed consensus

networks [1][4].

E) Zero-Trust Access Control

The framework enforces:

- Multi-Factor Authentication (MFA)
- Role-Based Access Control (RBAC)
- JWT-based API security
- Encrypted HTTPS communication
- Continuous session verification

This ensures no user or system is implicitly trusted, aligning with NIST Zero-Trust guidelines [3].

III. MATERIALS AND METHODS

A) System Architecture

The proposed framework consists of five primary layers:

1. Log Generation Layer
2. Hashing & Merkle Processing Layer
3. Blockchain Anchoring Layer
4. Backend Verification Layer
5. Frontend Monitoring Dashboard

The backend system is implemented using Node.js and Express.js to handle log ingestion, hashing, Merkle computation, and blockchain transactions.

The frontend dashboard is developed using React.js to provide real-time log visualization, Merkle proof validation, tamper detection alerts, and blockchain block viewing.

IV. RESULT AND DISCUSSION

4.1 System Implementation Results:

The proposed Zero-Trust Blockchain-Based Tamper-Resistant Logging Framework was implemented as a full-stack application integrating backend log processing, blockchain anchoring, and a secure web-based monitoring interface. The system was evaluated under simulated enterprise logging scenarios to validate integrity enforcement, tamper detection capability, and real-time monitoring performance.

Secure Authentication and Login Module:The framework incorporates a secure login mechanism aligned with Zero-Trust Architecture principles. Users must authenticate using credential-based verification, and access is controlled through Role-Based Access Control (RBAC). JSON Web Token (JWT)-

based session management ensures secure API communication and prevents unauthorized access. During testing, unauthorized access attempts were successfully blocked, demonstrating effective enforcement of identity verification and access control policies. This confirms compliance with Zero-Trust guidelines as defined in NIST SP 800-207 [3].

Dashboard and Real-Time Log Monitoring:

Upon successful authentication, users are redirected to a centralized monitoring dashboard. The dashboard provides: Total logs generated, Total blockchain-anchored batches, Number of verified logs, Detected tamper attempts, System integrity status. A real-time log table dynamically displays newly generated log entries along with:

- Timestamp
- Source
- Log Level
- SHA-256 Hash
- Verification Status

This visualization enhances operational transparency and allows security administrators to monitor system events continuously.

Tamper Detection Demonstration:

A controlled tampering experiment was conducted to evaluate system robustness. A previously stored log entry was manually modified within the database layer.

The following observations were recorded:

- SHA-256 hash mismatch occurred
- Recomputed Merkle Root differed from blockchain-stored root
- Blockchain verification failed
- Dashboard displayed “Tamper Detected” alert
- Log verification status changed to “Compromised”.

This confirms that even a minor alteration in log content propagates through the Merkle Tree and invalidates blockchain verification, demonstrating strong tamper-evidence capability.

Blockchain Block Viewer:

The application incorporates a blockchain viewer module that displays the Block Index, Timestamp, Previous Block Hash, Merkle Root, and Current Block Hash, thereby illustrating the cryptographic linkage between consecutive blocks. Inspection of the blockchain structure confirmed its immutability property, as any modification to a historical block would require recomputation of all subsequent block hashes, which is computationally infeasible under distributed consensus assumptions [1][4].

Merkle Tree Verification and Proof Visualization:

The system organizes log entries into batches and constructs a Merkle Tree for each batch, with the computed Merkle Root anchored onto the blockchain ledger to ensure integrity. The dashboard provides a Merkle Proof verification panel that allows administrators to select a specific log entry and validate its inclusion within the blockchain-anchored batch using $O(\log n)$ verification steps. Experimental evaluation confirmed successful validation of log inclusion proofs, demonstrating the correctness of the Merkle construction process and its efficient verification complexity.

V. CONCLUSION AND RECOMMENDATION

This research successfully designed and implemented a Zero-Trust Blockchain-Based Tamper-Resistant Logging Framework that ensures log immutability, transparency, and forensic reliability.

Key Conclusions

- SHA-256 ensures cryptographic integrity.
- Merkle Trees enable scalable batch verification.
- Blockchain anchoring guarantees immutability.
- Zero-Trust enforcement eliminates implicit trust.
- Tampering attempts are immediately detectable.

Recommendations:

- Deploy on public or consortium blockchain networks for enterprise scalability.
- Integrate with SIEM platforms for real-time security analytics.

- Implement smart contracts for automated compliance verification.
- Extend framework to IoT and distributed edge environments.

ACKNOWLEDGMENT

The authors would like to thank the Department of Computer Science, CKT ACS College, New Panvel, for their support and guidance.

REFERENCES

- [1] S. Nakamoto, *Bitcoin: A Peer-to-Peer Electronic Cash System*, 2008.
- [2] R. C. Merkle, "Protocols for Public Key Cryptosystems," IEEE Symposium on Security and Privacy, 1980.
- [3] NIST SP 800-207, *Zero Trust Architecture*, National Institute of Standards and Technology, 2020.
- [4] M. Crosby et al., "Blockchain Technology: Beyond Bitcoin," Applied Innovation Review, 2016.
- [5] W. Stallings, *Cryptography and Network Security*, 7th ed., Pearson, 2017.
- [6] K. Scarfone and P. Mell, "Guide to Computer Security Log Management," NIST SP 800-92