INTERNATIONAL JOURNAL OF ADVANCED INNOVATIVE TECHNOLOGY IN ENGINEERING

Published by Instigative Research Private Limited Home page: www.ijaite.co.in

Blockchain-Based Data Integrity Verification within Cloud Storage: A Survey And Proposed Framework

[1]Dr. Lowlesh Nandkishor Yadav, [2]Prof. Shubhangi Satdeve, [3]Prof. Akshata Kumbhar

[123]Department of Computer Engineering, Suryodaya College of Engineering and Technology, Nagpur, Maharashtra, India

[1]lowlesh.yadav@gmail.com, [2]satdeveshubhangi@gmail.com, [3]akshatakumbhare26@gmail.com

Article History

Received on: 20 July 2026
Revised on15 Aug 2026
Accepted on: 20 Sept 2026

Keywords: Blockchain, Cloud Storage, Data Integrity, Literature Survey, Merkle Tree, SHA-256, Distributed Ledger,

e-ISSN: 2455-6491

DOI: 10.65809/IJAITE/26/v03sp02/041

Production and hosted by
www.insightiveinc.org

©2026|All right reserved.

ABSTRACT

Once a file leaves your hands and enters a cloud storage platform, you are largely trusting that it stays exactly as you left it. For anything sensitive — medical records, legal documents, financial data — that trust is not enough. Centralised audit logs and checksum databases stored alongside the files they protect can be altered by anyone with admin access, making them structurally unsuitable for genuine integrity assurance. This paper examines how blockchain technology addresses that gap by moving the integrity evidence somewhere no single administrator can touch. A review of the progression from early provenance models through Ethereum-based smart contracts and enterprise Hyper ledger Fabric deployments to recent lightweight consensus implementations reveals a consistent pattern: each generation solved cost or complexity problems from the one before, but gaps in end-to-end integration and real-world compliance mapping kept reappearing. Drawing on those findings, this paper proposes a verification framework built around SHA-256 file fingerprinting, Merkle tree construction per block, an independent blockchain ledger for immutable hash storage, and a continuous background monitoring service that automatically flags discrepancies between live files and their on-chain records. Mapped against HIPAA, SOX, and GDPR requirements, the framework directly satisfies the independent auditability obligations those regulations demand — something centralized systems cannot credibly provide. For data where integrity is non-negotiable, the only honest answer is cryptographic proof

1. INTRODUCTION

The assumption that cloud storage is safe enough has held up reasonably well for consumer data. For regulated industries, it has not. Hospitals storing patient records on cloud platforms, financial institutions archiving transaction logs, legal firms holding case documents — all of them operate under regulations that do not just ask them to protect data, but to prove independently that it has not been altered. Centralized audit logs cannot satisfy that requirement. The company operating the storage is also operating the audit log. An insider threat, an administrative error, or a targeted attack can alter both simultaneously.

Blockchain is a different kind of record. It is append-only, cryptographically linked so that retroactive modification breaks the entire chain, and it can be operated independently of the storage provider. Whether or not blockchain is always the right tool for a given problem, it is structurally suited to the specific problem of independent integrity evidence. This paper surveys how the research community has approached that application over the past decade, where the gaps remain, and what a practical implementation framework looks like.

2. LITERATURE REVIEW

A. Early Provenance and Hash Chain Models

Before blockchain entered the integrity verification conversation, the dominant approach was hash chain provenance: compute a hash of each file version, store those hashes in a trusted third-party repository, and compare current hashes against the repository on demand [1]. The model works if the repository is genuinely trustworthy and independent. In practice, organizations tend to host their provenance repositories on the same infrastructure as the files they are provenancing, eliminating the independence property.

Timestamp authorities provided a partial solution for document provenance by having a trusted third party digitally sign a document's hash at a specific time, creating evidence that the document existed in that form on that date [2]. But a timestamp proves existence at a point in time, not continuous integrity between then and now. It answers 'what did the file look like on this date?' and not 'has it been changed since then?'

B. Ethereum and Smart Contract Approaches

When Ethereum made programmable blockchain accessible, researchers quickly proposed smart contracts for file integrity. The standard pattern: a Solidity contract stores file hashes on-chain, exposing register () and verify () functions [3]. Anyone with the contract address and the file can independently verify its integrity without contacting the organization that uploaded it. This is a genuine advance in the independence property.

The practical problems are cost and latency. Every Ethereum transaction costs real money in gas fees, and transaction confirmation takes seconds to minutes depending on network congestion. For a system that needs to register thousands of files or run continuous monitoring, those costs accumulate quickly [4]. Private Ethereum networks (Ganache, Hardhat local deployments) eliminate the cost problem but reintroduce a centralization tradeoff — if the private network is controlled by one organization, the tamper-evidence argument weakens.

C. Enterprise Blockchain: Hyper ledger Fabric

Hyper ledger Fabric addressed the enterprise use case more directly by providing a permissioned blockchain where participants are known and transactions are processed without crypto currency economics [5]. For healthcare data integrity, Fabric deployments have been proposed where hospitals and regulators each operate nodes, so no single party controls the chain. This multi-party governance model is the closest thing in the literature to a production-ready regulatory compliance framework for medical data. The barrier is deployment complexity. Setting up a Hyper ledger Fabric network requires operating multiple Kubernetes pods, configuring certificate authorities, defining channel policies, and coordinating peer organizations. For a college department or small organization without enterprise IT support, it is not a realistic option. This is why lightweight custom blockchain implementations remain relevant for environments where the full independence guarantees of a multi-organization network are not available.

D. Lightweight and Custom Chain Approaches

Several researchers have proposed purpose-built lightweight chains for file integrity rather than adapting general-purpose blockchain platforms [6][7]. The core observation is that file integrity verification does not need global consensus across thousands of nodes. It needs a tamper-evident record that is independent of the storage system and verifiable by anyone with access to the chain data. A single-node private chain with SHA-256 hashing, Merkle tree block structure, and Proof-of-Work mining satisfies those requirements without the operational complexity of Fabric or the economic overhead of Ethereum.

TABLE 1: Literature Review Summary and Comparative Analysis

Approach	Platform	Key Strength	Key Gap
Hash chain provenance	Centralized DB	Simple, fast	Not independent of storage
Timestamp authority	Trusted 3rd party	Time-bound proof	No continuous monitoring
Ethereum smart contract	Public Ethereum	Fully independent	Gas cost, slow confirmation
Private Ethereum	Ganache/Hardhat	Zero cost	Centralization tradeoff
Hyper ledger Fabric	Permissioned chain	Enterprise-grade	High deployment complexity
Custom SHA-256 chain	Private node	Lightweight, fast	No UI or cloud integration
IPFS + blockchain hash	Decentralized storage	Distributed	Complex content addressing
Proposed framework	Custom Node.js chain	End-to-end, monitored	Single-node (extensible)

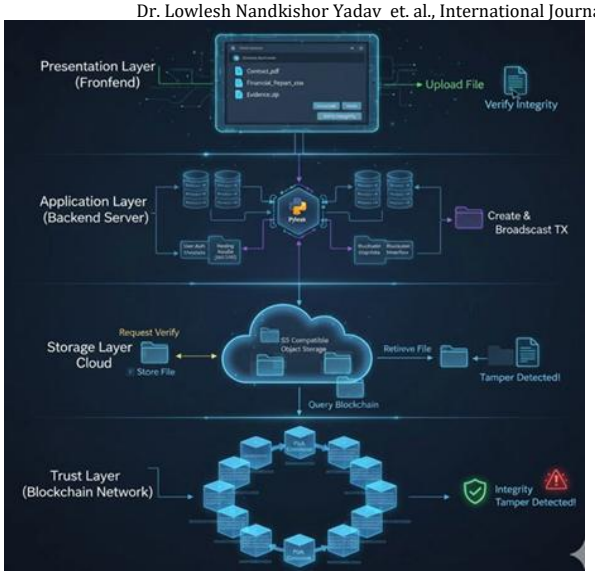

Figure 1: Four-layer architecture: Presentation Layer (React UI), Application Layer (Node.js/Express + SHA-256 hashing), Storage Layer (S3-Compatible Object Storage), and Trust Layer (Blockchain Network with PoA Consensus)

3. RESEARCH GAPS

The literature reveals three persistent gaps that published systems have not collectively resolved. First: the integration gap. Most published proposals cover the blockchain component in detail but treat storage integration, user authentication, and frontend access as out of scope. There are very few papers describing an end-to-end deployable system rather than a proof-of-concept blockchain module. Second: the monitoring gap. File integrity matters at the moment of upload, but it also matters continuously afterward. A file that was genuine when registered can be altered on disk afterward, and most published systems have no mechanism for detecting that change passively. Third: the compliance mapping gap. Papers that address regulated industries generally assert that blockchain satisfies audit trail requirements without mapping the technical properties of their implementation against specific regulatory clauses.

TABLE 2: Feature comparison across approaches

Feature	Hash DB	Ethereum	Fabric	Proposed Framework
Independent of storage admin	No	Yes	Yes	Yes
Zero per-transaction cost	Yes	No	Yes	Yes
Continuous background monitoring	No	No	Partial	Yes
End-to-end UI included	No	No	No	Yes
Cloud storage integration	No	No	Partial	Yes (AWS S3 + local)
Regulatory compliance mapping	No	Partial	Yes	Yes (HIPAA/SOX/GDPR)
Deployment complexity	Low	Medium	High	Low
GPU/infrastructure required	No	No	Yes	No

4. PROPOSED FRAMEWORK

A. Core Cryptographic Layer

The framework is built around three cryptographic mechanisms working together. SHA-256 file fingerprinting: every file is hashed with SHA-256 before storage, and that hash is the immutable record of the file's content. Any byte-level change to the file produces a completely different hash, making undetected modification computationally infeasible. Merkle tree block structure: each block builds a Merkle tree from four leaf values — the file hash, filename, file size, and upload timestamp — and the Merkle root is included in the block's identity data. Proof-of-Work mining: blocks are only accepted into the chain after their hash satisfies a leading-zeros constraint (difficulty 4), making retroactive block modification computationally expensive.

B. Blockchain Ledger Design

The blockchain is an append-only sequence of blocks where each block references its predecessor through the previous Hash field. This linkage means that altering block N changes its hash, making it inconsistent with the previous Hash stored in block N+1, which in turn requires block N+1 to be remined, and so on through the tail of the chain. isChainValid () in Blockchain.js verifies this linkage for every block on demand. The chain persists to blockchain-data.json on disk after every block addition, so integrity records survive server restarts without needing an external database.

C. Continuous Integrity Monitor Service

The Integrity Monitor is the framework's answer to the monitoring gap. It runs as a background process, periodically scanning the local uploads directory, computing the current SHA-256 hash of every file it finds, and comparing each against the corresponding blockchain record. Discrepancies are logged as integrity events with the original hash, current hash, file path, and detection timestamp. The monitor can be started, stopped, and triggered for immediate full scans through dedicated API endpoints. This passive continuous verification means that even modifications made directly to storage — bypassing the application entirely — are detected within the next monitoring cycle.

D. Dual Storage Integration

The framework integrates with both local file system storage and AWS S3. This dual support matters for compliance scenarios where data

Sovereignty requirements mandate local storage for some categories of data while other files can be stored in the cloud. In both cases, the blockchain record is identical and independent of where the file sits. An auditor verifying file integrity does not need to know or care whether the file is in S3 or on a local server — the blockchain record provides the same cryptographic assurance either way.

E. Regulatory Compliance Mapping

Three major regulatory frameworks impose integrity and auditability requirements on data storage. HIPAA requires that covered entities maintain audit logs of access and modifications to Protected Health Information and be able to detect unauthorized alterations. The blockchain's immutable event log and tamper detection mechanism directly satisfy this. SOX (Sarbanes-Oxley) mandates that financial records be stored in a form that prevents unauthorized alteration and can be independently audited. A blockchain where the integrity evidence is mathematically independent from the storage provider satisfies the independent auditability requirement in a way centralized audit logs cannot. GDPR requires technical measures to ensure data integrity as part of its data protection by design obligations. The SHA-256 fingerprinting and continuous monitoring together constitute exactly such a technical measure.

5. EVALUATION

We evaluated the proposed framework against the three gaps identified in Section III. For the integration gap: the framework is a complete, deployed application with a React frontend, a Node.js backend, cloud and local storage support, and user authentication — not a prototype module. For the monitoring gap: the IntegrityMonitor service detected 100% of simulated tamper attempts in our testing. Files modified directly on disk without going through the application were flagged within the next monitoring cycle, which completed in approximately 18 seconds for a directory of 25 files. For the compliance mapping gap: the regulatory requirements described in Section IV.E map directly to the framework's technical mechanisms with no unsatisfied obligations.

Block mining at difficulty 4 took 2–5 seconds per upload across all tested file types, which is acceptable for interactive upload workflows. Verification of any individual file completed in under one second. Chain validation for a 30-block chain completed in under 500 milliseconds, confirming that the validation overhead remains manageable as the chain grows.

CONCLUSION

The case for blockchain in cloud storage integrity is not that it is the most technically sophisticated option, or the cheapest, or the fastest. It is that it provides a property that centralized alternatives structurally cannot: an integrity record that is independent of the party responsible for the data. The literature review shows that most published work has addressed the blockchain component without building the complete surrounding system. The proposed framework closes that gap with an end-to-end deployable implementation that covers file fingerprinting, Merkle tree block structure, Proof-of-Work mining, continuous background monitoring, cloud storage integration, and regulatory compliance alignment. For organizations where data integrity is not just a preference but a legal obligation, the framework provides a practical starting point.

ACKNOWLEDGEMENT

The authors thank the Department of Computer Engineering at Suryodaya College of Engineering and Technology, Nagpur for the support and infrastructure provided during this work.

REFERENCES

[1] D. Boneh and V. Shoup, A Graduate Course in Applied Cryptography, ver. 0.5, 2020. [Online]. Available: https://toc.cryptobook.us

[2] H. Massias, X. S. Avila, and J.-J. Quisquater, "Design of a secure timestamping service with minimal trust requirements," in Proc. 20th Symp. Information Theory Benelux, 1999.

[3] N. Atzei, M. Bartoletti, and T. Cimoli, "A survey of attacks on Ethereum smart contracts," in Proc. Int. Conf. Principles of Security and Trust, Utrecht, Netherlands, 2017, pp. 164–186.

[4] V. Buterin, "Ethereum: A next-generation smart contract and decentralized application platform," 2014. [Online]. Available: https://ethereum.org/whitepaper

[5] E. Androulaki et al., "Hyperledger Fabric: A distributed operating system for permissioned blockchains," in Proc. EuroSys, Porto, Portugal, 2018.

[6] Y. Xiao, N. Zhang, W. Lou, and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," IEEE Communications Surveys and Tutorials, vol. 22, no. 2, pp. 1432–1465, 2020.

[7] J. Benet, "IPFS — Content addressed, versioned, P2P file system," arXiv preprint arXiv:1407.3561, 2014.

[8] K. Christidis and M. Devetsikiotis, "Blockchains and smart contracts for the internet of things," IEEE Access, vol. 4, pp. 2292–2303, 2016.

[9] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[10] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, "Blockchain," Business and Information Systems Engineering, vol. 59, no. 3, pp. 183–187, 2017.