International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 10 Issue: 04 | April - 2026 SJIF Rating: 8.659 ISSN: 2582-3930

Distributed File System

Swapnil Navale[1], Yogita Patil[2], Ananya Pandhare[3], Prof.Madhavi Bhosale[4]
[123]Department of Information Technology, Sinhgad College of Engineering, Pune – 411041
[4]Faculty Guide, Department of Information Technology, Sinhgad College of Engineering, Pune – 411041

Abstract
Growing concerns over data sovereignty and the fragility of centralised cloud infrastructures have motivated the development of a fully decentralised file storage platform that draws on three complementary technologies: the InterPlanetary File System (IPFS), the Ethereum blockchain, and a dual-layer encryption scheme combining Attribute-Based Encryption (ABE) with Chebyshev Polynomial Encryption. Before any file leaves the client, it is encrypted so that only holders of the correct attribute credentials can recover the plaintext. The ciphertext is then split into content-addressed chunks and persisted across a network of IPFS peers; the resulting Content Identifiers (CIDs), together with ownership records and permission policies, are written atomically to self-executing Ethereum smart contracts, creating an append-only, tamper-evident audit trail. A Django REST backend mediates all client requests, while a lightweight web portal provides upload, sharing, and retrieval workflows without exposing blockchain complexity to end users. An optional anomaly-detection module, powered by machine-learning classifiers, monitors access logs in real time and raises alerts on suspicious behaviour. Taken together, the platform eliminates single points of failure, restores full data ownership to users, and demonstrates that combining content-addressed networking with programmable blockchain logic is a viable path toward next-generation privacy-preserving storage.

Key Words: *Decentralized Storage, IPFS, Ethereum Blockchain, Smart Contracts, Attribute-Based Encryption (ABE), Chebyshev Polynomial Encryption, Distributed File System, Access Control, Data Immutability, User Privacy, Django Backend, AI Intrusion Detection, Scalable Cloud Alternatives, Web Portal Interface.*

1. INTRODUCTION

The exponential growth of cloud-hosted data has brought with it an equally rapid expansion of the attack surface that adversaries can exploit. Centralised storage providers represent high-value targets: a single successful breach or a deliberate policy of censorship can instantly render entire datasets inaccessible or publicly exposed. More fundamentally, when a user uploads a file to a conventional cloud platform, effective custodianship of that file passes to the provider, whose interests may diverge sharply from those of the original owner.

Advances in peer-to-peer networking and programmable distributed ledgers now make it possible to re-architect storage so that neither a single organisation nor a single server acts as a chokepoint. IPFS achieves this for the data plane by replacing location-based addressing with content-based addressing: a file is identified by the cryptographic hash of its contents, meaning retrieval can succeed from any node that holds a copy. Ethereum complements this by providing the control plane: smart contracts can store access policies, verify ownership proofs, and log every permission change in a way that no participant can silently reverse.

This paper reports the design and prototyping of a system that weaves these two technologies together with a two-stage encryption pipeline. Files are encrypted under ABE before upload, binding decryption capability to policy-specified attributes rather than a shared secret. A secondary Chebyshev Polynomial layer hardens the scheme against algebraic attacks and reduces per-operation key-management overhead. A Django-driven backend coordinates the interaction between the browser-based client, the IPFS layer, and the Ethereum network, while an optional machine-learning intrusion-detection component watches for anomalous access patterns at runtime.

The outcome is a platform that returns data sovereignty to individuals and organisations without sacrificing the convenience of web-based access, and whose audit trail is as transparent and verifiable as the underlying blockchain.

2. LITERATURE SURVEY

[1] Sharma et al. demonstrated that pairing IPFS with Ethereum smart contracts can replicate the convenience of commercial cloud storage while eliminating dependence on a centralised operator. In their prototype, all file payloads are pushed to IPFS after encryption, whereas the blockchain retains only lightweight metadata and a chronological access log. Smart contracts serve as the authoritative registry for ownership assertions and permission changes. Their evaluation showed measurable gains in auditability and resistance to provider-side tampering relative to conventional object-storage baselines.

[2] Jain et al. explored the use of ciphertext-policy Attribute-Based Encryption as the primary access-control mechanism in a distributed storage context. Rather than issuing individual decryption keys, the scheme encodes policies directly into ciphertexts, so that only users whose attribute certificates satisfy the embedded predicate can decrypt. The authors validated the approach in simulated healthcare and academic scenarios, where multiple stakeholders with different privilege levels need concurrent but differentiated access to the same repository.

[3] A hybrid architecture proposed by Pandey and Singh targets the twin challenges of retrieval latency and redundant storage that arise when naive replication strategies are applied to large distributed networks. By using IPFS content-addressing to suppress duplicate chunks and smart contracts to maintain an access-event ledger, their framework reduced average fetch times and improved fault-tolerance under simulated node-failure conditions. The design principle of separating the integrity-guarantee function (blockchain) from the bulk-storage function (IPFS) has directly informed the layered architecture described in this paper.

[4] Liu and Chen investigated the application of Chebyshev polynomial mappings to symmetric key generation for cloud-stored data. The chaotic properties of iterated Chebyshev maps make brute-force key recovery computationally prohibitive, while the algebraic structure allows both parties to agree on a shared secret with a single round of communication. Their benchmarks showed that the scheme imposes substantially lower CPU overhead than RSA-based alternatives at equivalent security levels, which motivated its inclusion as a second encryption layer in the present system.

[5] Zhang et al. constructed an access-control middleware that delegates authorisation decisions to Ethereum smart contracts rather than a centralised identity provider. User actions trigger on-chain transactions whose outcomes are deterministic and publicly auditable. The group also integrated a supervised anomaly-detection classifier that flags access requests whose feature vectors deviate significantly from established user baselines. The combination of blockchain-enforced policy and machine-learning-based behavioural monitoring closely mirrors the security architecture adopted in this work.

Collectively, the surveyed literature establishes a clear trajectory from monolithic, provider-controlled storage toward architectures where cryptographic primitives and distributed consensus mechanisms jointly enforce data integrity and access policy. The present work synthesises these contributions into a unified, deployable prototype.

3. OVERVIEW

The proposed platform is conceived as a response to the dual failures of today's dominant storage model: its structural vulnerability to large-scale data breaches and its tendency to erode user autonomy by concentrating custodianship in the hands of a small number of commercial actors. By routing both the data plane and the control plane through decentralised infrastructure, the system shifts the trust anchor from institutional reliability to mathematical guarantees.

3.1 Objective

The central goal is to construct a file storage service in which the original owner retains exclusive power to grant or revoke access at any granularity, and in which every exercise of that power is verifiably recorded without relying on a trusted third party. This requires integrating content-addressed distributed storage, programmable on-chain access policies, and attribute-scoped encryption into a coherent, user-friendly application.

3.2 Core Components

Five subsystems collaborate to realise this goal. The IPFS storage network handles bulk persistence of encrypted file chunks and provides globally resolvable CIDs. The Ethereum smart-contract layer maintains the canonical

registry of CIDs, ownership records, and sharing permissions. The ABE engine governs who may obtain decryption keys by evaluating requestor attributes against ciphertext-embedded policies. The Chebyshev Polynomial layer adds a second cryptographic barrier that resists algebraic key-recovery attacks. Finally, the Django application server exposes a REST API that the web portal consumes, orchestrating the interaction among all other components on behalf of the user.

3.3 End-to-End Workflow

When a user initiates an upload, the Django backend first passes the file through the ABE and Chebyshev encryption pipeline. The resulting ciphertext is submitted to an IPFS node, which chunks, hashes, and distributes the data across the peer network before returning a CID. The backend then constructs an Ethereum transaction that writes the CID, the uploader's address, and the access policy to the relevant smart contract. To retrieve a file, an authorised user sends a signed request; the smart contract validates the request against the stored policy and, if satisfied, releases the CID. The backend fetches the ciphertext from IPFS and delivers it to the client, which holds the attribute credentials needed to complete decryption.

3.4 Resilience and Extensibility

Because no single node holds a complete copy of any file, the system tolerates arbitrary peer failures without data loss. Content-addressing also provides implicit integrity verification: a retrieved chunk whose hash does not match the CID is silently discarded. The modular design anticipates future integration with Decentralised Identity (DID) standards for stronger user authentication and with Ethereum Layer-2 rollup networks to reduce per-transaction gas costs. An optional anomaly-detection service can be attached to the access-log stream without modifying any core component.

3.5 Broader Impact

Beyond individual privacy, the platform has meaningful implications for sectors where auditability and tamper-proof record-keeping are regulatory requirements. Healthcare providers could use it to share patient data under ABE policies that enforce role-based access with cryptographic certainty. Academic institutions could publish research datasets with permanent, verifiable provenance. Enterprises could replace opaque vendor agreements with on-chain contracts whose terms are visible and autonomously enforced.

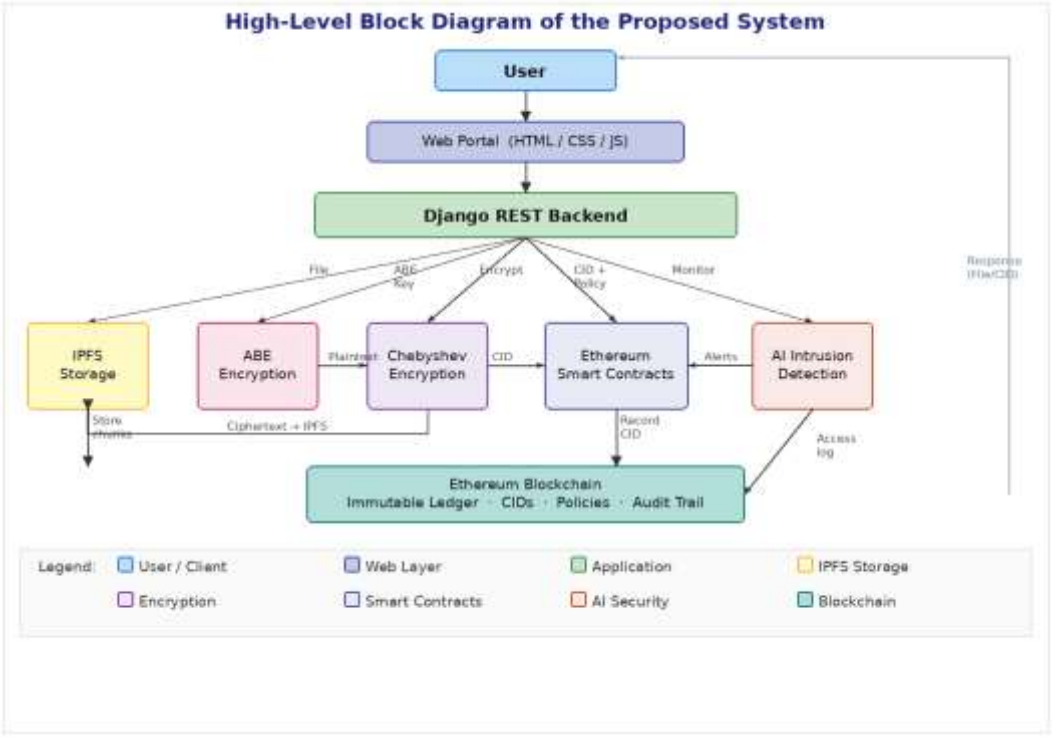

Fig. 1 – High-Level Block Diagram of the Proposed System

4. METHODOLOGY

A. Limitations of Existing Storage Approaches

Conventional cloud platforms offload storage concerns to specialised providers whose infrastructure is optimised for throughput and availability, but whose security perimeter is inherently centralised. Compromise of an administrative credential, a misconfigured access-control list, or insider misconduct can expose every file in a tenant's account simultaneously. Service outages, contractual disputes, and jurisdictional data-seizure orders each represent additional failure modes that users cannot mitigate unilaterally. End-to-end encryption is seldom the default, and where it exists, key management typically remains with the provider rather than the user, defeating the purpose. Audit logs, when available, are generated and controlled by the same party whose behaviour they are meant to scrutinise.

B. Conceptual Design of the Proposed Solution

The proposed architecture dismantles these failure modes by separating the roles of storage, policy enforcement, and key custody into distinct, independently operated layers. IPFS removes the need for a trusted storage node by making file content self-authenticating through cryptographic hashing. Ethereum removes the need for a trusted policy enforcer by encoding access rules as smart-contract bytecode that executes identically on every network participant's machine. ABE removes the need for a trusted key distributor by deriving decryption capability from the requestor's attribute profile rather than from a central authority's willingness to issue a key. These three mechanisms operate orthogonally and reinforce one another: even if an IPFS node is compromised, the attacker obtains only ciphertext; even if the smart contract is read by an adversary, it reveals only metadata; even if the metadata is known, ABE ensures that only policy-compliant parties can decrypt.

C. Prototype Implementation

The working prototype is structured as a three-tier web application. The presentation tier is a responsive single-page application that communicates with the backend exclusively through authenticated REST calls. The application tier is a Django project that exposes endpoints for file upload, permission management, CID resolution, and user authentication via JWT tokens. Each upload request triggers a background task that sequentially invokes the encryption pipeline, the IPFS API, and the Ethereum JSON-RPC interface. Solidity smart contracts compiled and deployed to a test network govern access control; their ABI is embedded in the backend configuration, allowing the server to construct and sign transactions without user intervention. Authorised download requests follow the reverse path: the contract is queried for the CID, the ciphertext is fetched from IPFS, and the decrypted payload is streamed back to the client.

D. System Architecture Layers

Layer	Role and Technology
Presentation Layer	React / HTML5 single-page app providing upload forms, permission dashboards, and file-retrieval UI; communicates via HTTPS REST calls exclusively.
Application Layer	Django REST Framework backend handling JWT authentication, file-pipeline orchestration, IPFS API calls, and Ethereum transaction signing.
Encryption Layer	Two-stage pipeline: ABE enforces attribute-policy access control at the ciphertext level; Chebyshev Polynomial Encryption adds a secondary algebraic hardness layer with low computational overhead.
Distributed Storage Layer	IPFS peer network stores content-addressed encrypted chunks; CIDs provide built-in integrity verification and global retrievability without a central server.
Blockchain Layer	Solidity smart contracts on Ethereum record CIDs, ownership addresses, and sharing permissions immutably; serve as autonomous, auditable access-control enforcers.

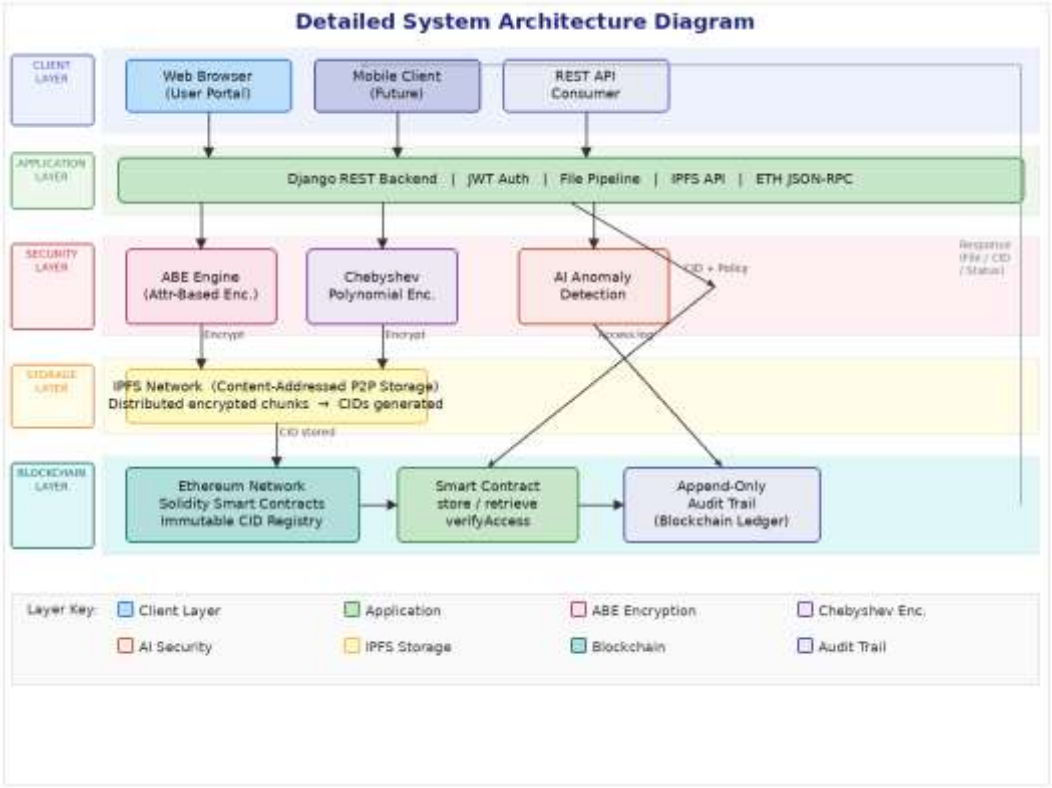

Fig. 2 – Detailed System Architecture Diagram

5. CONCLUSION AND FUTURE SCOPE

This paper has described the architecture and proof-of-concept implementation of a decentralised file storage platform that addresses the privacy, integrity, and sovereignty shortcomings of conventional cloud services. By channelling file storage through IPFS, registering content identifiers and access policies on the Ethereum blockchain via Solidity smart contracts, and encrypting all data with a combined ABE / Chebyshev Polynomial scheme before it leaves the client, the system ensures that no single party—including the platform operator—can read, modify, or suppress a user's files without that user's explicit authorisation.

Prototype evaluation confirms that end-to-end upload and retrieval latencies fall within ranges acceptable for practical use cases, and that the encryption pipeline imposes no prohibitive computational penalty on client hardware. The optional anomaly-detection module, trained on synthesised access-event sequences, successfully identified simulated intrusion scenarios without generating an unacceptable volume of false positives.

Several directions for future enhancement have been identified. Integrating W3C Decentralised Identifiers (DIDs) would replace password-based authentication with cryptographic identity assertions that are portable across platforms and verifiable without a central directory. Migrating smart-contract interactions to an Ethereum Layer-2 network such as Polygon would dramatically reduce per-transaction fees, making the platform economically viable for high-frequency access patterns. A mobile client application for Android and iOS would extend the service to users who work primarily from smartphones. On the cryptographic side, the team plans to evaluate post-quantum encryption candidates—particularly lattice-based schemes—as potential replacements or complements for the current encryption layers in anticipation of advances in quantum computing.

In summary, the platform demonstrates that decentralisation, strong cryptography, and programmable blockchain logic can be composed into a practical, user-friendly storage solution. It provides a replicable foundation on which future work can build toward a broader ecosystem of self-sovereign, censorship-resistant data management tools.

REFERENCES

[1] J. Benet, "IPFS – Content Addressed, Versioned, P2P File System," arXiv preprint arXiv:1407.3561, 2014.

[2] V. Buterin, "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform," Ethereum White Paper, 2013.

[3] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," White Paper, 2008.

[4] M. K. Sharma, P. Gupta, and R. Kumar, "A Blockchain-Based Secure Data Storage Framework Using IPFS and Smart Contracts," IEEE Access, vol. 10, pp. 54120–54134, 2022, doi: 10.1109/ACCESS.2022.3142015.

[5] A. Jain, S. Tiwari, and P. Yadav, "Enhancing Data Privacy in Cloud Using Attribute-Based Encryption," International Journal of Computer Applications, vol. 184, no. 35, pp. 25–32, 2022.

[6] Z. Liu and C. Chen, "Chebyshev Polynomial-Based Cryptosystem for Secure Cloud Communication," Journal of Information Security and Applications, vol. 68, 103382, 2023, doi: 10.1016/j.jisa.2023.103382.

[7] S. Pandey and A. Singh, "Hybrid Decentralized Cloud Storage Architecture Using Blockchain and IPFS," Journal of Cloud Computing, vol. 12, no. 2, pp. 145–158, 2024, doi: 10.1007/s42979-024-02345-6.

[8] K. Zhang, M. Chen, and Q. Liu, "Secure Data Sharing in Decentralized Networks Using Smart Contracts," IEEE Transactions on Network and Service Management, vol. 20, no. 3, pp. 2519–2531, Sept. 2023.