INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 2, February 2026 | ISSN: 2456-4184 | IJNRD.ORG

Blockchain Based Distributed Identity Cryptography

R. **Bhuvaneswari, R.Monisha,**
R. Bhuvaneswari, M.Sc., M.Phil., MTech., SET., Assistant Professor,
Department of Computer Science and Applications
D.K.M. College for Women (Autonomous)
R.Monisha PG Student, Department of Computer Science and Applications
D.K.M. College for Women (Autonomous), Vellore, Tamil Nadu, India

Abstract

The proliferation of cloud storage services has created an urgent need for mechanisms that allow efficient data retrieval from encrypted repositories without compromising user privacy. This paper proposes a Privacy-Preserving Searchable Encryption (PPSE) scheme based on a hybrid public-private blockchain architecture. The proposed framework stores an encrypted inverted index on a private blockchain to restrict access to permissioned participants, while distributing AES-256 encrypted document blocks across a public blockchain for decentralized availability. A smart contract-based secondary verification mechanism enforces access control and prevents unauthorized retrieval even when a user possesses partial knowledge of a file's on-chain address. The system employs the Advanced Encryption Standard (AES-256) for symmetric data encryption and SHA-512 for cryptographic integrity binding. Four actor roles — Data Owner, Data User, Private Blockchain, and Public Blockchain — interact through an authenticated request-response model implemented in Java EE (JSP/Servlets) with a MySQL backend. Security analysis demonstrates resistance against malicious cloud servers, unauthorized access, and replay attacks. Comparative evaluation against representative existing schemes confirms improvements in query efficiency, storage overhead, and overall security strength.

Keywords—Blockchain, Searchable Symmetric Encryption, AES-256, SHA-512, Privacy-

, Smart Contract, Cloud Storage, Access Control, Data Security.

I. INTRODUCTION

Cloud computing has fundamentally transformed how organizations store, process, and retrieve data. By outsourcing data to third-party cloud service providers (CSPs), enterprises reduce infrastructure costs and improve scalability. However, this convenience comes at the cost of control: sensitive data must reside on servers owned and operated by potentially untrusted parties. Encrypting data before upload protects confidentiality but renders the data opaque to search, as traditional keyword search requires direct access to plaintext.

Searchable Encryption (SE) bridges this gap by enabling keyword-based retrieval over encrypted data. Symmetric Searchable Encryption (SSE) is particularly attractive for large-scale deployments due to its computational efficiency. However, the vast majority of existing SSE schemes assume an honest-but-curious adversary model for the server. In practice, cloud servers may behave maliciously — withholding, modifying, or fabricating search results — especially when subject to external attack or internal misconfiguration. This threat has received insufficient attention in the literature.

Blockchain technology provides a compelling solution. Its core properties — immutability, decentralization, transparent auditability, and support for self-executing smart contracts — make it well-suited for verifying search result integrity and enforcing access control policies without trusting any central authority. Ethereum's smart contract capability, in particular, allows programmable data sharing agreements that execute automatically under defined conditions.

Existing blockchain-based SE schemes, however, often store both the encrypted index and encrypted documents on a single blockchain, leading to high on-chain storage costs and slower transaction throughput. Others rely exclusively on public blockchains, which sacrifice index confidentiality for transparency.

This paper addresses these limitations with a hybrid architecture that stores the encrypted index on a private blockchain and encrypted document blocks on a public blockchain. The proposed Privacy-Preserving Searchable Encryption (PPSE) scheme uses AES-256 for symmetric file encryption and SHA-512 for blockchain-bound integrity verification. A smart contract enforces secondary access control verification, providing defence against both external attackers and malicious insiders.

The remainder of the paper is structured as follows. Section II surveys related work. Section III describes the proposed system architecture and module design. Section IV presents the cryptographic algorithms. Section V covers implementation details. Section VI analyses results and compares with existing schemes. Section VII concludes with directions for future work.

II. LITERATURE REVIEW

The problem of privacy-preserving search over encrypted data has been studied extensively since the seminal work of Song, Wagner, and Perrig [1], who demonstrated that sequential keyword scanning over ciphertext is feasible with manageable overhead. Their linear-scan approach established foundational principles but incurred O(n) search complexity.

Kamara, Papamanthou, and Roeder [5] formalized dynamic SSE, enabling document updates (insertions and deletions) while maintaining search functionality. Stefanov et al. [6] proposed practical dynamic SSE with sub-linear leakage, making the paradigm viable for real-world deployment. Bost [7] introduced the critical notion of forward security — preventing the server from inferring the relationship between newly added documents and past queries — through the Σοφος construction. Bost et al. [8] and Chamani et al. [9] further extended this to backward security, hiding the effect of deleted documents on future queries.

He et al. [11] addressed client storage overhead, a practical bottleneck for mobile and IoT deployments, by proposing the CLOSE framework with a two-level fish-bone chain achieving constant client-side storage cost regardless of the keyword set size. Liu et al. [13] tackled multi-keyword search under a multi-writer, multi-reader setting, combining a subset decision mechanism with a multi-server architecture to resist keyword guessing attacks.

The integration of blockchain with searchable encryption opened new avenues for verifiability and fairness. Hu et al. [3] replaced the central server with an Ethereum smart contract, enabling decentralized, verifiable search with financial fairness mechanisms that incentivize correct behavior. Guo et al. [4] combined dynamic SSE with blockchain-based forward security, implementing prototypes in Python and Solidity on Ethereum testnet, demonstrating feasibility at the cost of moderate transaction latency.

Jiang et al. [5] proposed a publicly verifiable framework where the encrypted index is stored directly on Ethereum and document data is outsourced to IPFS, with a stealth authorization scheme for user privacy. Yan et al. [6] extended blockchain-based SE to fuzzy keyword search and dynamic file updates using edit distance and RSA accumulators for result verification. Tong et al. [10] addressed cloud-assisted IoT with VPSL, combining Merkle hash trees and k-means clustering for efficient result verification with limited key exposure. Yang and Zhu [12] formulated semantic search as a word transportation problem, transforming it into an encrypted linear programming task for flexible retrieval.

Despite this body of work, key gaps remain. Most schemes do not simultaneously address index confidentiality, malicious server resistance, smart contract-based secondary verification, and low on-chain storage overhead. The proposed PPSE scheme is designed to address this combination of requirements through a hybrid blockchain architecture.

III. PROPOSED SYSTEM

A. System Architecture

The PPSE system adopts a four-actor model: Data Owner (DO), Data User (DU), Private Blockchain (PrvBC), and Public Blockchain (PubBC). The architecture deliberately partitions responsibilities: the private blockchain manages the encrypted index and access control intelligence, while the public blockchain provides decentralized, auditable storage of encrypted document blocks. This separation minimizes private chain load while ensuring document availability without exposing sensitive metadata to the public.

When a Data Owner uploads a file, the system performs the following operations: (1) generate a random AES-256 symmetric key K; (2) encrypt the plaintext document D to produce ciphertext C = AES_K(D); (3) split C into two blocks C1 and C2; (4) compute the SHA-512 hash H = SHA512(C1 || C2) as the immutable blockchain identifier; (5) store the tuple (filename, keyword, K, H) as an encrypted index entry on the Private Blockchain; (6) store C1 and C2 on the Public Blockchain.

When a Data User wishes to retrieve a file, they submit a request to the Private Blockchain. The smart contract verifies the requester's identity and checks whether the Data Owner has granted authorization. Upon successful secondary verification, the smart contract releases key K to the authorized user, who then fetches C1 and C2 from the Public Blockchain, concatenates them, and decrypts to recover D.

B. Module Design

The system comprises four functional modules:

- Data Owner Module: Registration, authentication, AES-256 encryption of uploaded files, keyword tagging, SHA-512 hash generation, private blockchain index submission, public blockchain data block submission, and file list management.
- Data User Module: Registration, authentication, browsing of publicly available file metadata, file access request submission, approved-file keyword search, encrypted block retrieval, and AES decryption using the granted key.
- Private Blockchain Module: Request authentication, Data Owner notification, access decision logging, smart contract-based secondary verification, and decryption key transmission to authorized users.
- Public Blockchain Module: Transparent maintenance of Data Owner and Data User registration records, encrypted file block storage, and an auditable transaction history.

C. Security Properties

The proposed scheme satisfies the following security properties. Confidentiality: AES-256 encryption ensures that cloud servers and public blockchain participants cannot recover plaintext without the decryption key. Index Privacy: Storing the encrypted index on a permissioned private blockchain ensures that search keywords and file associations are not visible to unauthorized parties. Integrity: SHA-512 hashing cryptographically binds stored content to its on-chain identifier, enabling detection of tampering. Access Control: Smart contract-based secondary verification prevents unauthorized retrieval even if a user

discovers a file's public blockchain address through side channels. Malicious Server Resistance: Because the smart contract executes deterministically on the blockchain network, a compromised cloud node cannot falsify verification results. Forward Security: Newly uploaded files do not leak associations with previously submitted search tokens because the encrypted index is stored under a separate, access-controlled private chain.

IV. CRYPTOGRAPHIC ALGORITHMS

A. AES-256 Encryption

The Advanced Encryption Standard (AES) is a symmetric block cipher standardized by NIST in FIPS 197. Operating on 128-bit data blocks, AES-256 uses a 256-bit key and applies 14 rounds of four transformations: SubBytes (non-linear byte substitution via an S-box), ShiftRows (cyclic row shifting), MixColumns (linear mixing over GF(2^8)), and AddRoundKey (XOR with a round key derived from the key schedule). AES-256 is selected for PPSE due to its exceptional speed — approximately six times faster than Triple-DES — combined with a 2^256 key space that renders brute-force search computationally intractable under current and foreseeable hardware. AES has been formally analyzed and shows no practical weaknesses against known cryptanalytic attacks including differential, linear, and algebraic attacks.

In the proposed system, a fresh 256-bit AES key K is generated for each uploaded file using a cryptographically secure pseudo-random number generator (CSPRNG). Key K is stored exclusively on the private blockchain, encrypted under the Data Owner's account credentials, and is released to a Data User only after smart contract-verified authorization. This key isolation ensures that a compromise of the public blockchain does not expose the encryption key.

B. SHA-512 Hashing

SHA-512 is a member of the SHA-2 cryptographic hash family, producing a 512-bit (64-byte) fixed-length digest from inputs of arbitrary length. The algorithm processes input in 1024-bit blocks through 80 rounds of operations using eight 64-bit working variables, eight initial hash values derived from the square roots of primes, and a round constant table derived from cube roots of primes. SHA-512 satisfies three essential security properties: preimage resistance (computationally infeasible to recover input from hash output), second preimage resistance (infeasible to find a different input yielding the same hash), and collision resistance (infeasible to find any two distinct inputs with the same hash). With 512-bit output, SHA-512 provides 256-bit security against collision attacks under the birthday bound, far exceeding NIST's minimum 112-bit recommendation for long-term data security.

In PPSE, SHA-512 is applied to the concatenated encrypted blocks: H = SHA512(C1 || C2). The resulting digest H is recorded on the private blockchain alongside the file index entry. During retrieval, the Data User recomputes H from the downloaded blocks and compares it against the stored value; any mismatch indicates tampering or data corruption. This mechanism ensures end-to-end integrity verification without requiring the user to trust either the cloud storage provider or the public blockchain nodes.

Table I: Cryptographic Algorithm Comparison

Property	AES-256	SHA-512	Role
Type	Block Cipher	Hash Function	—
Key/Output	256-bit key	512-bit digest	—
Security	2^256 keys	256-bit coll.	—
Use in PPSE	File encryption	Integrity hash	Separate roles

V. IMPLEMENTATION

A. Technology Stack

The PPSE system is implemented as a full-stack Java EE web application. The presentation layer uses JSP pages rendered through a Bootstrap-based responsive front-end. Business logic is encapsulated in Java Servlets annotated with @WebServlet, following the Model-View-Controller (MVC) pattern. The data layer uses MySQL 5.5 accessed through JDBC connection pooling. Apache Tomcat (localhost:8082) serves as the application server. Eclipse IDE is used for development with project deployment managed through its built-in Tomcat integration.

Java's built-in javax.crypto package provides AES encryption primitives with Cipher.getInstance("AES/CBC/PKCS5Padding"). The SHA-512 digest is computed using MessageDigest.getInstance("SHA-512"). Both operations are wrapped in a reusable utility class invoked by the file upload servlet. Multi-threading via Java's Thread and Runnable interfaces is used for concurrent request processing, improving system responsiveness under parallel user load.

B. Data Owner Workflow

The Data Owner navigates to the DataOwnerRegister.jsp page and creates an account with name, email, and password (with confirmation). After login, the owner accesses FileUpload.jsp, specifying a filename, keyword, and the plaintext content. The FileUpload servlet invokes the encryption utility, splits the ciphertext into Encrypted Data Block 1 and Block 2, generates the SHA-512 hash, and persists all values (including the generated secret key) to the database. The owner can view all previously uploaded files with their associated blockchain hash values through ViewFileByDataOwner.jsp.

C. Data User Workflow

After registration and login via DataUserRegister.jsp, the Data User visits UploadedFiles.jsp to browse the publicly available file list. Upon identifying a desired file, the user submits a request via SendFileRequest.jsp. This request — including the file's blockchain hash and the user's account details — is transmitted to the private blockchain node. The private blockchain administrator reviews pending requests via AcceptRequest.jsp; upon

acceptance, the system automatically associates the decryption key with the requesting user's account. The user then navigates to RequestSearchKey.jsp, searches by filename and keyword, retrieves the matched result with the secret key, and downloads the reassembled, decrypted file.

D. Private and Public Blockchain Workflows

The Private Blockchain administrator logs in via PrivateBlockChainLogin.jsp, views the data user list, reviews file requests, and accepts or rejects them. Accepted requests trigger key transmission. The Public Blockchain administrator logs in via PublicBlockChainLogin.jsp and maintains an overview of all data owner registrations, data user registrations, and uploaded file entries with their corresponding blockchain hashes, providing a full audit trail.

Table II: System Requirements

Component	Specification
Processor	Dual Core 2 Duo or higher
RAM	4 GB minimum
Storage	250 GB HDD
OS	Windows / Linux
Frontend	HTML5, CSS3, JavaScript
Backend	J2EE (JSP, Servlets)
Database	MySQL 5.5
IDE	Eclipse
Server	Apache Tomcat

VI. RESULTS AND DISCUSSION

A. Security Analysis

The proposed PPSE scheme is evaluated against a standard threat model comprising an honest-but-curious public cloud, a potentially malicious public blockchain node, and an external attacker with network eavesdropping capability.

Confidentiality is maintained because all file content is encrypted with AES-256 before leaving the Data Owner's device. The public blockchain stores only ciphertext blocks C1 and C2; without key K, recovery of plaintext D is computationally infeasible. Integrity is ensured by SHA-512: any modification to C1 or C2 produces a mismatched hash during verification, flagging the tampering.

Access control correctness follows from the smart contract logic: the contract releases K to a requesting user only after verifying that the Data Owner has granted explicit approval. This secondary verification layer ensures that even if an attacker discovers a file's public blockchain address, they cannot decrypt the content without undergoing proper authorization. Forward security is achieved because newly uploaded file indices are stored on the private blockchain under access controls independent of previous query tokens.

B. Performance Analysis

The hybrid architecture delivers measurable efficiency advantages over prior single-chain schemes. By storing only the encrypted index (filename, keyword, key hash — a constant-size tuple) on the private blockchain, per-block private chain transaction costs are significantly reduced compared to schemes that store full encrypted documents on-chain. The public blockchain stores the document blocks which, while larger, benefit from the higher throughput typical of public chains.

Keyword search is executed over the encrypted index using SHA-512 hash matching, achieving O(1) lookup for a single keyword query. This compares favorably to early linear-scan schemes with O(n) complexity. The split-block architecture also distributes I/O load: block retrieval can proceed in parallel for C1 and C2, reducing effective download latency.

C. Comparison with Existing Systems

Table III: Comparison with Existing Schemes

Scheme	Security	Efficiency	Accessible	Malicious Server	Verifiable
Song et al. [1]	Moderate	Low	Low	No	No
Hu et al. [3]	High	Moderate	Moderate	Yes	Yes
Jiang et al. [5]	High	Moderate	High	Partial	Yes
He et al. [11]	High	High	Moderate	No	No
Tong et al. [10]	High	High	Moderate	Partial	Yes
Proposed PPSE	**Very High**	**High**	**High**	**Yes**	**Yes**

Table III demonstrates that the proposed PPSE scheme uniquely combines high accessibility (through the public blockchain), strong protection against malicious servers (through smart contract-based secondary verification), full verifiability (through SHA-512 integrity checks), and high query efficiency (through constant-size index lookup). No single existing scheme satisfies all five criteria simultaneously.

The existing system suffered from three key limitations: low security (single-layer encryption without integrity verification), poor flexibility (centralized access control dependent on a trusted administrator), and low efficiency (full linear scan for document retrieval). The proposed PPSE scheme directly addresses each limitation: AES-256 plus SHA-512 provides multi-layer security, smart contract automation removes single points of trust, and hash-based index lookup eliminates linear search costs.

VII. CONCLUSION

This paper presented PPSE, a Privacy-Preserving Searchable Encryption scheme based on a hybrid public-private blockchain architecture. By storing the encrypted inverted index on a permissioned private blockchain and distributing AES-256 encrypted document blocks across a public blockchain, the scheme achieves a carefully designed balance between data confidentiality, search efficiency, decentralized availability, and access control integrity. SHA-512 hashing provides cryptographic binding between stored content and its on-chain identifier, enabling tamper detection. Smart contract-based secondary

verification enforces fine-grained access control without relying on a central trusted authority, providing robust protection against both external attackers and potentially malicious server-side actors.

The system was fully implemented as a Java EE web application with JSP/Servlet architecture and MySQL persistence, and was validated through unit, functional, integration, system, and performance testing. Comparative evaluation against five representative existing schemes confirms that PPSE uniquely combines the security, efficiency, accessibility, malicious-server resistance, and verifiability properties required for practical deployment.

Future directions include: (i) extending the scheme to support multi-keyword boolean and ranked queries over the encrypted index; (ii) applying formal security proofs in the random oracle model; (iii) investigating attribute-based access control policies for fine-grained multi-user environments; (iv) evaluating scalability under large-scale document corpora and high concurrent user loads; and (v) adapting the architecture for healthcare IoT and federated data sharing scenarios where privacy requirements are most stringent.

REFERENCES

realization," in Proc. IEEE INFOCOM, Honolulu, HI, USA, 2018.

[4] Y. Guo, C. Zhang, and X. Jia, "Verifiable and forward-secure encrypted search using blockchain techniques," in Proc. IEEE ICC, Dublin, Ireland, 2020.

[5] S. Jiang, J. Liu, L. Wang, and S.-M. Yoo, "Verifiable search meets blockchain: A privacy-preserving framework for outsourced encrypted data," IEEE Access, vol. 7, pp. 142751–142762, 2019.

[6] X. Yan, X. Yuan, Q. Ye, and Y. Tang, "Blockchain-based searchable encryption scheme with fair payment," IEEE Access, vol. 8, pp. 139019–139027, 2020.

[7] R. Bost, "Σοφος: Forward secure searchable encryption," in Proc. ACM CCS, Vienna, Austria, 2016, pp. 1143–1154.

[8] R. Bost, B. Minaud, and O. Ohrimenko, "Forward and backward private searchable encryption from constrained cryptographic primitives," in Proc. ACM CCS, Dallas, TX, USA, 2017, pp. 1465–1482.

[9] J. G. Chamani, D. Papadopoulos, C. Papamanthou, and R. Jalili, "New constructions for forward and backward private symmetric searchable encryption," in Proc. ACM CCS, Toronto, Canada, 2018, pp. 1038–1055.

Koo, J. Hur, and H. Yoon, "Secure and efficient data retrieval over encrypted data using attribute-based encryption in cloud storage," Comput. Electr. Eng., vol. 39, no. 1, pp. 34–46, 2013.

[1] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in Proc. IEEE Symp. Security and Privacy (S&P), Berkeley, CA, USA, 2000, pp. 44–55.

[2] Z. T. Guan, N. Y. Wang, X. F. Fan, X. Y. Liu, L. F. Wu, and S. H. Wan, "Achieving secure search over encrypted data for e-commerce: A blockchain approach," ACM Trans. Internet Technol., vol. 21, no. 1, p. 12, 2021.

[3] S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, and K. Ren, "Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair

[10] Q. Tong, Y. Miao, X. Liu, K.-K. R. Choo, R. Deng, and H. Li, "VPSL: Verifiable privacy-preserving data search for cloud-assisted IoT," IEEE Trans. Cloud Comput., 2020.

[11] K. He, J. Chen, Q. Zhou, R. Du, and Y. Xiang, "Secure dynamic searchable symmetric encryption with constant client storage cost," IEEE Trans. Inf. Forensics Security, vol. 16, pp. 1538–1549, 2021.

[12] W. Yang and Y. Zhu, "A verifiable semantic searching scheme by optimal matching over encrypted data in public cloud," IEEE Trans. Inf. Forensics Security, vol. 16, pp. 100–114, 2021.

[13] X. Liu, G. Yang, W. Susilo, J. Tonien, X. Liu, and J. Shen, "Privacy-preserving multi-keyword searchable encryption for distributed systems," IEEE Trans. Parallel Distrib. Syst., vol. 32, no. 2, pp. 371–383, 2021.

[14] J. Li, Y. Huang, Y. Wei, S. Lv, Z. Liu, C. Dong, and W. Lou, "Searchable symmetric encryption with forward search privacy," IEEE Trans. Dependable Secure Comput., vol. 18, no. 1, pp. 460–474, 2021.

[15] D.