IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940



International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511157

Privacy-Preserving Intelligent CollaborationPlatform with Searchable Encryption andClient Side AI

Dr. Kothapalli Venkata Rao[1], Priya H M[2], Saniya Fatima[3], and Varsha R[4]

Professor & Head, Dept. of CSE, K.S. School of Engineering & Management, Bengaluru, India[1]

Student, Dept of CSE, K.S. School of Engineering & Management, Bengaluru, India[2-4]

Abstract: This paper presents a collaboration platform that keeps confidential information safe by processing it on the user's own device, ahead of any transfer to cloud servers. In typical cloud services the provider can read content while it is being stored, indexed or analysed by AI features. To remove this exposure, our design brings together AES encryption of files on the client, keyword-derived protected tokens that make encrypted content searchable, RSA wrapping of file keys for exchange between users, and compact AI models that run on the handset to produce document summaries and suggestions. The system is built as an Expo React Native app targeting Android and the Web. Clerk handles sign-in, while cryptographic keys are held in the device's SecureStore. A Node.js and Express server coordinates requests, encrypted files sit in private Supabase Storage, and PostgreSQL accessed through Drizzle ORM keeps the metadata. The server is only ever given ciphertext, metadata and encrypted keys; no readable file reaches it. The workspace dashboard, file-management and secure-upload screens show a privacy-first workflow in which authorised users can store and share files while the server remains unable to view their contents.

Keywords: Collaboration Privacy, Encrypted Keyword Search, On-Device AI, AES, RSA Key Exchange, Secure Sharing, Cloud Security

I. INTRODUCTION

Storing and exchanging documents in the cloud has become routine, and team collaboration now relies heavily on such services. However, the provider of a typical service can see user content when it is stored, indexed or fed to AI features. In effect, collaborators must give up readable copies of their files, which leaves confidential material open to servers and outside parties.

Existing systems suffer from four weaknesses. Servers can read the files that users upload. Searching often needs either decrypted content or an index kept in plaintext. AI features can ship private content away from the device. Finally, sharing files securely demands careful handling of keys. What is needed, therefore, is a design in which encryption, search and AI help all function without the server ever seeing plaintext.

Our work tackles these weaknesses by shifting the sensitive steps onto the client. Encryption is applied to files before they are uploaded, queries are answered through protected search information, and AI is meant to run on the device itself. The guiding idea is simple: data should be secured before it ever reaches the cloud. The goals are to encrypt files on the client, keep cloud storage closed to unauthorised parties, allow searching over encrypted files, run AI locally, let users share files securely, and strengthen privacy across collaborative tools.

The system offers these main capabilities:

- **On-Device Encryption:** AES is applied to each file on the user's device prior to upload, hence the server is never given readable data.
- **Protected Storage:** Ciphertext stays in a private cloud bucket, and only authenticated requests are allowed to access it.
- **Encrypted Search:** Protected tokens derived from keywords extracted from a file let users find it without revealing what it contains.
- **Local AI:** Compact AI models execute on the user's own device to produce summaries and suggestions.
- **Key-Wrapped Sharing:** A file's AES key is wrapped using the recipient's RSA public key; after the key exchange, authorised users decrypt the file on their own devices.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511157

- **Team Workspace:** Users can handle teammates, invitations, shared files, notifications and chat; chat content is kept in encrypted form.

II. RELATED WORK

A. Keyword Search over Public-Key Ciphertext

Boneh et al. [1] introduced Public Key Encryption with Keyword Search (PEKS), a scheme in which a server can look for keywords inside data held as public-key ciphertext. Its drawbacks are heavy computation, poor scalability and restricted query expressiveness. For this reason we adopt a lighter token-based search that is better suited to practical, mobile settings. Later studies point the same way: a survey on searchable symmetric encryption [6] reviews techniques for securing outsourced cloud data, and a review of searchable encryption features covering 2019–2023 [7] observes that efficiency gains are appearing as several functions are merged.

B. Symmetric Searchable Schemes for Cloud Data

Chen et al. [2] employed Dynamic Searchable Symmetric Encryption (Dynamic SSE) to enable efficient, privacy-aware search on cloud-stored data, but the scheme leaks search patterns and is complex to implement. Our response is protected search tokens combined with a light design for mobile clients. Gui et al. [8] likewise observe that SSE schemes tolerate a degree of leakage to gain speed, and urge a fresh look at how complete SSE systems are constructed; this observation underpins our protected-token approach.

C. Privacy-Aware AI on Mobile Devices

Nguyen et al. [3] relied on federated learning, training models across devices so that mobile AI keeps data private. Such training brings heavy communication costs and intricate handling of model updates. Our platform sidesteps this by restricting AI to inference on the client, so no substantial on-device training is needed. A survey of federated learning for 6G services [9] similarly observes that sending model updates consumes many resources and adds communication load. Looking at AI that runs on the device itself, a survey on mobile large language models [10] warns that such deployments handle sensitive data and carry security and privacy risks of their own, while a mobile edge intelligence survey [11] reports that local models are economical, responsive and more private than cloud ones but constrained by edge hardware.

D. Combining AES and RSA for Sharing

Patel and Sharma [4] paired AES with RSA in a hybrid scheme for sharing data safely in the cloud, although managing keys is complex and overall speed suffers. In our design AES protects the file contents and RSA is used only to pass keys, which keeps sharing quick and uncomplicated. A survey of hybrid cryptography for file storage in the cloud [12] outlines the same division, assigning AES to symmetric and RSA to asymmetric encryption and listing user authentication and key management as necessary functions, while Ahirwar and Tyagi [13] build on it with RSA key exchange, AES-OTP encryption, time-bound access control and adaptive key management. Why key handling deserves care is illustrated by a cryptographic study of five end-to-end encrypted cloud storage services [14]; against a malicious server, many of them proved inadequate, for instance by allowing the server to supply inauthentic keys during file sharing or to alter metadata.

E. Lightweight Cryptography on Constrained Devices

Singh and Verma [5] examined lightweight cryptographic primitives for resource-limited devices and found a distinct trade-off between strength of security and speed. We pair lightweight encryption with on-device AI so that privacy is preserved while the app stays usable. The Ascon-based lightweight cryptography standard from NIST [15], finalized in 2025, echoes the demand for algorithms fit for constrained hardware. Overall, the surveyed studies leave one need unmet: encryption, search and AI help that operate together without exposing plaintext to the server.

III. SYSTEM DESIGN AND METHODOLOGY

The architecture has three parts: a client app, a server that only ever sees protected data, and a database for structured metadata. The client is a cross-platform Expo React Native application that runs on Android and in the browser, in which Expo Router drives navigation, Clerk covers authentication and sessions, and SecureStore holds cryptographic keys on the device. Its interfaces comprise Home, Files, Upload, Chat, Search, Team and Notifications. The server uses Node.js with Express REST APIs, with Clerk guarding user-specific endpoints, and it administers users, files, sharing, invitations, chat and notifications. Encrypted files go straight to private Supabase Storage, and the server deals solely with ciphertext, metadata and encrypted key material, never plaintext.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511157

Structured data is managed in PostgreSQL through Drizzle ORM, which holds metadata about users, files, teammates, invitations and notifications, with chat kept as encrypted message data. Public keys of registered devices are saved in the user_device_keys table, whereas device-specific wrapped keys are held in the message_device_keys and file_share_device_keys tables.

As Fig. 1 shows, the system follows a client-to-cloud privacy pipeline. Encryption and indexing of files happen on the user's device ahead of upload; the server keeps just ciphertext and encrypted tokens and matches queries blindly; and results come back encrypted, to be decrypted and handled by the client-side AI.

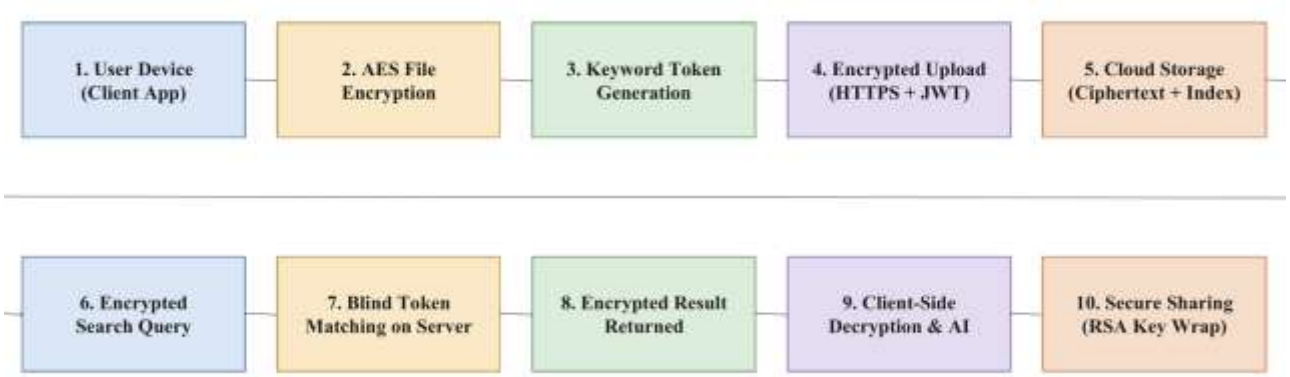


Fig. 1 System pipeline for privacy-preserving storage, encrypted search and client-side AI

A. Protection and Upload Phase

As Fig. 1 illustrates, the first half of the pipeline secures data before it leaves the device and has five steps:

1. User Device: All sensitive operations run in the client app on the user's device, starting with a local scan of the chosen file.

2. AES File Encryption: Before upload, the file is encrypted on the device with AES.

3. Keyword Token Generation: Keywords are extracted and turned into protected tokens, so searching needs neither decrypted data nor plaintext indexes.

4. Encrypted Upload: The encrypted file travels over HTTPS with JWT authentication and goes directly into private Supabase Storage.

5. Cloud Storage: Only the ciphertext and the encrypted index are kept in the cloud; the backend sees solely ciphertext, metadata and encrypted key material.

B. Search, Sharing and On-Device AI Phase

The latter half of the pipeline allows users to search, share and work with files while plaintext stays hidden:

6. Encrypted Search Query: The query is protected on the user's device before being transmitted to the server.

7. Blind Token Matching on Server: The server compares the protected query with the stored tokens and learns neither the keywords nor the file contents.

8. Encrypted Result Returned: The server returns results to the client strictly in encrypted form.

9. Client-Side Decryption and AI: The client decrypts the results locally, and compact AI models there generate summaries and suggestions.

10. Secure Sharing (RSA Key Wrap): Sharing a file means wrapping its AES key with the recipient's RSA public key. After this key exchange, authorised users decrypt the file on their own devices using device-specific wrapped keys.

 This work is licensed under a Creative Commons Attribution 4.0 International License

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026

DOI: 10.17148/IJARCCE.2026.1511157

The layered architecture of the implementation is summarised in Fig. 2. Android Studio or VS Code together with Git served as development tools. Security and privacy rely on AES, RSA, searchable-encryption tokens and JWT, and the client-side AI uses compact on-device models for summaries and suggestions. The hardware needed is an Android or iOS phone plus a laptop or desktop with at least 8 GB of RAM and an i5-class or equivalent processor.

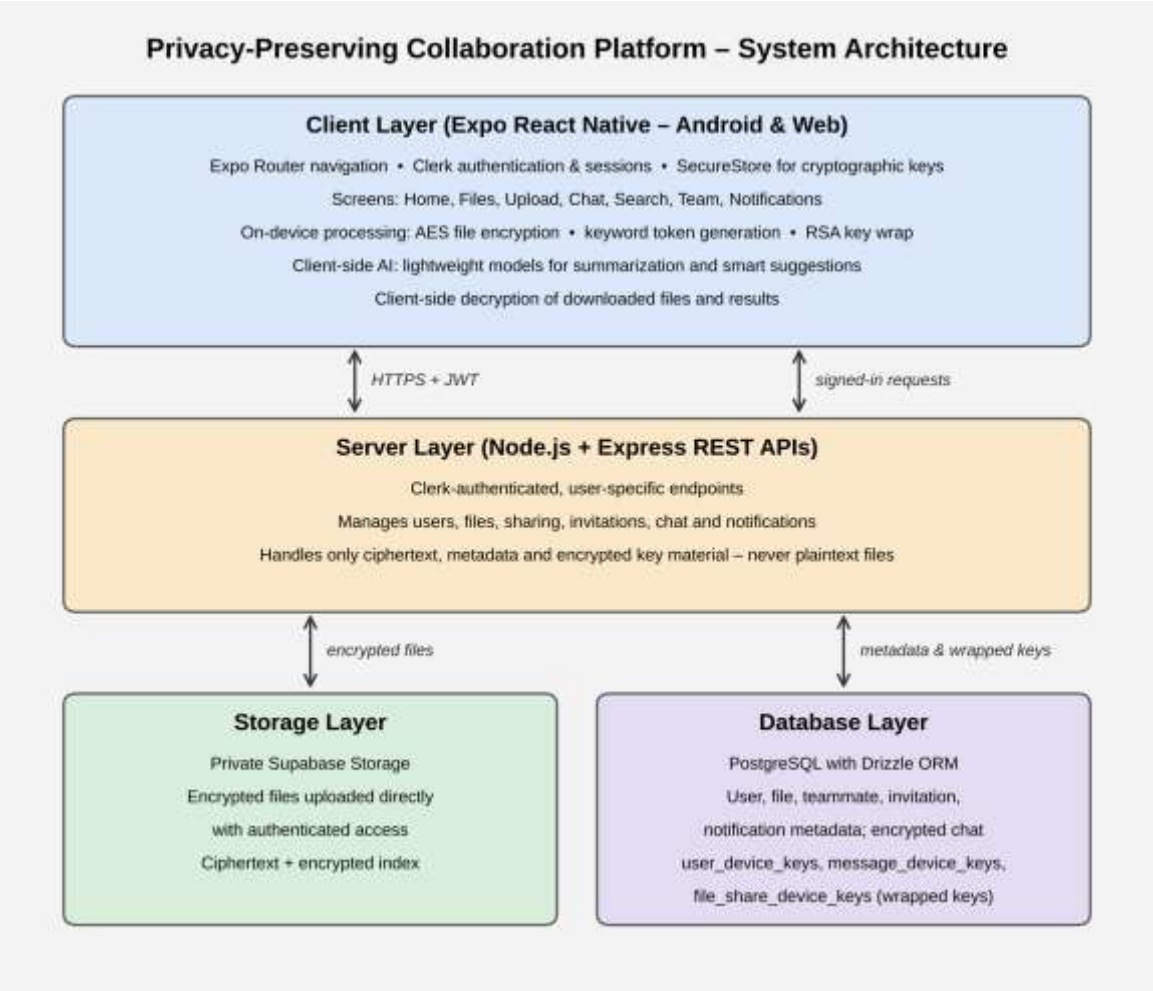


Fig. 2 High level design

IV. RESULTS AND DISCUSSION

We ran the application on an Android device and examined its principal screens. The workspace dashboard shows that encryption is active, along with a security indicator and the secure storage in use, given as capacity and percentage consumed. The workspace offers quick actions for Upload, Team, Share and AI Assist, while the bottom bar gives access to Home, Files, Upload, Chat and Search.

On the file-management screen, each file is listed with its type, size and date modified. The screen supports file search and filters for All, PDF, DOCX, PPTX and XLSX, and a lock icon next to each file marks it as protected. Users thus get a dedicated space for managing protected files.

The secure upload screen steps through local scanning, local encryption and secure upload, and indicates completion once the encrypted file has been uploaded. These screens demonstrate that protection on the client is part of ordinary file handling and not a separate utility.

Relative to the approaches reviewed, the platform avoids burdening the user with heavy computation: lightweight tokens replace public-key keyword search, AI performs inference on the client without any on-device training, and RSA is reserved for exchanging keys. At present the implementation supports Android and Web; iOS, more file types and more advanced search across large collections of encrypted files are left for future work.

 This work is licensed under a Creative Commons Attribution 4.0 International License



International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 11, November 2026
DOI: 10.17148/IJARCCE.2026.1511157

V. CONCLUDING REMARKS AND FUTURE SCOPE

This work delivers a secure setting for messaging, storing files and sharing them. Data are encrypted on the client before the server receives them, and encrypted search lets files be located without disclosing their content. Authorised users get controlled file sharing, while plaintext and sensitive keys never leave user devices. By uniting encryption, privacy and teamwork, the platform lowers the chance of unauthorised exposure of data.

Planned extensions are: a voice assistant for secure spoken commands to look up files and send messages; fingerprint and face recognition for access control; wider format support covering PDF, DOCX, PPTX, XLSX and images; an iOS version for cross-platform teamwork; and faster, more capable search over large sets of encrypted files.

ACKNOWLEDGEMENT

We would like to express our heartfelt gratitude to the Department of Computer Science and Engineering, KSSEM, Bengaluru, for giving us the facilities and the encouraging atmosphere we needed to carry out this project. Our special thanks go to our guide, Dr. K. Venkata Rao, whose patient advice, timely feedback and constant motivation kept us on track from the first idea to the final write-up. We are also thankful to the cryptography and privacy researchers whose published work shaped our understanding and influenced many of the design choices made here.

REFERENCES

[1]. D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, “Public Key Encryption with Keyword Search,” in Advances in Cryptology - EUROCRYPT 2004, Springer, 2004.
[2]. Y. Chen, J. Li, and K. Zhang, “Efficient and Privacy-Preserving Searchable Encryption for Cloud Storage,” IEEE Access, 2024.
[3]. T. Nguyen, M. Pathak, et al., “Federated Learning for Privacy-Preserving AI on Mobile Devices,” IEEE Communications Surveys & Tutorials, 2024.
[4]. R. Patel and A. Sharma, “Privacy-Preserving Data Sharing in Cloud using Hybrid Encryption,” Springer, 2025.
[5]. P. Singh and D. Verma, “Lightweight Cryptographic Techniques for Mobile Cloud Security,” 2025.
[6]. F. Li, J. Ma, Y. Miao, X. Liu, J. Ning, and R. H. Deng, “A Survey on Searchable Symmetric Encryption,” ACM Computing Surveys, vol. 56, no. 5, pp. 1–42, 2024, doi: 10.1145/3617991.
[7]. B. Kishiyama and I. Alsmadi, “A Review on Searchable Encryption Functionality and the Evaluation of Homomorphic Encryption,” arXiv:2312.14434, Dec. 2023.
[8]. Z. Gui, K. G. Paterson, and S. Patranabis, “Rethinking Searchable Symmetric Encryption,” in Proc. 2023 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 2023, pp. 1401–1418.
[9]. M. Ben Driss, E. Sabir, H. Elbiaze, and W. Saad, “Federated Learning for 6G: Paradigms, Taxonomy, Recent Advances and Insights,” arXiv:2312.04688, Dec. 2023.
[10]. H. Xu, K. Li, W. Chen, D. Zheng, Z. Li, and Z. Cai, “A Survey: Towards Privacy and Security in Mobile Large Language Models,” arXiv:2509.02411, Sep. 2025.
[11]. G. Qu, Q. Chen, W. Wei, Z. Lin, X. Chen, and K. Huang, “Mobile Edge Intelligence for Large Language Models: A Contemporary Survey,” arXiv:2407.18921, 2024 (rev. 2025).
[12]. A. Chaudhari, “A Survey on Hybrid Cryptography for Secure File Storage on the Cloud,” International Journal for Research in Applied Science and Engineering Technology (IJRASET), 2023, doi: 10.22214/ijraset.2023.54089.
[13]. M. Ahirwar and M. Tyagi, “A Novel Secure Cloud Storage Solution: Combining AES-OTP, RSA, and Time-Limited Access Control with Adaptive Key Management,” Current Trends in Signal Processing, vol. 14, no. 3, pp. 11–20, 2024.
[14]. J. Hofmann and K. T. Truong, “End-to-End Encrypted Cloud Storage in the Wild: A Broken Ecosystem,” in Proc. 31st ACM Conference on Computer and Communications Security (CCS), 2024, doi: 10.1145/3658644.3690309.
[15]. M. S. Turan, K. A. McKay, D. Chang, J. Kang, and J. Kelsey, “Ascon-Based Lightweight Cryptography Standards for Constrained Devices: Authenticated Encryption, Hash, and Extendable Output Functions,” NIST Special Publication 800-232, Aug. 2025, doi: 10.6028/NIST.SP.800-232.

 This work is licensed under a Creative Commons Attribution 4.0 International License