Privacy-Preserving Cloud Storage with Searchable Encryption and Steganographic Key Management

Project Synopsis Submitted

to

MANIPAL ACADEMY OF HIGHER EDUCATION

For Partial Fulfillment of the Requirement for the

Award of the Degree

Of

Bachelor of Technology

in

Computer and Communication Engineering

by

Ishan Satyanand Thakur, Vedant Agarwal, Mayur R Das

Reg. No. 230953356, Reg. No. 230953312, Reg. No. 230953414,

Under the guidance of

Dr. Divya Rao (Lab Faculty 1)
Associate Professor
School of Computer Engineering
Manipal Institute of Technology
MAHE, Manipal, Karnataka, India

Ms. Pragya Jha (Lab Faculty 2)
Assistant Professor
School of Computer Engineering
Manipal Institute of Technology
MAHE, Manipal, Karnataka, India

MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
A Constituent Unit of MAHE, Manipal

August 2025

Objective:

To build a secure cloud storage system where files are AES-encrypted before upload, keyword search works over encrypted indexes (privacy-preserving), and all user keys + index mappings are stored steganographically in the user's profile picture which is delivered once and stored locally — eliminating repeated key transmission and minimizing interception risk.

Scope:

- AES-256 encryption for files and SHA-256 integrity checks.
- Searchable Symmetric Encryption (SSE) for encrypted keyword search.
- Steganographic storage of per-user AES keys and encrypted index mapping inside the user's profile picture (one-time secure delivery).
- Client-side key extraction and decryption (no keys sent with downloads).
- 2FA authentication, role-based access, tamper-proof audit logs.
- Deployment on Microsoft Azure (Blob Storage, CosmosDB/MongoDB, Key Vault, App Service).

Need for the Application:

Even with encrypted files, cloud systems remain vulnerable when keys or searchable indexes are exposed, or when keys are transmitted with files. Attackers can intercept keys during transit or compromise DBs to retrieve keys. This project prevents those attack vectors by:

- Never transmitting keys during regular operations.
- Hiding keys + encrypted index mapping inside profile pictures (steganography).
- Requiring client-side extraction of keys, so intercepted files are useless without the local stego image and extraction routine.

Project Description:

Problem statement

Cloud storage commonly leaks sensitive information through exposed keys, indexes, or repeated key transmissions. Searchable encryption schemes help, but key distribution and where indexes live remain attack surfaces. The project solves: (1) how to search encrypted content privately, (2) how to manage keys without sending them repeatedly, and (3) how to avoid revealing sensitive metadata.

Proposed solution

- Files are encrypted with AES-256 and stored in Azure Blob Storage.
- Keywords are extracted, hashed, and encrypted to build a search index.
- The encrypted index mapping (Enc(keyword) → BlobID) and file AES keys are themselves encrypted and embedded inside the user's profile picture using steganography.
- The profile picture (stego container) is delivered once securely at registration (or securely when updated) and stored on the user's device. The server does not resend it on each search/download.

- During search, the client sends an encrypted search token (Enc(keyword)). The server matches against encrypted indexes (extracted from DB images if necessary) and returns only the encrypted file (BlobID). The user uses their locally stored profile picture to extract the AES key and decrypt locally.

Functionalities to be implemented

User Module

- Register/login with 2FA using email.
- Generate or accept a profile picture; embed user keys/index mapping into it during registration (client or server-assisted) and deliver it once over TLS for local storage.
- Profile pic update flow when new keys are needed.

File Storage Module

- AES-256 encrypt files locally prior to upload.
- Compute & store SHA-256 digest for integrity.
- Store encrypted files in Azure Blob Storage (BlobID returned).

Indexing Module

- Keyword extraction (basic NLP/tokenization, stop words removal, using Google Cloud Natural Language API).
- Hash (SHA-256) + encrypt keywords to form secure index tokens.
- Map Enc(keyword) → BlobID and store this small map inside the user's profile picture (or as encrypted blobs referenced by image metadata).

Search Module

- Client-side: hash & encrypt query → send encrypted token over HTTPS.
- Server-side: match token to encrypted indexes (no plaintext revealed).
- Return encrypted file BlobID only.

Steganography Module

- Embed encrypted key material and index map into a benign JPEG/PNG profile image with minimal visual distortion.
- Extraction routines on client side to retrieve keys and mapping.
- Support small payloads (a few KB), resilient to common image operations (within reason).

Key Update / Revocation

- When user uploads a new file, client can update the local profile picture (Option B) and re-upload the updated profile image once over TLS, or server can produce updated stego-image and prompt a one-time secure download (Option A).

- Implement key rotation and revocation mechanisms: mark old keys invalid in server metadata, require clients to fetch updated stego-image once after revocation.

Audit Module

- Tamper-evident logs (signed timestamps) of uploads, searches, downloads, profile-image updates.

Security measures against interception & compromise

- All network ops use TLS/HTTPS.
- Search tokens are encrypted client-side; server never sees plaintext queries.
- Files in transit are already AES-encrypted; keys are not transmitted with files.
- Profile picture (stego) is delivered only once over TLS and stored locally; subsequent operations do not require re-fetching it.
- Stego payload itself is AES-encrypted before embedding (double encryption).
- Key rotation & revocation supported; old keys discarded and new stego-image delivered once via secure channel.

Hardware Requirements:

- CPU: Intel i5 / AMD Ryzen 5 or better
- RAM: 8 GB minimum
- Storage: 500 GB (local + cloud)
- Network: Broadband internet for cloud interaction

Software Requirements:

- Frontend: React.js, HTML, CSS
- Backend: Node.js (Express) + Python (Flask/FastAPI for crypto & stego modules)
- Database: MongoDB / Azure CosmosDB (GridFS for storing images)
- Cloud: Microsoft Azure — Blob Storage, Key Vault, App Service
- Crypto libs: PyCryptodome / cryptography, bcrypt, JWT
- Stego libs: Pillow, Stegano / custom LSB + compression-aware embedding
- API: Google Cloud Natural Language API
- Dev tools: VS Code, Git, Postman (debugging and testing)

Submitted by:

Name	Registration number	Roll Number	Semester & Branch	Section
Ishan Satyanand Thakur	230953356	42	V (CCE)	C
Vedant Agarwal	230953312	36	V (CCE)	C
Mayur R Das	230953414	47	V (CCE)	C