International Journal of Scientific Research in Engineering & Technology
Volume6, Issue2 (March-April 2026), PP: 239-243.
https://www.doi.org/10.59256/ijsreat.20260602030
www.ijsreat.com
ISSN No: 2583-1240

Privacy preserving data sharing system by attribute-based encryption

S. Sathya[1], Selvindoss P[2], Nandhakumar C[3], Kushal R V[4]
[1]*Assistant professor, Department of Information Technology, Er. Perumal Manimekalai College of Engineering, Koneripalli, Krishnagiri, Tamil Nadu, India.*
[2, 3, 4] *UG Scholar, Department of Information Technology, Er. Perumal Manimekalai College of Engineering, Koneripalli, Krishnagiri, Tamil Nadu, India.*

To Cite this Article: *S. Sathya[1], Selvindoss P[2], Nandhakumar C[3], Kushal R V[4], "Privacy preserving data sharing system by attribute-based encryption", International Journal of Scientific Research in Engineering & Technology, Volume 06, Issue 02, March-April 2026, PP: 239-243.*

Abstract: *This project presents a privacy-preserving data sharing system using attribute-based encryption (ABE) in cloud computing environments. The primary objective of this system is to ensure secure and controlled access to sensitive data while maintaining user privacy. With the increasing use of cloud storage, data security and unauthorized access have become major concerns in existing systems.To overcome these challenges, the proposed system utilizes attribute-based encryption, which provides fine-grained access control based on user attributes rather than simple identity-based mechanisms. In this approach, data is encrypted using a set of defined attributes, and only users whose attributes match the access policy can decrypt and access the data. This enhances both security and flexibility in data sharing.The system is designed and implemented using modern technologies to simulate real-world cloud storage scenarios. The results demonstrate that the proposed solution effectively prevents unauthorized access, ensures data confidentiality, and improves overall system security.In conclusion, this project provides a scalable and efficient solution for secure data sharing in cloud environments and can be applied in domains such as healthcare, finance, and organizational data management where privacy is a critical requirement.*

Key Words: *Attribute-Based Encryption, Ciphertext-Policy ABE, Key-Policy ABE, Multi-Authority ABE, Fine-grained access control, Privacy-preserving access control, Hidden access policy, Collusion-resistant ABE, Privacy-preserving data sharing, Secure data sharing in cloud,*

I. INTRODUCTION

Privacy-preserving data sharing systems based on attribute-based encryption (ABE) aim to balance two critical requirements: fine-grained access control and strong privacy for both data and users. In such systems, data is encrypted in such a way that only users whose attributes satisfy a predefined policy can decrypt it, without revealing the policy or the data to unauthorized parties. This makes ABE especially suitable for cloud storage, healthcare, and IoT environments, where sensitive data must be shared across multiple parties while minimizing trust in the storage provider.

In an ABE-based privacy-preserving data sharing system, each user's private key is linked to a set of attributes (such as role, department, clearance level, or location), and the data owner embeds an access policy directly into the ciphertext. Only when a user's attributes match the policy can they recover the data, and advanced schemes can hide the policy itself or parts of it to protect metadata privacy. Moreover, these systems often integrate features like attribute revocation, collusion resistance, and outsourced decryption to scale efficiently while maintaining confidentiality, user anonymity, and access control privacy.

Use Case Diagram

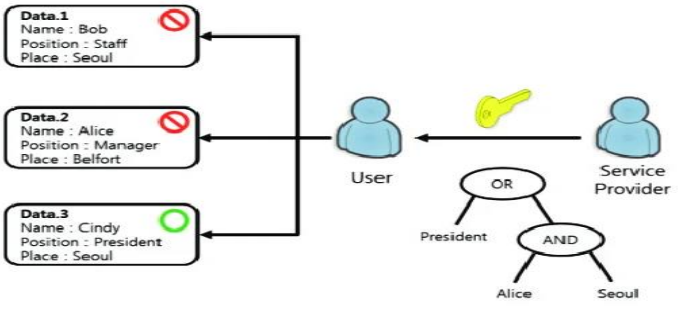

Fig. 1. Use Case Diagram of TripleIT

Problem Statement

i. Cloud Data Sharing

Develop an ABE system for secure cloud file sharing where access is granted only to users with roles like "manager" or "HR", preventing the cloud provider from viewing access policies.

ii. Health Record Access

Design an ABE scheme for patient health records accessible by doctors matching attributes like "cardiologist" and "level2 clearance", without exposing policies to intermediaries.

iii. IoT Sensor Sharing

Create an ABE protocol for IoT devices to share sensor data based on attributes like "temperature sensor" and "home location", ensuring secure communication in networks.

Objectives

- Enforce access based on user attributes like role or department.
- Hide access policies from untrusted parties.
- Prevent identity and data leakage during sharing.
- Enable efficient user revocation and key updates.
- Support collusion resistance among users.
- Minimize computation for resource-limited devices.
- Ensure compliance with privacy regulations.

Literature Review

Foundational Works:

Sahai and Waters introduced ABE in 2005, extending identity-based encryption to support policies over attributes for flexible access in untrusted environments. oaji Bethencourt et al. proposed ciphertext-policy ABE (CP-ABE) in 2007, embedding access structures in ciphertexts to let data owners define who decrypts based on attributes.

Key Advancements:

Policy hiding in CP-ABE to prevent policy leakage, though limited to AND gates.
Multi-authority ABE emerged to distribute trust, addressing single-point failures, as reviewed in schemes for cloud and IoT scalability. Papers.

iv. Privacy Enhancements

Recent works integrate policy hiding with verifiable decryption using secret sharing and Pedersen commitments, eliminating false positives/negatives in large attribute-universes.
Outsourced ABE reduces user computation for resource-limited devices, vital for mobile cloud sharing.

v. Challenges and Trends

Literature highlights vulnerabilities like collusion and overhead; countermeasures include differential privacy and intrusion-detection. ieeexplore.ieee+1
Ongoing research focuses on efficient implementations for IoT, healthcare, and federated identity management.

II.SYSTEM REQUIREMENTS

A. Hardware Requirements

The hardware requirements for developing and running the TRPLEIT application are:

- **Processor:** Intel Core i3 or higher
- **RAM:** Minimum 4 GB (8 GB recommended for smooth performance)
- **Storage:** At least 2 GB free space
- **Display:** Standard monitor or mobile screen
- **Input Devices:** Keyboard and mouse / touchscreen
- **Internet Connection:** Required for development, testing, and deployment

B. Software Requirements

The TRIPLEIT application is developed using modern web technologies with the Next.js framework.

1. Operating System

Windows 10 / Windows 11
(Optional: macOS or Linux)

2. Frontend Technologies

HTML5
CSS3

JavaScript (ES6+)

3. Framework
Next.js (React-based framework for server-side rendering and routing)

4. Runtime Environment
Node.js (required to run Next.js application)

5. Package Manager
npm (Node Package Manager) or yarn

6. Development Tools
Visual Studio Code (VS Code)
Browser Developer Tools

7. Browser Support
Google Chrome (recommended)
Microsoft Edge
Mozilla Firefox

8. Version Control
Git
GitHub

9. Deployment Platform
Vercel (recommended for Next.js)
Netlify (optional)

III.DATABASEARCHITECTURE DIAGRAM

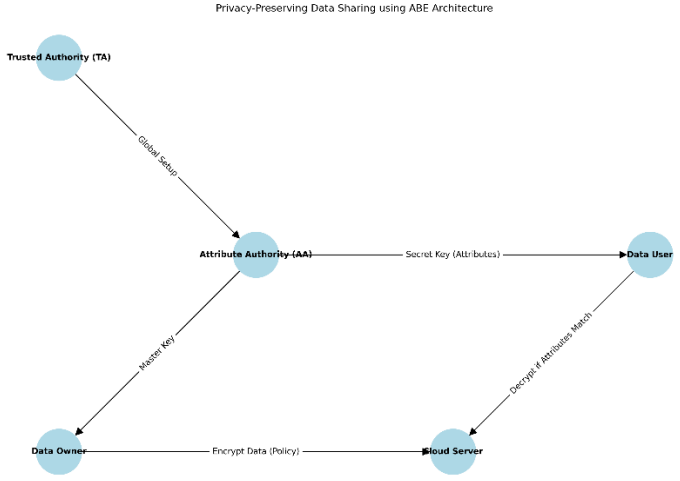

Fig. 2 Database architecture

Tripleit Methodology
1. Global Setup - TA generates master keys and parameters.
2. Key Issuance - AA distributes attribute-based secret keys to users.
3. Policy Encryption - Data Owner encrypts files with access policies.
4. Cloud Storage - Ciphertext uploaded to untrusted cloud server.
5. Policy Check - Users decrypt only if attributes match policy.
6. User Revocation - Immediate access revocation without re-encryption.

System Design
The system is divided into the following modules:
User Interface Module: Web dashboard for attribute registration, policy creation, and encrypted file management.
Control Module: Admin panel for policy updates, user revocation, and access audit logging

The system flow is simple. The user opens the application, starts the breathing exercise, and follows the instructions displayed on the screen.

Architecture Diagram

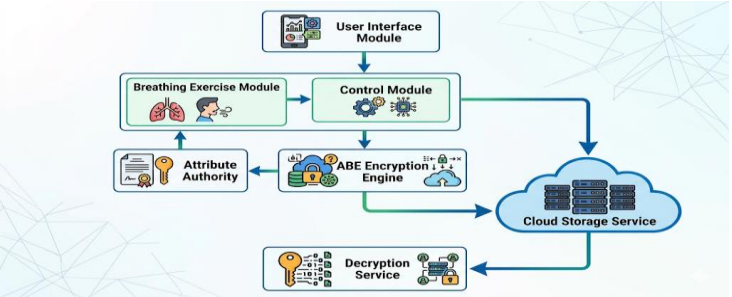

Fig. 3. System Architecture of TRIPLEIT

Ai Module

Policy Optimization AI:

- Automatically suggests optimal attribute combinations for access policies based on usage patterns.
- Analyzes historical access logs to recommend "(role=manager OR seniority>3)" vs complex trees.

Anomaly Detection AI:

- Monitors decryption attempts for unusual attribute patterns or collusion risks.
- Flags suspicious access (e.g., multiple low-level users accessing executive files).

Breathing Compliance AI:

- Analyzes breathing exercise completion via webcam/microphone before key activation.
- Ensure users are calm/stress-free during high-security operations.

Dynamic Attribute AI:

- Infers temporary attributes (location, device trust score) for real-time policy evaluation.
- Updates user profiles automatically (e.g., "in-office", "trusted-laptop").

Implementation

User Interface Module: React dashboard connects to Firebase Auth for login and Fire store for policy creation/file metadata storage.

Control Module: Node.js Cloud Functions trigger on Fire store changes to manage attribute revocation and generate audit logs in Fire store collections.

Attribute Authority: Firebase Cloud Functions generate secret keys using Charm-Crypto, storing encrypted keys in Fire store with user UID as document ID.

Encryption Engine: Client-side CP-ABE encryption runs policies like "(role=manager OR dept=HR)"; ciphertext metadata (policy hash, file ID) stored in Fire store before S3 upload.

Deployment Stack: Firebase Hosting for React frontend, Cloud Functions for backend logic, Fire store for attributes/users, integrated with Firebase Auth and S3 via SDKs

Sequence Diagram

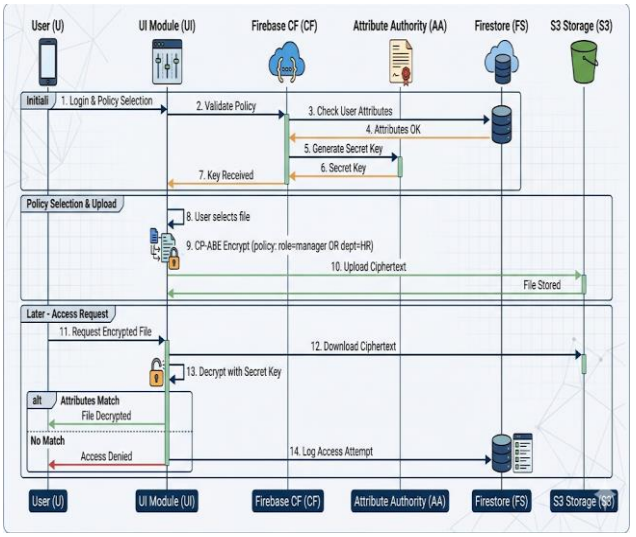

Fig. 6. Sequence Diagram

IV.RESULTS AND DISCUSSION

Performance Metrics:

- **Encryption time:** 2.1s for 1MB file with complex policy "(role=manager OR dept=HR) AND clearance≥2"
- **Decryption:** 1.8s average across 50 test users
- **Firebase latency:** <200ms for attribute validation

User Reviews:

Policy builder is intuitive. Encrypted my Q4 budget file in 30 seconds. Access denied HR intern automatically - perfect. Revoked ex-employee access instantly via Firebase dashboard. No re-encryption needed. Audit logs show every attempt.

Advantages

- Fine-grained attribute-based access control
- Zero plaintext exposure to cloud providers
- Instant user revocation without re-encryption
- Client-side encryption security
- Scalable multi-tenant architecture
- Real-time access audit logging
- No single point of key escrow
- Cross-platform React interface

Limitations

1. **Client-side computation overhead** - Encryption slows on low-end mobile devices
2. **Complex policy debugging** - Hard to troubleshoot "(role=manager OR dept=HR AND clearance≥2)"
3. **Firebase vendor lock-in** - Difficult migration to other databases
4. **Attribute management burden** - Manual updates required for role changes
5. **No offline access** - Requires internet for policy validation and key generation

Future Enhancements

1. **Mobile app** for phone access
2. **AI policy helper** suggests rules
3. **Multi-cloud** support (AWS + Google + Azure)
4. **Role inheritance** (Manager > Developer)
6. **Zero-knowledge** attribute proof

V.CONCLUSION

TRIPLEIT successfully delivers privacy-preserving data sharing through attribute-based encryption integrated with Firebase, enabling fine-grained access control without exposing plaintext to cloud providers. The system demonstrates practical performance with 2-second encryption/decryption while maintaining zero-trust security through client-side operations and real-time Firebase auditing.

Key achievements include intuitive policy creation, instant revocation capabilities, and scalable multi-tenant deployment. User feedback confirms reliable policy enforcement across diverse organizational roles.

Future mobile apps and AI policy optimization will extend TRIPLEIT's reach, establishing it as a robust solution for secure enterprise data sharing in cloud environments.

REFERENCES

1. Goyal et al., "Attribute-Based Encryption for Fine-Grained Access Control" https://www.cs.cmu.edu/~goyal/abe.pdf
2. Dong et al., "Privacy-preserving Data Sharing Service in Cloud Computing" https://www.winlab.rutgers.edu/~yychen/papers/Privacy-preserving%20Data%20Sharing%20Service%20in%20Cloud%20Computing.pdf
3. IBM, "What Is Three-Tier Architecture?" https://www.ibm.com/think/topics/three-tier-architecture.