International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

Volume 13, Issue 3, 2025

Attribute-Based Encryption Approach For Storage, Sharing And Retrieval Of Data

Dr M Sandhya Rani, D Pranisha Reddy, Ch Udaya Sree, G Vibha Maithreyi

[1]Associate Professor, Department of Information Technology, Bhoj Reddy Engineering College for Women

[2.3.4]B,tech students, Department of Information Technology, Bhoj Reddy Engineering College for Women

pranishareddy21@gmail.com

ABSTRACT

In the era of cloud computing, secure data storage and controlled data access are critical challenges. Traditional encryption techniques, while offering data confidentiality, often hinder efficient data retrieval and lack fine-grained access control. This project presents FABECS (Fully Attribute-Based Encryption Scheme for Cloud Storage), a secure, efficient, and privacy-preserving framework that enables encrypted data storage, searchable encryption, and attribute-based access control. FABECS incorporates an index-based searchable encryption mechanism that allows data users to perform secure keyword searches without revealing content to the cloud service provider. The system ensures only authorized users can decrypt and access relevant files based on predefined attributes, offering fine-grained security, low computational overhead, and scalability. Developed using Python and Django, and deployed on a cloud server, FABECS effectively addresses the limitations of existing systems while aligning with modern data privacy requirements.

Keywords: Attribute-Based Encryption (ABE), Ciphertext-Policy ABE (CP-ABE), Searchable Encryption, Fine-Grained Access Control, Privacy-Preserving Retrieval, Encrypted Indexing, Secure Data Sharing, Cloud Security, Data Confidentiality.

1. INTRODUCTION

Cloud storage has become one of the most cost-effective services for businesses and individuals. However, storing sensitive data on untrusted cloud servers raises security and privacy concerns. Users need to encrypt their data before outsourcing it, but traditional encryption methods limit searchability and sharing [1], [2]. The challenge is to store, share, and retrieve encrypted data efficiently while maintaining security. Existing systems either require heavy computational power for searching encrypted data [3] or lack fine-grained access control [4], making them impractical. This project proposes a secure, searchable, and access-controlled cloud storage system using Attribute-Based Encryption (ABE) [5].

Existing system:

Traditional cloud storage systems rely on encryption to protect sensitive data before outsourcing it to the cloud [1]. However, these systems face several challenges, such as limited searchability, inefficient access control, and high computational overhead [6], [3]. Users must download encrypted data, decrypt it, search for relevant information, and then re encrypt and upload it, making the process slow and resource-intensive. Moreover, conventional encryption methods lack fine-grained access control, meaning all authorized users either have full access or none, leading to security risks. Additionally, centralized key management systems are prone to attacks, increasing the risk of unauthorized access [8].

Proposed System:

We propose FABECS (Fully Attribute-Based Encryption Scheme for Cloud Storage, Sharing, and Retrieval), a comprehensive security framework that ensures secure and efficient data management in cloud environments. FABECS leverages Attribute-Based Encryption (ABE) to provide fine-grained access control, enabling data owners to encrypt documents under specific attribute-based policies and distribute keys securely through a trusted authority. To support efficient retrieval without compromising data privacy, FABECS employs an encrypted, index-based structure [3], [9] that allows data users (DUs) to locate and query relevant documents using trapdoors generated from their attributes, all without revealing search terms or access patterns to the Cloud Service Provider (CSP) [10]. The scheme ensures that only authorized users can access or search encrypted data, thus maintaining confidentiality and access control simultaneously. Furthermore, FABECS prioritizes the retrieval of the most relevant documents based on query relevance, optimizing user experience while maintaining strong security guarantees. Importantly, the system adheres to modern cryptographic standards by enforcing encryption levels of at least 128 bits [5], thereby complying with industry-recommended security benchmarks. Through these mechanisms, FABECS effectively meets the demands of secure storage, controlled sharing, and efficient retrieval in cloud-based environments.

2-RELATED WORK

Cloud storage systems have become widely adopted due to their convenience and scalability. However, storing sensitive data on untrusted servers poses significant security and privacy risks. To address these issues, various encryption techniques have been developed. Traditional encryption methods like symmetric and public-key encryption provide basic data confidentiality but are limited when it comes to efficient data sharing and retrieval. These systems typically require users to download, decrypt, search,

re-encrypt, and re-upload data, which introduces unnecessary computational overhead and delays [1], [3].

Searchable encryption (SE) was proposed to allow keyword-based search over encrypted data without decryption [2], [6]. While this approach improves usability, most SE schemes do not provide adequate access control. For example, anyone with a search token can retrieve the data, which is not ideal in a multi-user environment. Researchers like Curtmola et al. and Boneh et al. introduced early SE models, such as symmetric SE and public key encryption with keyword search (PEKS) [2], [7], but these models still lacked support for fine-grained user authorization. To introduce more refined access control, Attribute-Based Encryption (ABE) was developed. ABE enables encryption based on user attributes rather than identities, allowing data owners to define flexible access policies. Key developments in this area include Key-Policy ABE (KP-ABE) proposed by Goyal et al. and Ciphertext-Policy ABE (CP-ABE) by Bethencourt et al. These models provided the foundation for role-based access but still lacked efficient support for data search and retrieval [4], [5].

3. REQUIREMENT ANALYSIS

Functional Requirements:

These define the core operations that the system must perform to ensure secure cloud storage, sharing, and retrieval using Attribute-Based Encryption (ABE). They include encryption, access control, searchable encryption, and secure data retrieval. Secure user registration & authentication.

- Efficient encryption & decryption for secure data storage.
- Role-based access control using attribute-based encryption.
- Index-based search for encrypted data retrieval
- Cloud service provider (CSP) executes encrypted queries securely
- Returns k-most relevant results while preserving privacy.

3.2 Non-Functional Requirements:

Secure cloud storage using Attribute-Based Encryption (ABE) offers a robust solution for protecting sensitive data in distributed environments, and its effectiveness greatly depends not only on core functionality but also on well-defined non-functional requirements. These requirements play a critical role in ensuring that the system operates securely, efficiently, and reliably under real-world conditions. They encompass various essential aspects such as performance, which ensures that encryption, decryption, and search operations are completed within acceptable time frames even as data volume and user numbers grow; data privacy, which guarantees that unauthorized entities, including the Cloud Service Provider (CSP), cannot access plaintext data or infer sensitive metadata; and scalability, which allows the system to accommodate increasing loads, users, and data sizes without degradation in service quality. Additionally, non-functional requirements emphasize usability and responsiveness, making the system intuitive and practical for end users who must access, search, and share encrypted files without facing complexity.

Software Requirements:

- Programming Language / Platform : Python
- IDE : PyCharm
- Web Framework : Django
- Database : SQLITE
- Cloud Server : DriveHQ
- Front End : HTML ,CSS ,JS
- Testing Framework : Pyunit

3.3.2 Hardware Requirements:

- RAM : 4GB and Higher
- Hard Disk : 20 GB or More
- Operating System : Windows 10, macOS, or Linux

4. DESIGN

System Architecture:

The system architecture of the proposed FABECS (Fully Attribute-Based Encryption Scheme for Cloud Storage) outlines the interaction between various components such as the data owner, data user, cloud service provider, and the encryption modules. The data owner is responsible for encrypting the files using Attribute-Based Encryption (ABE) before uploading them to the cloud [4]. Along with encryption, the data owner defines access policies based on user attributes and creates searchable keyword indexes for efficient retrieval [6].

The data user can register, authenticate, and then perform encrypted keyword-based searches on the cloud-stored data. If the required data is found, the user sends a key request to the data owner. Upon verifying that the user's attributes match the defined access policies, the data owner grants the decryption key [7]. The user can then download and decrypt the file. The cloud service provider acts only as a storage medium and executes encrypted searches but cannot view or manipulate the data contents [8]. This architecture ensures a secure, efficient, and privacy-preserving mechanism for data sharing and retrieval in a multi-user cloud environment.

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 13, Issue 3, 2025

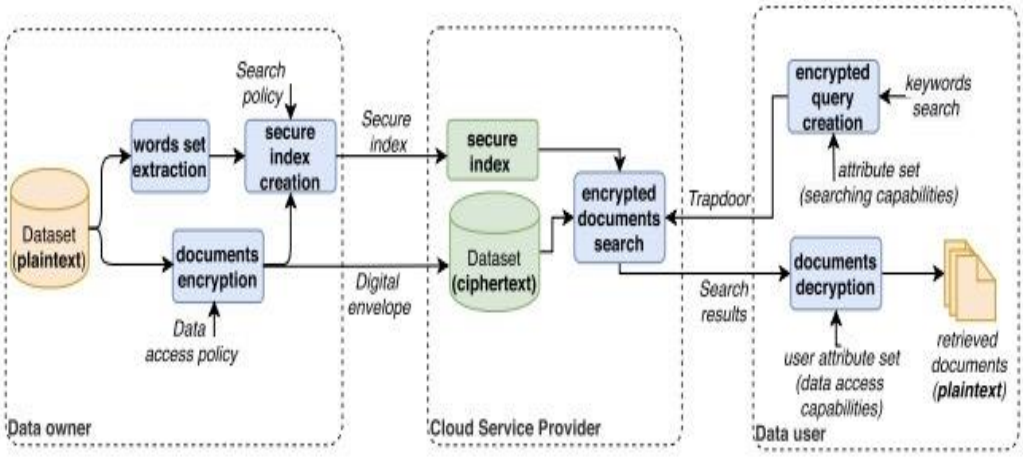


Fig. 4.1.1.1 System Architecture

Technical Architecture:

The technical architecture describes the technology stack and environment used to build and deploy the FABECS system. The frontend of the system is developed using HTML, CSS, and JavaScript, providing a clean and user-friendly interface for both data owners and users. Users can easily navigate through options such as registration, login, file upload, search, key requests, and downloads.

The backend is implemented in Python using the Django framework, which handles all co re operations including user authentication, file encryption/decryption, access control, and secure communication with the cloud. The system uses SQLite as the database to store user information, file metadata, access logs, and keyword indexes. For cloud storage, services like DriveHQ are used to securely host the encrypted files and indexes.

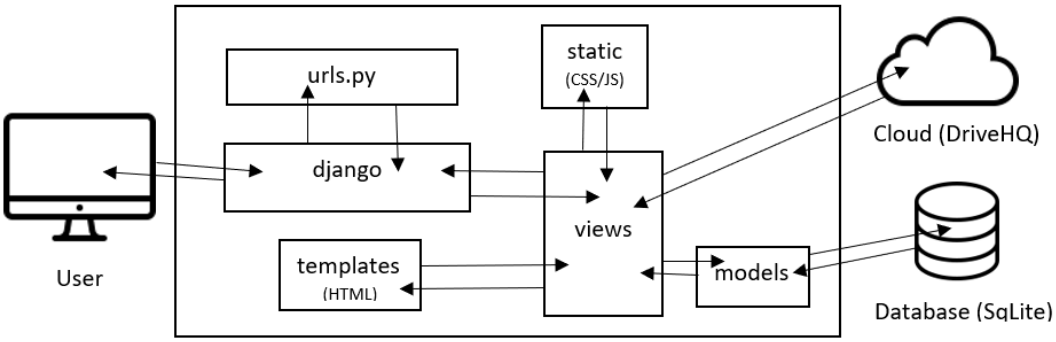


Fig. 4.1.2.1 Technical Architecture

Algorithm:

The algorithm used in your project is Ciphertext-Policy Attribute-Based Encryption (CP-ABE), which allows data access based on user attributes instead of specific identities [9], [10]. In a hospital management context, this means that access to sensitive patient data, medical reports, or internal documents is granted only to users whose attributes match certain predefined policies. For example, a medical report might be encrypted with a policy like “Role: Doctor AND Department: Cardiology,” ensuring that only a cardiologist has permission to view that file [9].

In this system, a doctor (data owner) uploads encrypted patient records to the cloud. Before uploading, the doctor defines an access policy and uses the CP-ABE algorithm to encrypt the file accordingly. Simultaneously, they also encrypt keywords like "ECG report" or "heart patient" and generate encrypted indexes using searchable encryption [2], [6]. This allows for secure, privacy-preserving searches on encrypted data.

Later, a data user, such as a nurse from the cardiology department, logs into the system and searches for relevant records using keywords. The system encrypts the nurse’s search query and sends it to the cloud service provider, which performs the

search using the encrypted indexes and returns matching encrypted files [2], [6], [8]. Importantly, the cloud never learns the actual data or the keyword, keeping everything fully private and secure [6], [8].

Once the nurse identifies a needed file, she sends a key request to the doctor (data owner). The system checks whether the nurse's attributes (e.g., "Role: Nurse", "Department: Cardiology") satisfy the policy under which the file was encrypted. If the nurse's attributes match, she receives the decryption key and gains access to the file. If not, access is denied, even if the file was found in the search [9].

This approach ensures fine-grained access control, meaning that sensitive hospital data is only accessible to appropriate medical staff. A receptionist or a lab technician, for example, would not be able to view a cardiologist's reports, even if they are part of the hospital network. Moreover, since users can search encrypted data without decrypting everything, the system remains efficient and scalable, suitable for large healthcare databases [8].

5. IMPLEMENTATION

5.1.1 asgiref==3.8.1:
ASGI (Asynchronous Server Gateway Interface) reference implementation used primarily in Django to support asynchronous programming. It helps bridge synchronous and asynchronous code, enabling modern features like WebSockets and async views. This library is lightweight yet essential for Django apps requiring concurrency support.

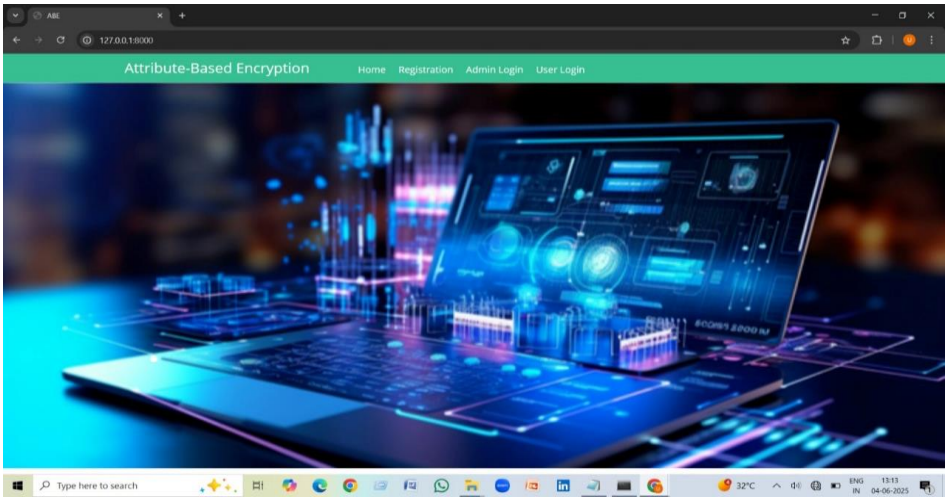
5.1.2 Cffi:
It stands for C Foreign Function Interface and allows Python code to interact with C code efficiently. It's widely used by libraries like cryptography to achieve high performance while maintaining safety and portability. It is essential for low-level operations and interfacing with C libraries from Python.

5.1.3 cryptography :
It is a robust library that provides cryptographic recipes and primitives such as encryption, decryption, digital signatures, and secure key management. Built on top of OpenSSL, it is widely trusted for implementing security features in Python applications. It's essential for HTTPS, JWTs, and password protection.

5.1.4 Django:
It is a high-level web framework that encourages rapid development and clean, pragmatic design. It includes an ORM, admin panel, authentication system, and built-in security features. Based on the MTV (Model-Template-View) pattern, Django is a popular choice for building secure and scalable web applications.

5.1.5 nltk (Natural Language Toolkit):
It is a powerful library for natural language processing and computational linguistics. It provides tools for tasks like tokenization, stemming, tagging, parsing, and access to large corpora. It's commonly used in academic and research settings for exploring text data.

6. SCREENSHOTS

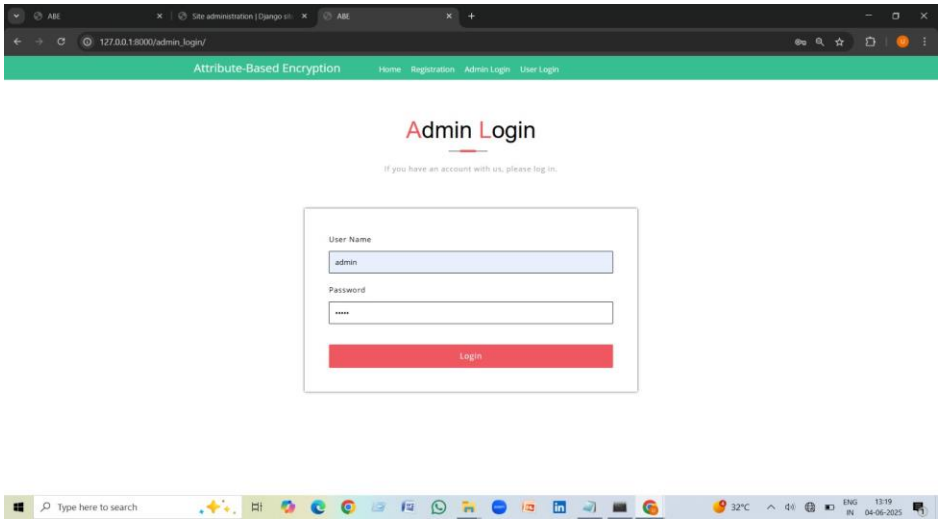


Screenshot 6.1 Home Page

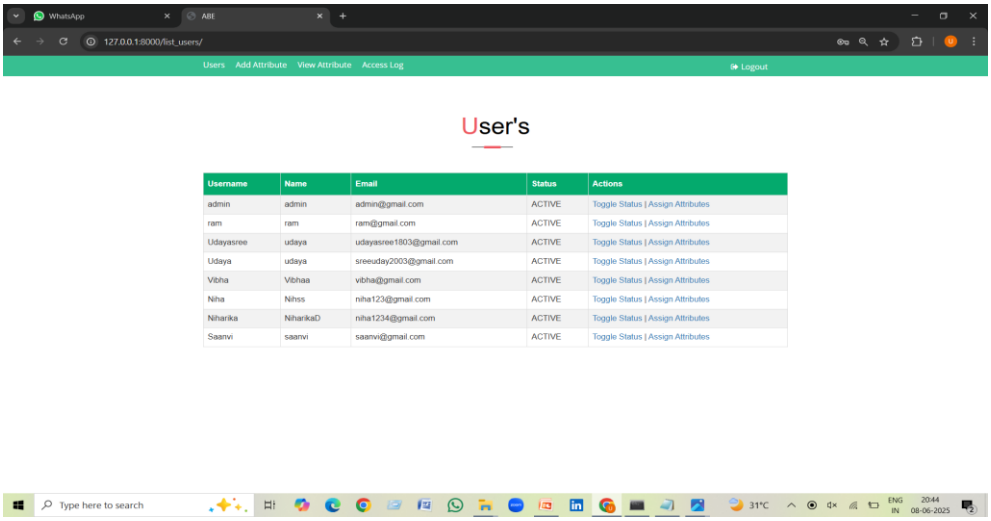
International Journal of Information Technology & Computer Engineering

ISSN 2347–3657

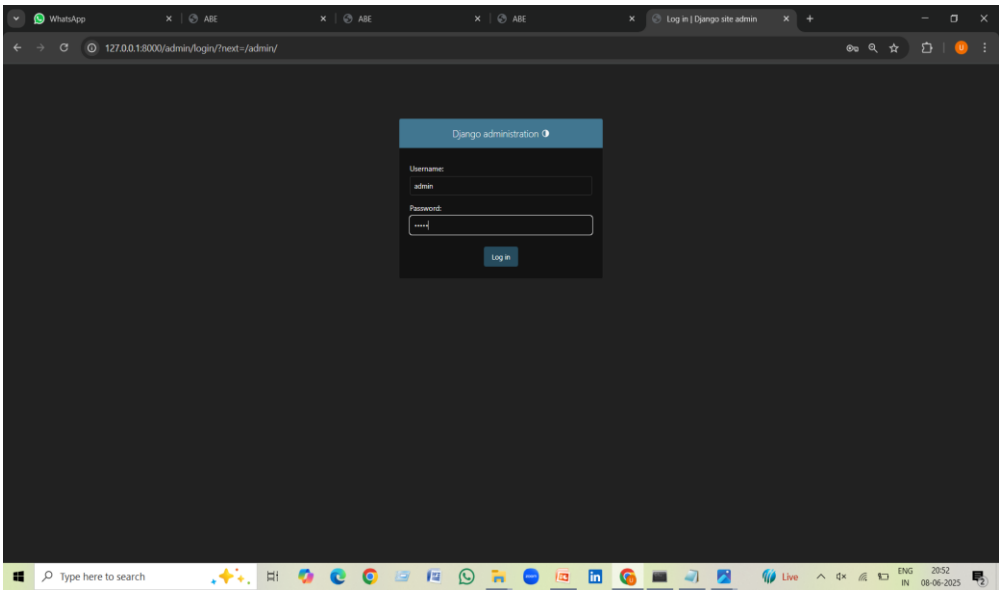
Volume 13, Issue 3, 2025



Screenshot 6.2 Admin Page

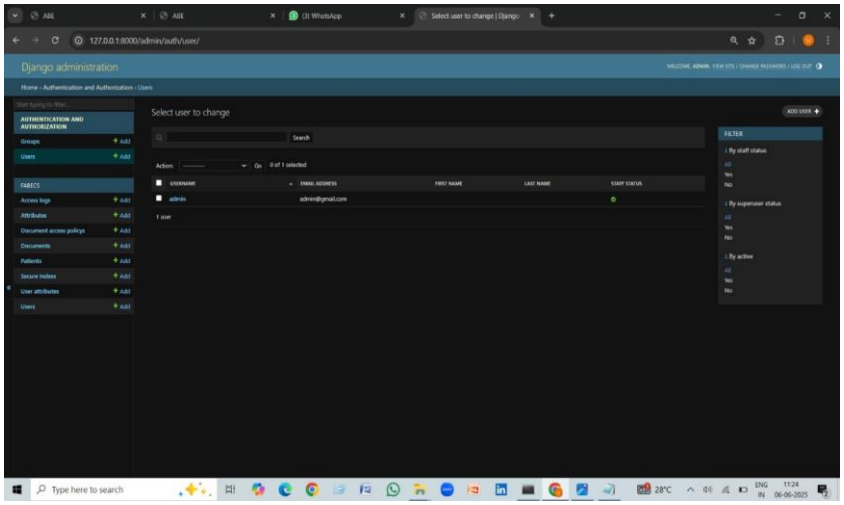


Screenshot 6.3 Admin Functionalities

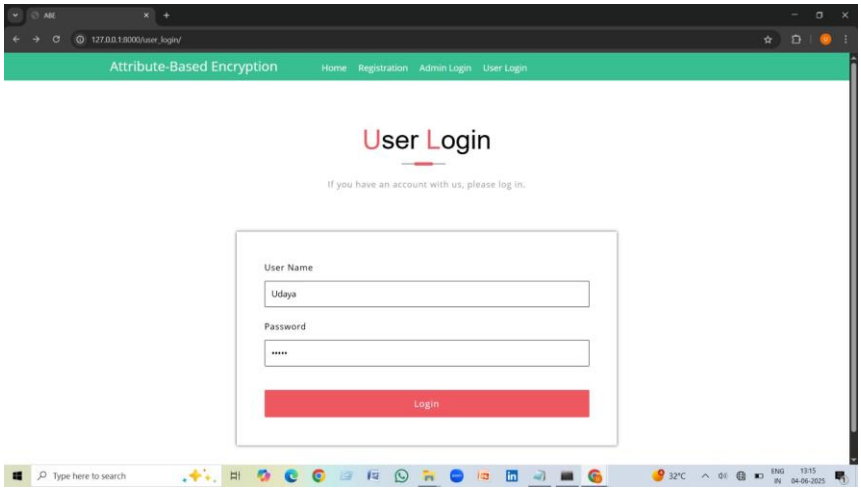


Screenshot 6.4 Django login

International Journal of Information Technology & Computer Engineering
ISSN 2347–3657
Volume 13, Issue 3, 2025



Screenshot 6.5 Django Administration



Screenshot 6.6 User Login Page

7-CONCLUSION

In this project, we have designed and implemented FABECS — a Flexible Attribute-Based Encryption and Cloud Storage system aimed at providing a highly secure and privacy-preserving framework for storing, sharing, and retrieving sensitive data in the cloud. FABECS addresses some of the critical challenges faced in cloud-based environments, such as data confidentiality, unauthorized access, and the inability to perform secure searches over encrypted data. By incorporating Attribute-Based Encryption (ABE), the framework enforces fine-grained access control, allowing only authorized users — based on defined attributes or policies — to decrypt and access the data.

One of the key features of FABECS is its support for secure, keyword-based searchable encryption. This allows users to search for specific content within encrypted documents without revealing either the search keywords or the actual document contents to the cloud service provider. To facilitate this, an **index-based retrieval mechanism** is integrated, ensuring that the system remains efficient and scalable even with large volumes of encrypted data.

REFERENCES

[1] A. Bagherzandi, B. Hore, and S. Mehrotra, Search over Encrypted Data. Boston, MA, USA: Springer, 2011, pp. 1088–1093.

[2] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, ‘‘Searchable symmetric encryption: Improved definitions and efficient constructions,’’ in Proc. 13th ACM Conf. Comput. Commun. Secur., New York, NY, USA, 2011, pp. 79–88, 2006.

[3] W. Song, B. Wang, Q. Wang, Z. Peng, W. Lou, and Y. Cui, ‘‘A privacypreserved full-text retrieval algorithm over encrypted data for cloud storage applications,’’ J. Parallel Distrib. Comput., vol. 99, pp. 14–27, Jan. 2017.

[4] H. Yin, J. Zhang, Y. Xiong, L. Ou, F. Li, S. Liao, and K. Li, ‘‘CP-ABSE: A ciphertext-policy attribute-based searchable encryption scheme,’’ IEEE Access, vol. 7, pp. 5682–5694, 2019.

[5] A. Sahai and B. Waters, ‘‘Fuzzy identity-based encryption,’’ in Advances in Cryptology (Lecture Notes in Computer Science), vol. 3494, R. Cramer, Ed. Berlin, Germany: Springer-Verlag, 2005, pp. 457–473.

[6] H. Pham, J. Woodworth, and M. A. Salehi, ‘‘Survey on secure search over encrypted data on the cloud,’’ Concurrency Comput. Pract. Exper., vol. 31, p. 1– 15, Apr. 2019.

[7] M. Zeng, H.-F. Qian, J. Chen, and K. Zhang, ‘‘Forward secure public key encryption with keyword search for outsourced cloud storage,’’ IEEE Trans. Cloud Comput., early access, Sep. 27, 2019, doi: 10.1109/TCC.2019.2944367.

[8] A. G. Kumbhare, Y. Simmhan, and V. Prasanna, ‘‘Designing a secure storage repository for sharing scientific datasets using public clouds,’’ in Proc. 2nd Int. workshop Data Intensive Comput. Clouds, 2011, pp. 31–40.

[9] Z. Yang, J. Tang, and H. Liu, ‘‘Cloud information retrieval: Model description and scheme design,’’ IEEE Access, vol. 6, pp. 15420–15430, 2018.

[10] S. Kamara, C. Papamanthou, and T. Roeder, ‘‘Cs2: A searchable cryptographic cloud storage system,’’ Microsoft Res., Redmond, WA, USA, Tech. Rep. MSR-TR-2011-58, May 2011.