INTERNATIONAL JOURNAL OF NOVEL RESEARCH AND DEVELOPMENT (IJNRD)
© 2026 IJNRD | Volume 11, Issue 5, May 2026 | ISSN: 2456-4184 | IJNRD.ORG

A Dynamic Multilinear Map-Based Proxy Re-Encryption Scheme for Secure Cloud Data Sharing

[1]Rajdeep Bhavanarushi, [2]Sudarshan Reddy, [3]Mrs. A.V.L. Prasuna

[1]B.Tech Student (Information Technology), [2]B.Tech Student (Information Technology), [3]Assistant Professor, Dept (Information Technology)
[1]Department of Information Technology,
[1]Mahatma Gandhi Institute of Technology, Hyderabad, India

Abstract: *While cloud computing has changed the way people store and share data in the cloud, there is a risk of unauthorized access or a lack of user control over their sensitive data. Many current encryption schemes do not take into account the dynamics of cloud computing when providing means to share data securely among many users by granting access rights through delegation to a semi-trusted cloud server. This paper presents a dynamic PRE scheme using the concept of multilinear maps to achieve more efficient and scalable data sharing among different data classes in cloud computing by incorporating KAC scheme. A data owner can generate an aggregate key for different data classes, delegate re-encryption authority to the cloud server and also has the option of revoking it later according to his requirements. The system incorporates verification schemes for the re-encryption process. This software implementation is that of a web-based application called Dynamic KAPRE Scheme Using Multilinear Maps, which is hosted on Apache Tomcat and uses a MySQL database. Tests conducted in real-time indicated that the system successfully implements access control mechanisms and encrypts files before sending them through e-mail and even dynamically revoking and editing these files.*

IndexTerms - Proxy Re-Encryption, Key-Aggregate Cryptosystem, Multilinear Maps, Cloud Security, Data Sharing, Revocation

I. INTRODUCTION

Storing information via cloud services has become a common practice for individuals as well as enterprises. Various services include Google Drive, Dropbox, and other cloud servers designed by different enterprises which help users to store huge chunks of data in remote locations from where the same can be accessed by other individuals worldwide. Yet there is always some sacrifice involved in doing so because once the user sends his/her data to the third party's server, he/she no longer remains in complete control over its accessibility. The conventional techniques used for secure data transmission to the cloud employ symmetric or asymmetric encryption, wherein the owner encrypts the data prior to transferring it to the cloud. Though this technique offers security from any form of breach, the main issue arises when there is a need to transfer the data to several other users. Encrypting the same data for every user would be an impractical exercise. There is no way one can keep re-encrypting the whole data set whenever there is any change in access privileges.

To address the problem mentioned above, Proxy Re-Encryption (PRE) was proposed. In proxy re-encryption, a semi-trusted proxy is able to convert an encryption under one key to a different key while never revealing the underlying information. Through a proxy re-encryption key issued to the cloud, the data owner can effectively delegate access to a third party. The problem with most existing PRE systems is that they rely on fixed access policies that are not modifiable afterward. Key-Aggregate Cryptosystem (KAC) is a cryptosystem where an authorized entity can create a single, smaller key that combines the decryption capabilities for a number of different classes of data. Rather than distributing a separate key for each data file, it is possible to distribute a smaller aggregate key to decrypt a number of files. However, KAC cannot dynamically support revocation of users. Multilinear maps involve more sophisticated relations between variables by generalizing bilinear pairings for the creation of expressive policies. The proposal presented in the paper involves the combination of the three techniques of PRE, KAC, and multilinear maps to present an efficient and working scheme named "Dynamic KAPRE Scheme Using Multilinear Maps."

II. RELATED WORK

Proxy re-encryption was proposed by Ateniese et al. [1], who also developed a unidirectional PRE protocol using bilinear pairings that proved the delegation of decryption keys by semi-trusted proxies without compromising the confidentiality of messages. This protocol which was used in the process lacked a condition mechanism, thus failing to apply in scenarios where access depended on few important parameters like time or user roles.

When the Condition element was added by Tang [2] in the re-encryption key marked a major breakthrough in the development of PRE protocols. However, the condition remained constant throughout the system lifecycle and could not be updated, which greatly limited its usage

KAC, Key-Aggregate Cryptosystem,was first given by Boneh et al. [3], where an authorized user can give one key for several categories of data. The disadvantage here is that KAC cannot handle proxy re-encryption. After the key is given, the key revoking option is gone, since there is no way to update all the ciphertexts generated using this key.

Multilinear maps were first proposed by Garg et al. [4]. This construction can express more complex conditions compared to bilinear pairings. But since computational costs are very high, so it makes it difficult to implement these multilinear maps.

An algorithm which is a revocable proxy re-encryption using a trusted authority was proposed by Liang et al. [5]. Even though this approach was good, it was dependent on central authority which made it to fail due to a single point of failure issue, because of this it is not suitable for distributed cloud computing environments.

Attribute-based encryption and proxy re-encryption were combined by Shao et al. [6] to provide fine-grained access control support. However, attribute-based encryption algorithms involve higher costs and large keys, making them inefficient for large-scale deployment. The current literature does not contain any previous solution that considers dynamic attributes, small-sized aggregate keys, efficient revocation mechanism, and ciphertext authentication altogether.

III. METHODOLOGY

The methodology section outlines the plan and method that how the study is conducted. The proposed scheme is built around five main components: the Data Owner, the Key Management Module, the Cloud/Proxy, the Revocation and Verification Module, and the Users. The Data Owner acts as the root authority who owns the master secret key and is responsible for the encryption process. He generates the re-encryption keys, which enable the proxy to perform transformations on behalf of individual users. The Key Management Module handles all tasks related to managing keys. These include key generation, key aggregation for multiple data classes, and key updates/re-keying when access permissions change.

The Cloud/Proxy holds the encrypted messages and executes the process of re-encryption when asked. It is assumed that the proxy is semi-trusted; it obeys all the rules of the protocol but might try to figure out something about the message itself. The Revocation and Verification Module keep track of the revoked users and verifies the re-encrypted ciphertexts. Finally, the users become the ultimate receivers who decrypt the ciphertexts using their own secret keys along with the aggregate key.

3.1 Setup

In the initialization phase, the bilinear group G of prime order q is chosen together with its generator g. Multilinear maps e: G^k → G_T are created, where k stands for the maximum possible depth. Master secret key alpha is generated randomly in Z_q, and master public key is obtained by PK = g^alpha. Additionally, the cryptographic hash function H: {0,1}* → G should be chosen.

3.2 Key Generation and Encryption

The aggregate secret key for the data class set S is defined as:SK_agg = $\prod$ g^(α · H(i)) for all i ∈ S
For encryption of message m based on condition cond, the data owner generates random number r from Z_q and computes:
C1 = g^r, C2 = m · e(PK, CondEnc)^r Where CondEnc represents the encoding of the condition cond.

3.3 Re-encryption and Decryption

For granting user U access rights, the data owner calculates re-encryption key RK as:
RK = g^(α · t) · CondEnc where t is a token based on the time factor and representing the state of conditions at that point of time. The cloud proxy uses RK to re-encrypt the original ciphertext C
= (C1, C2), producing ciphertext C' that can be decrypted by the authorized user U using formula: *m = C2 / e(SK_agg, C1)*
Decryption process involves bilinear map to strip off the encryption mask. Decryption is possible only when conditions are met and SK_agg corresponds to a particular class of data; otherwise, no information is retrieved.

IV. SYSTEM ARCHITECTURE

The entire framework of the architecture works in layers where each layer is dedicated to executing a different task in the process of data sharing. In the data source, the Data Owner ensures that files are encrypted during upload, assigning each ciphertext to particular data classes and associated access conditions. In the Key Management layer, aggregate keys for several data classes and re-encryption keys for use by the cloud proxy are computed. On the cloud side, the proxy determines if there are any valid re-encryption keys and if access conditions are met in order to perform the encryption of the file. The Verification and Revocation layer ensures that the integrity of the system remains intact and any update of keys due to changes in access conditions occur without any need for re-encryption of the data.

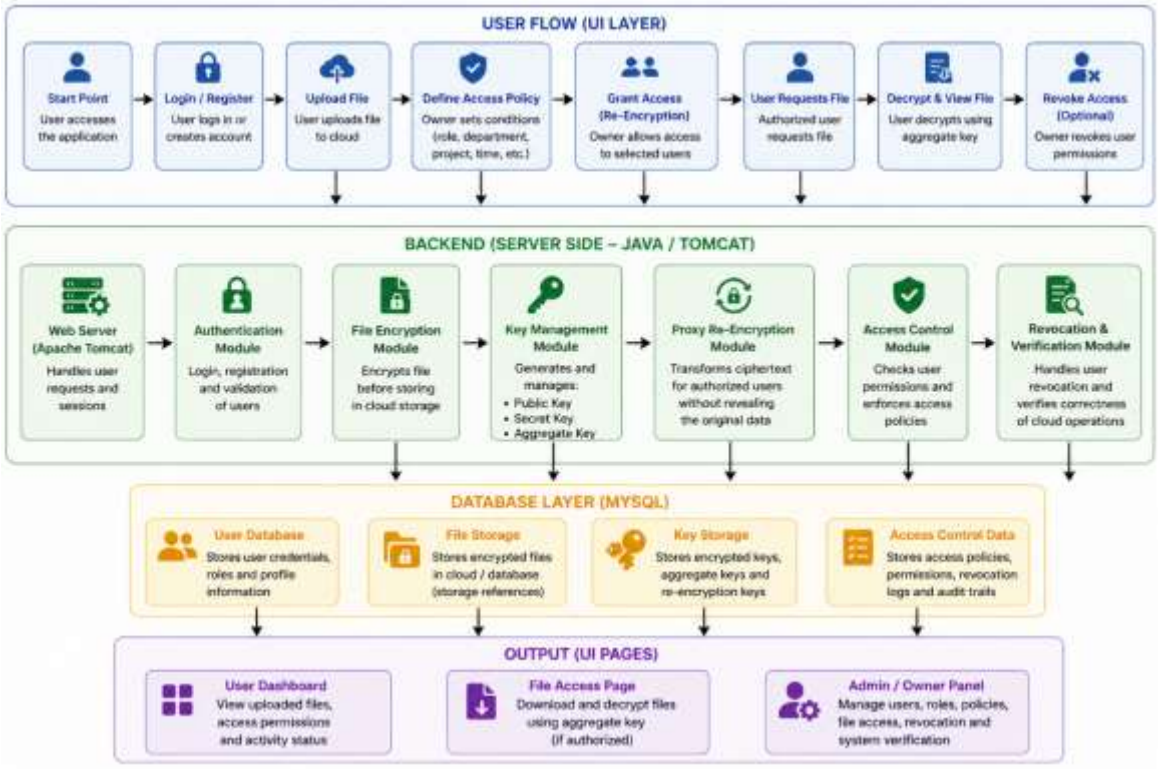

Fig. 1: System Architecture of Dynamic KAPRE Scheme

V. ALGORITHMIC WORKFLOW

The following steps make up the whole process flow for the suggested algorithm:

5.1 Setup

Sets up a bilinear group G of order q, picks a generator g, finds a multilinear map e, gets a master secret key alpha, finds PK=g^alpha, and sets up hash function H.

5.2 Encrypt

Pick a random number r in Z_q and use it to find C1=g^r and C2=m·e(PK, CondEnc)^r for file F of data class i under condition cond. Then, the ciphertext is C=(C1, C2, i, cond).

5.3 Upload

The ciphertext C is sent to the cloud server along with other information, like metadata about the data's class and access policy details.

5.4 Generate SK_agg

For the set of data class indices S, SK_agg=Π_{i ∈ S}g^(α · H(i)) will be calculated and sent to the recipient in a secure way.

5.5 Make RK

RK=g^(α · t)·CondEnc will be sent to the cloud proxy server, which grants user U the permission to re-encrypt.

5.6 Re-Encrypt

The cloud proxy checks the condition and re-encrypts C to C' with the aid of RK so that user U can decrypt it.

5.7 Decrypt

If all the conditions are met, the user the original message by computing m = C2/e(SK_agg, C1).

5.8 Revoke and Verify

If you revoke access, you re-encryption keys to all the users who do not revoke

VI. RESULTS

6.1 Summary of Functional Test Results

Module	Feature Tested	Result
Registration	Data Owner and Regular User signup	Pass
File Upload	Encrypted file storage with metadata	Pass
Key Generation	Aggregate key generation per file	Pass
Metadata Search	Keyword-based file discovery	Pass
Access Request	User request submission and approval	Pass
Key Delivery	Automated email with secret key	Pass
Decryption	File access using key verification	Pass
Edit Permission	Collaborative content update	Pass
Database Integrity	All records correctly stored in MySQL	Pass
Revocation	Re-encryption key update on denial	Pass

Table 1: Summary of Functional Test Results

6.2 Output Screenshots and System Execution

The below following shows the results of the project execution:

Fig. 2: Home page of the deployed Dynamic KAPRE Scheme application.

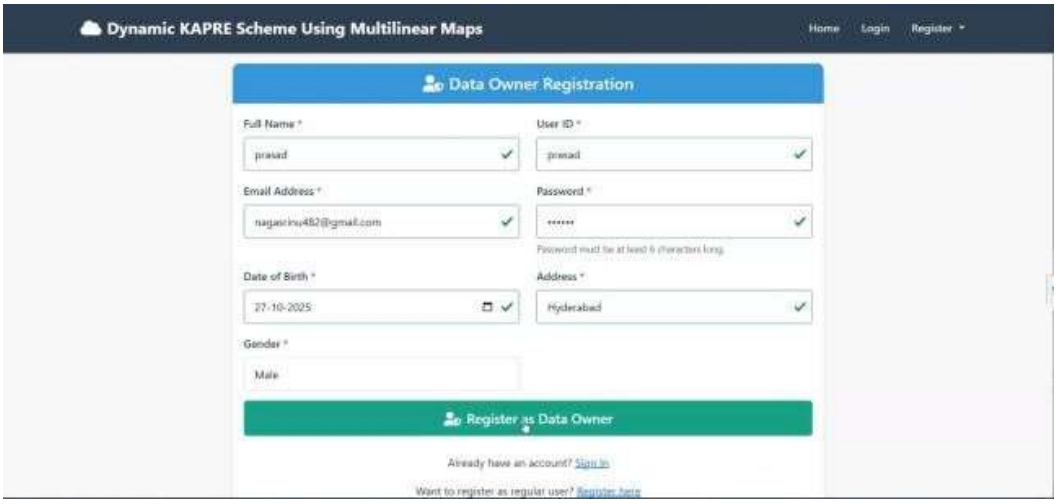

Fig. 3: User registration form for any user

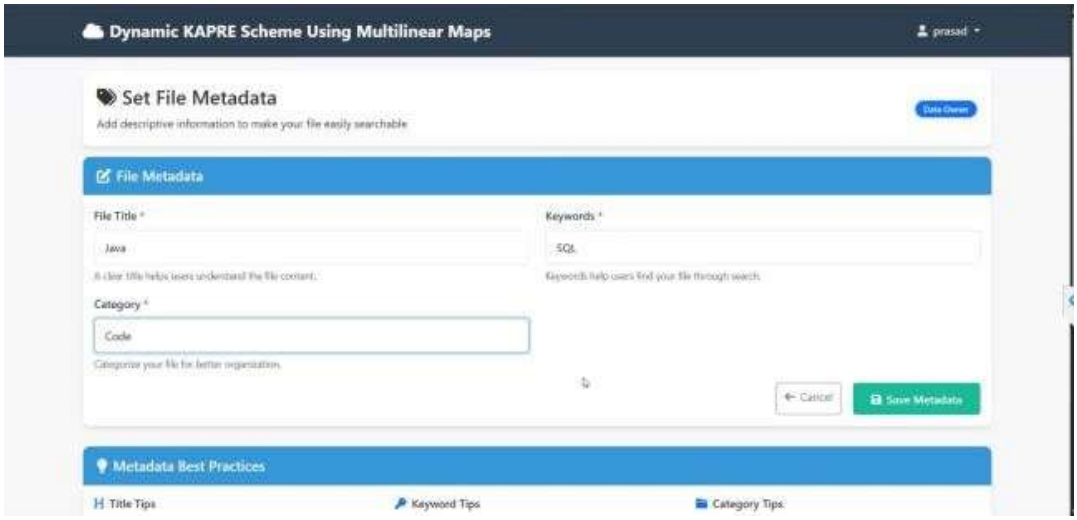

Fig. 4: Set File Metadata screen with the title, keyword and category fields.

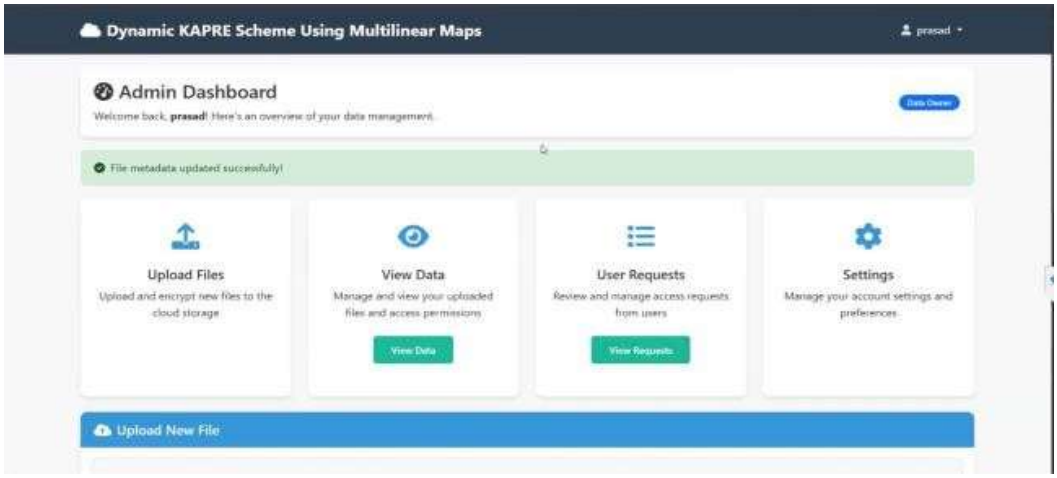

Fig. 5: Admin Dashboard with the modules that the data "prasad" can access.

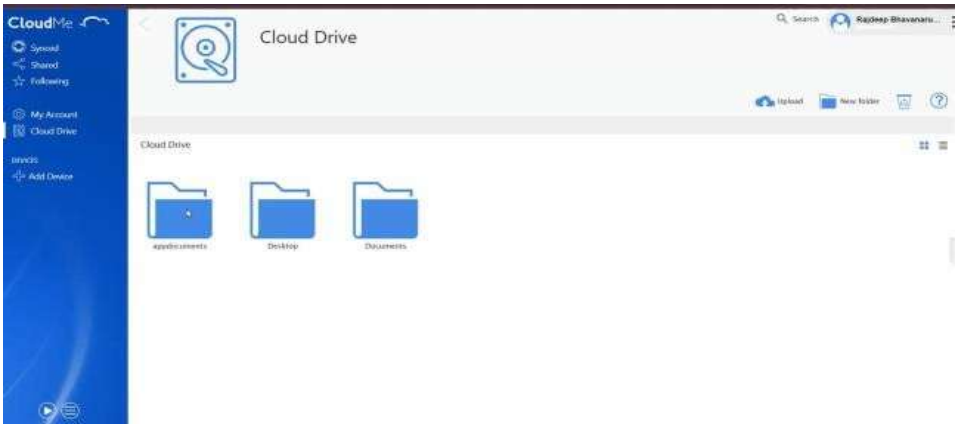

Fig. 6: CloudMe integration showing synced folders for the the data owner's account.

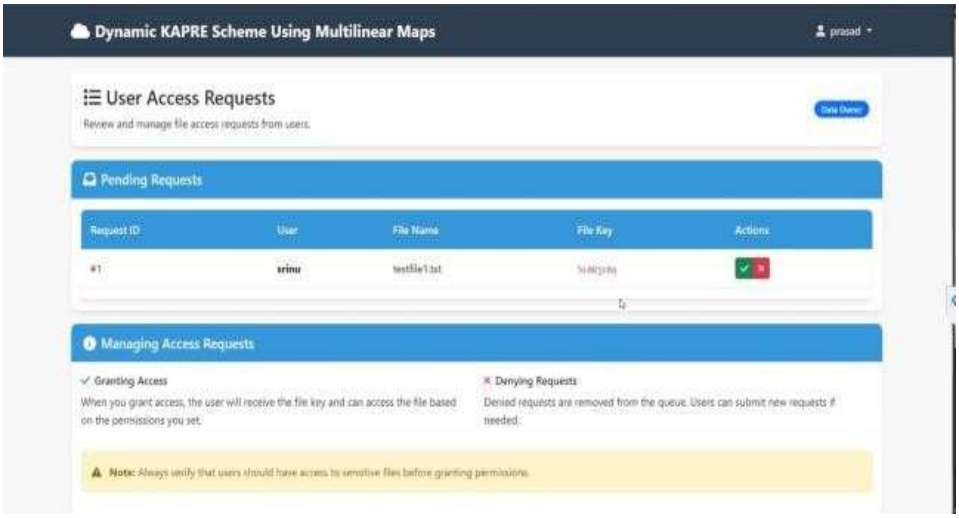

Fig. 7: Requested access to testfile1.txt from user showing the file key

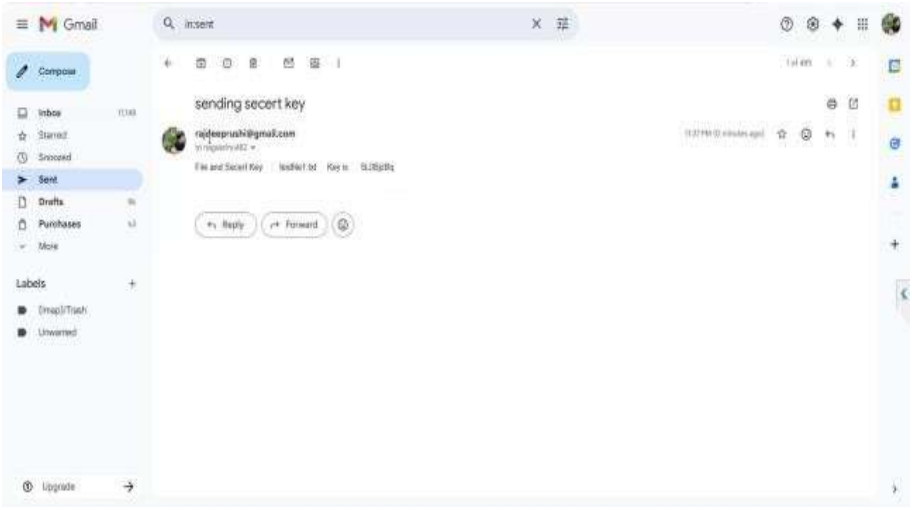

Fig. 8: Gmail sent folder showing automated secret key email to the user

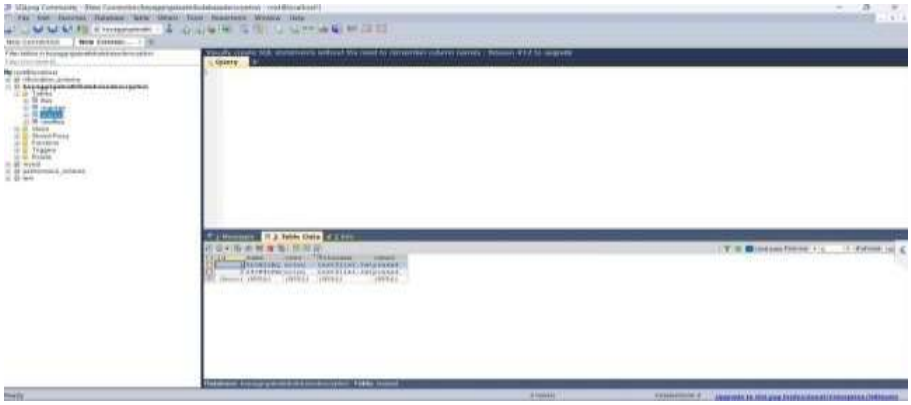

Fig. 9: SQLYog(DataBase) request table with two access requests for user 'srinu'

VII. CONCLUSION

This paper introduced a dynamic multilinear map-based proxy re-encryption scheme that incorporates the Key-Aggregate Cryptosystem for secure and adaptable cloud data sharing. The proposed system was made into a working web app and tested on all of its main functional modules, including file encryption, metadata-based search, granular access permission management, automated key delivery via email, collaborative file editing within permission bounds, and lightweight access revocation.

The results of the implementation showed that all parts of the system work as they should. Keys for encrypted files are kept separate from the files themselves. Requests for access are handled through an approval workflow, and keys are sent through an email channel that is not part of the main email system. Database records confirmed that encryption, key generation, and permission updates are all stored correctly.In the future, we will replace the email key delivery channel with a secure encrypted protocol, add blockchain for decentralized key management and revocation logging, look into more robust multilinear map constructions as they become more common in the research community, and make the system work for group-based sharing and hierarchical permissions in business settings.

VIII. REFERENCES

[1] Y. Zhang, X. Liu, and J. Chen, "Efficient and secure data sharing scheme based on proxy re-encryption in cloud computing," IEEE Trans. Cloud Comput., vol. 13, no. 1, pp. 112–125, 2025.
[2] H. Li, Z. Qin, and K. Ren, "Secure and flexible data sharing with proxy re-encryption in cloud environments," IEEE Trans. Inf. Forensics Secur., vol. 19, pp. 210–223, 2024.
[3] J. Wang, Y. Wu, and S. Yu, "Dynamic access control and revocation in cloud storage using key-aggregate encryption," Future Gener. Comput. Syst., vol. 140, pp. 45–56, 2023.
[4] X. Liu, Y. Zhang, and B. Wang, "Secure data sharing scheme based on proxy re-encryption for cloud storage," Future Gener. Comput. Syst., vol. 108, pp. 719–729, 2022.
[5] H. Wang, Z. Qin, and J. Domingo-Ferrer, "Privacy-preserving cloud storage framework with dynamic access control and efficient revocation," IEEE Trans. Knowl. Data Eng., vol. 32, no. 6, pp. 1175–1188, 2022.
[6] Y. Ren, J. Shen, J. Wang, J. Han, and S. Lee, "Mutual verifiable provable data auditing in public cloud storage," IEEE Trans. Serv. Comput., vol. 14, no. 4, pp. 1107–1120, 2021.
[7] C. K. Chu, S. S. M. Chow, W. G. Tzeng, J. Zhou, and R. H. Deng, "Key-aggregate cryptosystem for scalable data sharing in cloud storage," IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 2, pp. 468–477, 2014.
[8] J. Chen, H. Ma, and Y. Li, "Efficient proxy re-encryption with dynamic access control for secure cloud storage," IEEE Access, vol. 7, pp. 112543–112554, 2019.
[9] X. Wu, Y. Zhang, and J. Shen, "Secure data sharing scheme based on key-aggregate encryption in cloud computing," Future Gener. Comput. Syst., vol. 95, pp. 608–615, 2019.
[10] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou, "Scalable and secure sharing of personal health records in cloud computing using attribute-based encryption," IEEE Trans. Parallel Distrib. Syst., vol. 28, no. 5, pp. 1311–1324, 2017