Privacy-Preserving Cloud Data Security: Integrating the Novel Opacus Encryption and Blockchain Key Management

S. Poorani[1*], and R. Anitha[1]
[1]Department of Computer Science and Engineering, Sri Venkateswara College of Engineering,
Sriperumbudur, Tamilnadu, 602117, India
[e-mail: spoorani@svce.ac.in, ranitha@svce.ac.in]
[*]Corresponding author: S. Poorani

Received July 18, 2023; revised September 26, 2023; accepted October 24, 2023;
published November 30, 2023

Abstract

With the growing adoption of cloud-based technologies, maintaining the privacy and security of cloud data has become a pressing issue. Privacy-preserving encryption schemes are a promising approach for achieving cloud data security, but they require careful design and implementation to be effective. The integrated approach to cloud data security that we suggest in this work uses CogniGate: the orchestrated permissions protocol, index trees, blockchain key management, and unique Opacus encryption. Opacus encryption is a novel homomorphic encryption scheme that enables computation on encrypted data, making it a powerful tool for cloud data security. CogniGate Protocol enables more flexibility and control over access to cloud data by allowing for fine-grained limitations on access depending on user parameters. Index trees provide an efficient data structure for storing and retrieving encrypted data, while blockchain key management ensures the secure and decentralized storage of encryption keys. Performance evaluation focuses on key aspects, including computation cost for the data owner, computation cost for data sharers, the average time cost of index construction, query consumption for data providers, and time cost in key generation. The results highlight that the integrated approach safeguards cloud data while preserving privacy, maintaining usability, and demonstrating high performance. In addition, we explore the role of differential privacy in our integrated approach, showing how it can be used to further enhance privacy protection without compromising performance. We also discuss the key management challenges associated with our approach and propose a novel blockchain-based key management system that leverages smart contracts and consensus mechanisms to ensure the secure and decentralized storage of encryption keys.

Keywords: Cloud, privacy and security, blockchain, homomorphic encryption, access control, key management.

http://doi.org/10.3837/tiis.2023.11.015 **ISSN : 1976-7277**

1. Introduction

The demand for safe and private cloud data management has expanded due to the quick development of cloud-based technologies [1]. As a result of the increase in cyber-attacks and data breaches, protecting the security and confidentiality of cloud data has become a top priority for both individuals and businesses. Researchers and industry experts have explored various approaches to achieving privacy-preserving cloud data security to address this challenge [2-4]. Cloud-based technologies have transformed how organizations and individuals store, process, and access data. By allowing remote access to computing resources and data storage, cloud computing has opened up new possibilities for collaboration, innovation, and scalability [5-7]. However, this convenience and flexibility also have inherent security and privacy risks. While there are many promising approaches to achieving privacy-preserving cloud data security, there are also several challenges that must be overcome to ensure the effectiveness and practicality of these approaches [8].

1.1 Key challenges associated with privacy-preserving cloud data security

Usability and Performance:

One of the main challenges of privacy-preserving cloud data security is balancing the need for safety with the usability and performance of cloud-based technologies [9]. Encryption and access control mechanisms sometimes result in increased processing time and decreased system performance, negatively impacting user experience and adoption. It is essential to strike a balance between the need for security and the need for usability and performance.

Scalability:

Another challenge of privacy-preserving cloud data security is scalability. As a growing amount of data is produced and kept on the cloud, there is a need for scalable solutions that can handle large volumes of data while maintaining privacy and security [10-12]. This requires efficient data structures and encryption schemes that can be scaled to meet the demands of cloud-based systems.

Key Management:

Effective key management is critical to privacy-preserving cloud data security. Encryption keys must be stored securely and managed effectively to guarantee the data's secrecy and integrity [13-15]. Centralized key management systems can be vulnerable to attacks and data breaches, so decentralized and blockchain-based key management systems are becoming increasingly popular.

Interoperability:

Interoperability is a challenge when it comes to privacy-preserving cloud data security. Cloud systems may use various encryption schemes or access control mechanisms, making integrating and managing data across different systems difficult. Interoperability standards and protocols can help to address this challenge, but more work is needed in this area.

Compliance:

Compliance with regulations and industry standards is critical in privacy-preserving cloud data security. Different industries and jurisdictions may have additional regulations and requirements for data privacy and security, making it difficult to implement a comprehensive approach to cloud data security [16]. Interoperability standards and protocols can help address this challenge, but more work is needed.

Complexity:

Finally, privacy-preserving cloud data security can take time and effort to implement. Encryption schemes, access control mechanisms, and key management systems all require careful design and implementation to be effective. The system requires specialized knowledge and expertise, which may only be available to some organizations. Simplifying and streamlining the implementation process can help to overcome this challenge.

1.2 Objectives

The proposed method for privacy-preserving cloud data security aims to protect cloud data's privacy and security while maintaining high levels of performance and usability.

- To enable secure computation on encrypted data while preserving privacy.
- Implement precise access controls depending on user attributes by utilizing CogniGate Protocol. This gives cloud data access more flexibility and control, guaranteeing those with permission have access to sensitive data.
- To store and retrieve encrypted data using an effective data structure. To guarantee that cloud data may be accessed swiftly and effectively without jeopardizing security or privacy.

Researchers and industry experts have been exploring various approaches to achieving privacy-preserving cloud data security to address this challenge. Encryption to safeguard confidential information while in transmission or at rest constitutes a promising strategy. Data is encoded using encryption, so it cannot be read without a decryption key. Data is made secure even if it is intercepted or viewed without permission by encrypting it while it is sent online or kept there. However, traditional encryption methods have limitations regarding cloud data security. For example, if encrypted data needs to be processed or analyzed in the cloud, it must first be decrypted, which exposes it to potential security risks. Furthermore, traditional encryption methods do not provide granular access control, which means that anyone with access to the decryption key can access all the data.

2. Related Work

Privacy-preserving cloud data security is a field of research that seeks to develop methods and techniques for ensuring cloud data security. As cloud-based technologies are popular, the need for effective security measures becomes increasingly important. Cloud computing allows individuals and organizations to store and process data on remote servers accessed via the Internet. While cloud computing provides many benefits, such as cost savings, scalability, and flexibility, it also poses significant data privacy and security risks. Cyberattacks, data breaches, and unauthorized access are just some of the potential threats that cloud users face. One approach to addressing these challenges is privacy-preserving encryption. Data is encrypted using this method before being delivered to the cloud, ensuring that sensitive data remains private and secure. However, traditional encryption methods have limitations that can compromise cloud data security.

An authentication along with an information security framework for the use of cloud computing was created by Pawar et al. [17]. Their strategy emphasizes ensuring safe data sharing and privacy preservation by using methods like hashing, 3DES, interpolation, and XOR. In contrast, the research suggested in this study offers an integrated approach for cloud data security that integrates Opacus encryption, CogniGate Protocol, index trees, and blockchain key management. It focuses on encryption that protects user privacy, granular access control, adequate data storage, and reliable key management employing blockchain technology. The proposed work also investigates differential privacy to improve protection without sacrificing performance, tackling significant issues, and utilizing upcoming technology. Kuldeep et al. [18] suggested a multi-class privacy-preserving cloud computing approach that uses confidentiality for data encryption and compressed detection for compact sensor data encoding. The plan proposes three iterations of the MPCC (Multi-Class Privacy-Preserving Cloud) architecture that are specially made for decrypting smart meter data statistically, deanonymizing image data, and decrypting electrocardiogram (ECG) signals. The MPCC concept addresses a number of significant problems with cloud computing, including those relating to massive IoT sensor data transfer, energy use, storage, and privacy issues with IoT data. The approach taken by Kuldeep et al. primarily emphasizes compressive sensing and secrecy techniques for specific applications, in contrast to the proposed work in this paper, which concentrates on a comprehensive strategy for cloud data security using Opacus encryption, CogniGate Protocol, index trees, and blockchain key management. Although both approaches strive to increase cloud computing privacy, there are differences in the precise methodologies used in the intended applications, especially the degrees of anonymity attained.

Shanmugapriya et al. [19] approach emphasizes using deep learning networks for common pattern mining. It proposes the SSO-PE strategy to protect data privacy as opposed to the work that is proposed in this study. They specifically address the computing complexity problem frequently present in pattern mining activities. The suggested work offers an integrated strategy integrating Opacus encryption, CogniGate Protocol, index trees, and blockchain key management. It focuses on encryption that protects user privacy, granular access control, adequate data storage, and efficient handling of keys leveraging blockchain technology. While both strategies strive to improve data privacy and security, they differ in the methodologies and the particular issues they intend to solve. A secure multi-authority access management method was presented by Gupta et al. [20]. Their method accommodates users across public and private domains despite protecting policy privacy by simply disclosing the names of policy characteristics while maintaining hidden values. In contrast to the work suggested in this research, Gupta et al.'s approach focuses on tackling access control issues in the healthcare industry, especially for online sharing of data scenarios. Their system prioritizes privacy by concealing sensitive attribute values and enabling fine-grained access restriction. Additionally, it supports the multi-authority environment, essential for controlling access in complicated healthcare situations.

Xiong et al. proposed a concealing gain access strategy property-based anonymized attribute-based broadcasting authentication method [21]. Their system enables a data owner to distribute data to several users who comply with their access regulation and constitute a component of a predetermined receiver set. The authors provided an actual random oracle-free anonymized attribute-based transmission method of encryption and thorough security proof. An innovative Internet of Things (IoT) system for healthcare was unveiled by Xu et al. [22], which brings together the benefits of attribute-based encryption, computing in the cloud, and edge computing. In medical IoT networks, their technology delivers an effective, adaptable, and secure fine-grained authentication approach combined with validation of data capabilities. Amiri et al. [23] unveiled an intelligent parking system that protects privacy by utilizing blockchain technology and retrieving personal data. In order to guarantee security, transparency, and the accessibility of parking offers, their strategy is the creation of a collective blockchain owned by several parking lot owners. The authors use a private data retrieval method that enables drivers to discreetly retrieve parking offers via nodes on the blockchain to secure drivers' physical privacy. Additionally, a brief randomizable signature is used, enabling drivers to covertly recognize themselves when booking open parking spaces from parking owners. A unique Multi-Source Order-Preserving Sync Encryption method was developed by Yao et al. [24], which allows the cloud to combine secret information queries across several data providers without knowing the contents of the indexes. Without being aware of the scope of the inquiry, the cloud analyzes the protected information across each of the pertinent suppliers, enabling quick and discreet query management. The authors also suggested an improved system that facilitates data queries more successfully by using an organizational framework amongst data suppliers.

A secure authorization and data-sharing mechanism for active user groups was proposed by Yang et al. [25]. Three essential components make up their strategy. To ensure that only people with permission to access a given data set may do so, they first design and implement access controls based on data properties. They also make it possible for the key generation centre to administer access control flexibly and dynamically by efficiently updating account information for dynamic user groups. The authors created an effective ciphertext delegation-based attribute-based encryption (ABE) technique to accomplish these goals. Zhang et al. [26] method focuses on merging blockchain technology with public key accessible encryption to accomplish data security and safe data sharing, in contrast to the technique suggested in this paper. To develop a control mechanism and guarantee impartiality in data transactions, they highlight the usage of digital contracts. To improve the accessibility and safety of safeguarding priceless and susceptible information, Zhang et al. [27] presented the CP-ABE encryption method. The authors developed a privacy-preserving CP-ABE system with reliable authority authentication to protect confidential and essential facts. Notably, their technique maintains a consistent size for the secret keys. Zhang et al.'s approach, in contrast to the work indicated in this research, emphasizes using CP-ABE to safeguard sensitive and essential information while upholding user privacy.

The proposed approach in this paper takes a comprehensive and integrated approach to cloud data security. It combines techniques such as Opacus encryption, CogniGate Protocol, index trees, and blockchain key management to address various aspects of data privacy, access control, data sharing, and secure key management. While the existing works address specific challenges such as authentication, multi-class privacy preservation, healthcare data sharing, frequent pattern mining, smart parking, and attribute-based encryption, the proposed approach provides a broader solution that encompasses these concerns and offers a more comprehensive framework for ensuring cloud data security, privacy preservation, fine-grained access control, efficient data storage, and secure key management. It leverages emerging technologies and explores additional aspects like differential privacy and scalable encryption schemes.

3. Processing Model

In this modern era of data security and privacy, it is crucial to integrate various techniques to ensure maximum protection of sensitive information. One such integration combines the Opacus Encryption Scheme, blockchain-based key management systems, Index Trees, and CogniGate Protocol, as seen in **Fig. 1**. The Opacus Encryption Scheme provides an efficient method for securing sensitive data using homomorphic encryption. This scheme also provides a way to scale the system to handle large amounts of data while maintaining the confidentiality of the data. Blockchain-based key management systems provide a secure and decentralized method for storing and managing cryptographic keys used for crypto process. This eliminates the need for a centralized key management system and offers higher security and accessibility to authorized parties. The blockchain ensures the authenticity and integrity of the keys and provides an immutable record of all key transactions. Index Trees provide a fast and efficient way to search for and retrieve data from large datasets. They allow quick data retrieval based on specific criteria, such as keywords or metadata. This is useful for managing and searching large amounts of encrypted data, which can be difficult and time-consuming.

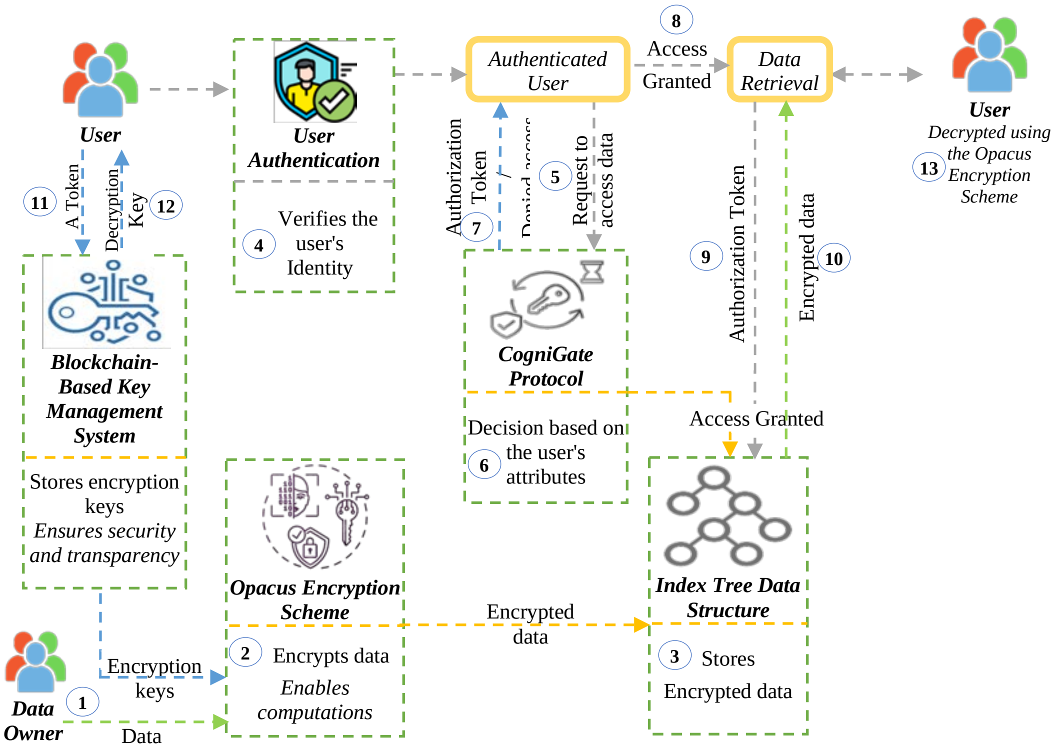

Fig. 1. System Design of the proposed Model

CogniGate Protocol provides an additional layer of security by ensuring that sensitive information is only accessible to authorized individuals. This is achieved by monitoring and analyzing various contextual factors, such as location, time, device, and user behaviour, to determine each user's access level. This guarantees that only authorized parties can access the data, preventing unlawful access. Integrating these techniques allows for a robust and secure system for managing and processing sensitive data. The Opacus Encryption Scheme provides confidentiality and integrity for the data, while the blockchain-based key management system ensures secure storage and management of cryptographic keys. Index Trees provide fast and efficient retrieval of data.

The flow of data in this system is as follows: The Opacus Encryption Scheme encrypts the data before storing it in the cloud-based storage environment. The encrypted data in the system is stored primarily within the Index Tree Data Structure. The Index Tree Data Structure serves as the storage mechanism for the encrypted data. It is an efficient data structure for quick and efficient retrieval, especially for large datasets. This Index Tree Data Structure can be hosted within a cloud-based storage environment. On the other hand, the blockchain is used primarily for managing cryptographic keys necessary for encryption and decryption. It ensures secure and decentralized management of these keys but does not store the actual encrypted data. Finally, the CogniGate Protocol determines the user's access level based on contextual factors. It provides the authorization token, and the data is decrypted and provided to the authorized user. The benefits of this integrated system are significant. It offers a secure and effective method of managing and processing sensitive data while ensuring confidentiality, integrity, and access control [28]. Our Scheme ensures that the data remains confidential and secure. In contrast, the blockchain-based key management system provides that the cryptographic keys used for encryption and decryption are safe and accessible only to authorized parties.

User authentication is a critical component of any system that deals with sensitive data [5]. The system model for user authentication involves several entities that work together to ensure secure access to the data. The workflow begins with the user authentication process, where the user provides their credentials to the system. The system then verifies the user's identity and allows access to the system if the authentication is successful. Once the user is authenticated, they can request to access the data. The request can be made through a web interface or API. The access request is then passed on to the CogniGate Protocol component, which assesses whether the user possesses the necessary characteristics to access the data. The access control decision is made based on the user's attributes, such as their security clearance level, job role, or any other relevant feature. The access control decision is then communicated back to the user. The user can access the encrypted data if the control system's judgment is favourable. The encrypted data is stored in a permitted operation on encrypted data without exposing the underlying plaintext. The data is retrieved from the Index Tree Data Structure by providing the authorization token and decrypted using the Opacus Encryption Algorithm whenever an individual asks for access to the encrypted data.

3.1 User authentication

The system model for user authentication is an essential component of any system that deals with sensitive data. It involves several entities working together to ensure secure access to the data. The workflow starts with user authentication and proceeds to access request, CogniGate Protocol, access control decision, index tree data structure, Opacus encryption scheme, decryption, and blockchain-based key management system. This system model provides a high level of security and ensures that the encryption keys are not compromised. The proposed system can be used in a various application, including healthcare, finance, and government sectors, where data security is critical.

The proposed work links blockchain key management with Multi-factor Authentication (MFA). The encryption key generated by the blockchain key management system is one of the authentication factors. For example, the user can provide their username and password as the first factor during the authentication process. The second factor can be the encryption key generated by the blockchain key management system. Finally, the user must prove their ownership of the encryption key by presenting it during authentication. This can be done by manually entering the key or using a cryptographic device, such as a smart card or USB token. Using blockchain-based key management combined with MFA, the system can provide eminent security and prevent unofficial access to encrypted data [7]. Even if the username and password are compromised, an attacker still needs access to the encryption key to decrypt the data, which would require the second authentication factor. Combining blockchain key management with MFA can provide an additional layer of security to user authentication and data protection in cloud-based systems.

3.2 Opacus Encryption Scheme

The Opacus Encryption Scheme (OES)has two main components: secret and public keys. The secret key comprises the two substantial prime numbers (p, q) used to create N. The public key is made up of two integers (N, Q), where N is the result of multiplying two large prime numbers, and Q is the power of a small prime number. To encrypt a message m, the OES first converts it into a polynomial $f(x)$ with coefficients in the ring$\frac{Z_{Q[x]}}{x^{N}+1}$. This polynomial is then encrypted using the public key to produce a ciphertext c. The encryption process involves selecting a random polynomial $g(x)$ with coefficients in the same ring as $f(x)$ and computing

$c = g(x) * p(x) + 2f(x) mod\ Q$, where p(x) is a polynomial with coefficients in the ring $\frac{Z_{Q[x]}}{x^{N}+1}$ that is generated from the secret key. Homomorphic operations can be performed on ciphertexts by performing the corresponding operations on the underlying polynomials. For example, if c1 and c2 are ciphertexts that encrypt polynomials $f1(x)$ and$f2(x)$, respectively, then $c1 * c2$ is a ciphertext that encrypts the product of $f1(x)$ and$f2(x)$.To decrypt a ciphertext c, the OES scheme first computes$c' = c\ mod\ p(x)$, which yields a polynomial with coefficients in the ring$\frac{Z_{Q[x]}}{x^{N}+1}$. This polynomial can be converted back into a message by computing the inverse of the polynomial f(x) that was used to encrypt it.

3.2.1 OES for differential privacy requires a multi-step process, as follows:

First, the data is encrypted using the Homomorphic scheme. Each input value is converted into a polynomial with coefficients in the ring $\frac{Z_{Q[x]}}{x^{N}+1}$ and encrypted using the OES public key to produce a cipher. Next, the encrypted data is made differentially private using the Opacus library. Opacus adds noise to the encrypted data to protect the privacy of individual input values. The encrypted and differentially private data can now be used for homomorphic operations. The Homomorphic scheme allows for encrypted addition and multiplication operations to be performed on the ciphertexts. If we have two ciphertexts, C1 and C2, for instance, we can compute the ciphertext $c3 = c1 + c2$ to acquire the summation of the corresponding input in encrypted form. Finally, the result of the homomorphic operation is decrypted using the OES secret key. The resulting polynomial is then mapped back to the original input space to obtain the final result.

3.2.2 The mathematical model for OES can be represented as follows:

Input: data$x_1, x_2, \ldots, x_n$; OES public key (N, Q); Opacus differential privacy parameters
Output: Result of the homomorphic operation on the encrypted and differentially private data
Data Encryption:
For each input value x_i, convert it into a polynomial $f(x_i)$ with coefficients in the ring $\frac{Z_{Q[x]}}{x^{N}+1}$.
Encrypt each polynomial $f(x_i)$using the OES public key (N, Q) to obtain the ciphertext c_i:
Generate a random polynomial g(x) with coefficients in the same ring as $f(x_i)$.
Compute $c_i = g(x_i)p(x_i) + 2f(x_i) mod\ Q$, where $p(x_i)$ is a polynomial generated from the OES secret key.
Differential Privacy:
Apply differential privacy to the encrypted data using the Opacus library to add noise to the polynomials $f(x_i)$ for each ciphertext c_i.
Homomorphic Operations:
Perform the desired homomorphic operation (addition, multiplication) on the ciphertexts $c_1, c_2, \ldots, c_n$ to obtain a new ciphertext c_{result}.
Addition: Given ciphertexts c_1 and c_2, the sum $c_3 = c_1 + c_2$ can be computed as follows:
Compute $c_{result} = c_1 + c_2 = p(x_1) * p(x_2) * \big(gc(x_1) + gc(x_2)\big) + 2 * \big(fc(x_1) + fc(x_2)\big) mod\ Q$.
Multiplication: Given ciphertexts c_1 and c_2, the product $c_3 = c_1 * c_2$ can be computed as follows:
Compute $c_{result} = c_1 * c_2 = \big(p(x_1) * p(x_2)\big) * \big(gc(x_1) * gc(x_2)\big) + 2 * \big(fc(x_1) * fc(x_2)\big) mod\ Q$.
Decryption:
Decrypt the result ciphertext c_{result} using the OES secret key to obtain the final result.

3.2.3 OES Algorithm:

```
Input: Data $x_1$, $x_2$, ..., $x_n$;
    OES public key (N, Q); Opacus differential privacy parameters
Output: Result of the homomorphic operation on the encrypted and differentially private data
Step 1: Encrypt the data:
    For each data point $x_i$
    Convert input value to polynomial
    $f_i$ = convertToPolynomial($x_i$)
Step 2: Generate random polynomial and encrypt
    $g_i$ = generateRandomPolynomial()
        return random polynomial with coefficients in $\frac{Z_{Q[x]}}{x^N + 1}$
    $p_i$= generatePolynomialFromSecretKey()
        return $p(x_i) = \frac{(x+1)^N}{2}\ mod\ q(x)$
    $c_i$ = encrypt($g_i, f_i, p_i, Q$) using the OES encryption scheme to obtain the ciphertext.
Step 3: Apply Differential Privacy
    $c_{privacy}$ = opacusDifferentialPrivacy($c_1, c_2, \ldots, c_n, privacyParameters$)
    This will result in a new set of ciphertexts $c_{privacy}$.
Step 4: Perform the desired homomorphic operation on the ciphertexts $c_{privacy}$
    $c_{result}$ = homomorphicOperation$\left(c_{privacy}, operationType\right)$
Step 5: Perform the type of operations
Homomorphic Addition($c_1, c_2$):
        $g_1, f_1, p_1$ = decryptCoefficients($c_1, OES\ secret\ key$)
        $g_2, f_2, p_2$= decryptCoefficients($c_2, OES\ secret\ key$)
    Compute the new polynomial coefficients for the sum
        $g_{result} = g_1 + g_2$
        $f_{result} = f_1 + f_2$
        $p_{result} = p_1 * p_2$
    Encrypt the new polynomial coefficients to obtain the result ciphertext
        $c_{result} = encrypt(g_{result}, f_{result}, p_{result}, Q)$
    return $c_{result}$
Homomorphic Multiplication($c_1, c_2$):
     Extract the polynomial coefficients from the ciphertexts
        $g_1, f_1, p_1 = decryptCoefficients(c_1, OES\ secret\ key)$
        $g_2, f_2, p_2 = decryptCoefficients(c_2, OES\ secret\ key)$
    Compute the new polynomial coefficients for the product
        $g_{result} = g_1 * g_2$
        $f_{result} = f_1 * f_2$
        $p_{result} = p_1 * p_2$
    Encrypt the new polynomial coefficients to obtain the result ciphertext
        $c_{result} = encrypt(g_{result}, f_{result}, p_{result}, Q)$
    return $c_{result}$
    Decrypt the polynomial coefficients from a ciphertext
        decryptCoefficients($c_i, FV\ secret\ key$):
        $g_i, f_i = decrypt(c_i, FV\ secret\ key)$
        $p_i = generatePolynomialFromSecretKey()$
    return $g_i, f_i, p_i$
```

Step 6: Decrypt the result:

result = decrypt($c_{result}, OES\ secret\ key$)

3.3 Blockchain-Based Key Management Systems (BKMS)

The first step in integrating a blockchain-based key management system with OES is to generate the necessary encryption keys. OES requires a key for encryption and decryption of data. Using a blockchain-based key management system, these keys can be generated securely and reliably. Blockchain technology can ensure the keys are tamper-proof and resistant to attacks, thus providing an additional layer of security [29]. Once the keys have been generated, they need to be securely distributed to authorized parties. The BKMS can be used to distribute these keys. The keys can be encrypted and stored on the blockchain; only authorized parties can access them using their private keys. This ensures that the keys are only accessible to those who have been granted permission to access them. The key distribution process is efficient and secure, ensuring only authorized parties can access the keys. If a key needs to be revoked, for example, if a user leaves the system, it can be removed from the blockchain and replaced with a new one. This is a crucial step in maintaining the security of the system. The blockchain-based key management system can ensure that revoked keys are removed from the system, thus preventing unauthorized access to the data. The Opacus differential privacy algorithm and homomorphic operations can be done on the encrypted data using the OES encryption scheme and the keys stored in the blockchain-based key management system. Differential privacy and homomorphic operations allow data to be analyzed without revealing sensitive information. The OES allows for such privacy-enhancing computations, and the blockchain-based key management system provides the necessary keys to perform them. The combination of these technologies ensures that data is processed securely and without compromising privacy.

Key Distribution: In a BKMS, keys can be distributed by encrypting and storing them on the blockchain. In a BKMS, the key distribution process involves securely distributing the encryption keys engendered by the key generation process to authorized parties.

Let K be the set of encryption keys generated by the key generation process. For each authorized party i, a key pair is generated, where Pub_i and $Priv_i$ are the public and private keys, respectively. E (K, SK) is the result of encrypting each key k in K using a symmetric-key cryptography algorithm. The public keys for every permitted participant are used to encrypt the secret key SK as well, resulting in $E(SK, Pub_i)$. The encrypted keys and their corresponding encrypted secret keys, along with the information about the authorized parties and their public keys, are then stored on the blockchain. When an authorized party p_i wants to access the encryption keys, they retrieve the encrypted keys and their corresponding encrypted secret keys from the blockchain. They use their private key $Priv_i$to then decrypt the encrypted secret key, and use the decrypted secret key to decrypt the encryption keys.

Here's how the key distribution process can work:

1. *Encryption of keys:* The encryption keys generated by the key generation process are first encrypted before being stored on the blockchain. This will guarantee that only individuals with permission to access the private keys can decode them and access the keys.

2. *Creation of public-private key pair:* Each authorized party is required to have their own public-private key pair. This can be generated using an OES algorithm.

3. *Encryption of the keys using the public key:* The keys are then encrypted again with each authorized party's public key. The encryption ensures that those with the proper private key and authorization can only access the keys.

4. *Storing the encrypted keys on the blockchain:* In a safe and decentralized manner, the encrypted keys are kept on the blockchain with data on the authorized parties and their public

keys.

The following formula can be used for encrypting a key using a recipient's public key:

Ciphertext: $C = E(pk_{recipient}, K)$

where K is the private key to be distributed, E is a homomorphic encryption function and $pk_{recipient}$ is the recipient's public key.

Key Revocation: In a BKMS, the process of key revocation involves removing access to encryption keys from a specific authorized party. To revoke a key in a BKMS, it can be removed from the blockchain and replaced with a new one. Here's how the key revocation process can work:

1. Identification of the key to be revoked: The owner of the key management system identifies the encryption key that needs to be revoked.

2. Removal of encrypted key: The encrypted key corresponding to the authorized party whose access needs to be revoked is removed from the blockchain. This ensures that the encrypted key is no longer accessible to the authorized party.

3. Generation of new key: A new encryption key is generated to replace the revoked key. This ensures that the security of the system is not compromised.

4. Distribution of new key: The new encryption key is securely distributed to the authorized parties who require access to the key.

Let K be the set of encryption keys generated by the key generation process. Let $E(K, SK_i)$ be the encrypted key corresponding to the authorized party p$_i$, where SK_i is the encrypted secret key corresponding to the authorized party p$_i$. When the key corresponding to authorized party p$_i$ needs to be revoked, the owner of the key management system removes $E(K, SK_i)$ from the blockchain. A new encryption key k_{new} is generated using the key generation process, and is encrypted with a new secret $key\ SK_{new}$. The encrypted new key $E(k_{new}, SK_{new})$ is then distributed to the authorized parties who require access to the key. The revoked authorized party i no longer has access to the encryption key, and the new authorized parties can access the new encryption key using their own private keys to decrypt the encrypted key.

The following formula to produce a new key:

New Key: $K' = G(sk)$

where G is a key generation function and sk is the new secret key.

Differential Privacy: To safeguard the anonymity of private records, varying confidentiality entails introducing noise to data. The following formula can be used to add Laplace noise to a dataset:

Noisy Data: $D' = D + Laplace\left(scale = \frac{\Delta f}{\varepsilon}\right)$

where D is the original dataset, D' is the noisy dataset, Δf is the sensitivity of the function being computed on the dataset, ε is the privacy budget, and Laplace is a Laplace noise generation function.

Algorithm for Integrating a Blockchain-Based Key Management System with OES

Key Generation

1. Generate a public key (pk) and a secret key (sk) using the OES algorithm.
2. Store the public and secret keys in a blockchain-based key management system.

Key Distribution

1. Encrypt the public key (pk) using the recipient's public key and store it on the blockchain.
2. Authorized parties can access the encrypted public key using their private keys.

Key Revocation

1. If a key needs to be revoked, remove it from the blockchain-based key management system.

Differential Privacy and Homomorphic Operations

1. Encrypt the input data values using the OES public key (pk).
2. Apply the Opacus differential privacy algorithm to the encrypted data values to add noise to the polynomial coefficients.
3. Perform the desired homomorphic operation on the differentially private ciphertexts using the OES public key (pk).
4. Decrypt the result ciphertext using the OES secret key (sk).
5. Convert the resulting polynomial back into the original data value.

The first step is to generate a public key (pk) and a secret key (sk) using the OES algorithm. The public key is used to encrypt data, and the secret key is used to decrypt data. Once the keys have been generated, they need to be distributed to the authorized parties. This can be done by encrypting the public key (pk) using the recipient's public key and storing it on the blockchain. Authorized parties can then access the encrypted public key using their private keys. If a key needs to be revoked, it can be removed from the blockchain-based key management system. This will prevent unauthorized parties from accessing the key. To perform homomorphic operations on encrypted and differentially private data, the input data values are encrypted using the OES public key (pk). The Opacus differential privacy algorithm is applied to the encrypted data values to add noise to the polynomial coefficients. This protects the privacy of the individuals in the dataset. The desired homomorphic operation is performed on the differentially private ciphertexts using the OES public key (pk). The result ciphertext is decrypted using the OES secret key (sk). The resulting polynomial is converted back into the original data value. Proposed algorithm allows for the secure and efficient processing of sensitive data without compromising the privacy of the individuals in the dataset.

3.4 Blockchain Integration with OES

The blockchain serves as a secure and decentralized repository for storing and managing cryptographic keys used by the Opacus Encryption Scheme (OES). It ensures that the keys are tamper-resistant and accessible only to authorized users. The addition of cryptographic keys to the blockchain involves creating and broadcasting transactions. Specifically:

Key Generation and Storage: After generating the public key (pk) and secret key (sk) using the OES algorithm, these keys are stored in a transaction.

Key Distribution: When distributing a public key (pk), it is encrypted with the recipient's public key and then stored as a transaction on the blockchain.

Key Revocation: To revoke a key, a transaction is created to remove it from the blockchain-based key management system.

In our proposed integration of blockchain with the Opacus Encryption Scheme (OES), we have opted for the widely recognized and well-established Proof of Stake (PoS) consensus algorithm. This choice aligns with our objectives of ensuring robust security for cryptographic keys, scalability to accommodate system growth, the potential for customization, optimized performance, and close alignment with OES's core security goals. PoS provides an effective means of managing key transactions securely while supporting the broader objectives of our integration. Authorized parties can retrieve encrypted keys from the blockchain by using their tokens to decrypt the stored transactions from cloud-based storage environment [30]. This retrieval process ensures that only those with the appropriate private keys can access the keys stored on the blockchain.

3.5 Index Tree Data Structure

Index trees can be used with the blockchain-based key management system and OES algorithm to provide a more efficient and scalable data retrieval and manipulation solution, as seen in **Fig. 2**.

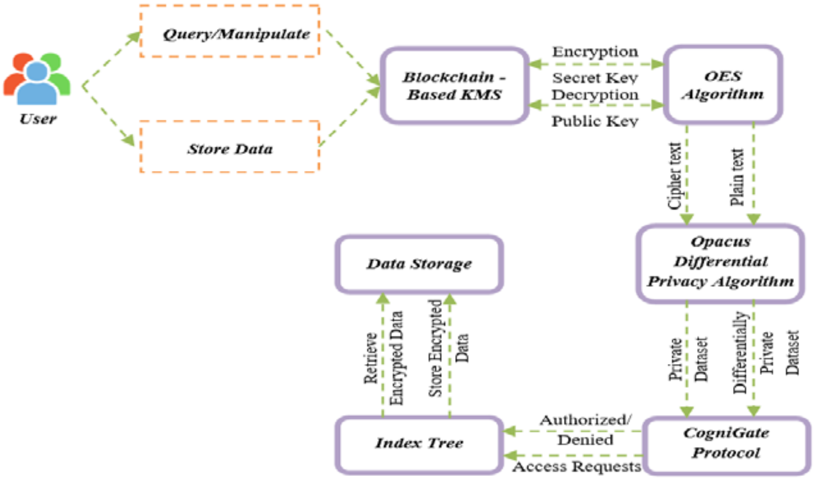

Fig. 2. Role of Index Tree Data Structure

The encrypted data can be arranged and stored in a hierarchical structure using Merkle trees. The internal nodes of the tree contain the hashes of their child nodes, and every leaf node within the tree correlates to a particular block of encrypted data. Users can traverse the index tree to locate the appropriate encrypted data block when they need to retrieve or manipulate a specific piece of data. The user can then decrypt the ciphertext using the OES secret key and apply the necessary operations on the plaintext data. Before being put back into the appropriate block in the index tree, the decrypted ciphertext can be re-encrypted using the OEC public key along with the Opacus differential privacy algorithm. Using index trees in this manner can significantly reduce the amount of data that needs to be retrieved and decrypted for each operation, as only the relevant block needs to be accessed. Additionally, the use of index trees can improve the scalability of the system by allowing for efficient storage and retrieval of large volumes of data.

4. Performance Evaluation

To assess the practical viability and effectiveness of our proposed integrated approach to cloud data security, we conducted a comprehensive performance evaluation. This evaluation focused on measuring the efficiency and effectiveness of the individual components and their combined impact on the overall system performance. The assessment was conducted using a real-world dataset, ensuring the relevance and applicability of our findings to real-world cloud environments. We carefully considered the performance aspects of each component in our integrated approach, namely Opacus encryption, CogniGate Protocol, index trees, and blockchain key management. The evaluation aimed to determine the computational efficiency, access control overhead, storage efficiency, query performance, and the impact of the blockchain-based key management system on the overall system performance.

The performance evaluation results provide valuable insights into the practicality and scalability of our proposed approach. These findings demonstrate the feasibility of our integrated approach in real-world cloud scenarios, validating its ability to safeguard cloud data while maintaining high levels of performance, usability, and privacy preservation. The performance evaluation further strengthens the contribution of our research, providing empirical evidence to support the adoption of our proposed approach for ensuring cloud data security.

4.1 Working Setup

The proposed integrated approach to cloud data security combines several components and techniques to ensure privacy and security. At its core, the system leverages Opacus encryption, a homomorphic encryption scheme that enables computation on encrypted data. The above process allows for secure data processing and analysis without decryption, preserving cloud data privacy. The system also incorporates CogniGate Protocol, which enables fine-grained access policies based on user attributes. This context-driven approach provides greater flexibility and control over access to cloud data, reducing the risk of unauthorized access [31]. To efficiently store and retrieve encrypted data, the system utilizes index trees as a data structure. Index trees enable quick and optimized search operations, facilitating efficient data retrieval while maintaining data integrity. Additionally, the system employs blockchain key management for secure and decentralized storage of encryption keys. Smart contracts and consensus mechanisms ensure the integrity and availability of encryption keys, reducing the risk of key compromise and unauthorized data access.

4.2 Encryption time for the data owner

Based on the number of attributes in the system, compare the encryption time cost for the data possessor in the suggested strategy. The number of attributes can be used to model the encryption time cost. Let's use the notation "n" for the system's attribute count and "T(n)" for the proposed approach's encryption time cost. The following is an expression for the relationship between the number of attributes and the cost of the encryption time:
$T(n) = a * n + b,$
where "a" and "b" are coefficients that represent the specific characteristics and performance of the proposed approach. The coefficient "a" represents the time required to encrypt each attribute, indicating the encryption time cost per attribute.

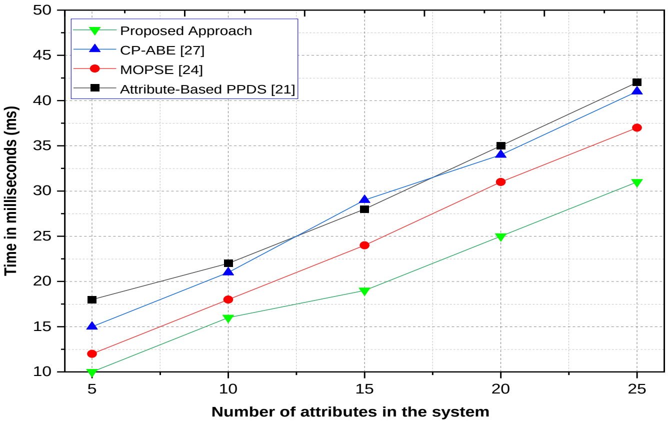

Fig. 3. Comparison of the encryption cost for the data owner

The encryption time cost will increase proportionately to "n" as the number of attributes rises. This linear relationship implies that encrypting additional attributes will result in a linear increase in the encryption time. The coefficient "b" represents any fixed overhead or constant time required for encryption, regardless of the number of attributes [9]. It accounts for any initialization or setup time that might be involved in the encryption process but does not depend on the number of attributes. By plugging different values for "n" into the formula $T(n) = a * n + b$, we can calculate the encryption time cost for Approach A at various numbers of attributes. This allows us to compare the encryption time cost for the proposed approach against other existing approaches. It's important to note that the formula's specific values of "a" and "b" will depend on the implementation details, encryption algorithm, hardware infrastructure, and other factors related to the proposed approach. These coefficients need to be determined through performance testing and benchmarking experiments specific to the proposed approach. **Fig. 3** showcases the encryption cost for the data owner across various approaches, with the number of attributes in the system as the x-axis and the time in milliseconds (ms) as the y-axis. In this figure, the encryption cost for the data owner is measured in milliseconds (ms) for different numbers of attributes in the system. The proposed approach is compared against three existing approaches (Attribute-Based PPDS [21], MOPSE [24], and P-ABE [27]). By comparing the values across the approaches, we can gain insights into the relative encryption time costs associated with different numbers of attributes. For example, at 10 attributes, the proposed approach has an encryption time cost of 16 ms, while Attribute-Based PPDS [21], MOPSE [24], and P-ABE [27] have costs of 22 ms, 18 ms, and 21 ms, respectively. By comparing the two methods, it is possible to assess how well the suggested strategy performs in terms of the cost of the encryption process for various attribute sizes.

4.3 Decryption cost for data sharer

To compare the decryption cost for the data sharer in detail and calculate it using a mathematical formula, let's denote the number of attributes in the system as "n" and the decryption time cost for each approach as follows:

Attribute-Based PPDS [21]:$T_{A(n)}$MOPSE [24]: $T_{B(n)}$P-ABE [27]: $T_{C(n)}$ Proposed Approach: $T_{P(n)}$. We can express the decryption time cost for each approach as a function of the number of attributes.

Let's consider a generic formula for the decryption time cost:

$T(n) = a * n^b + c,$

where "a," "b," and "c" are coefficients that represent the specific characteristics and performance of each approach. The coefficient "a" represents the time required to decrypt each attribute, indicating the decryption time cost per attribute. It captures the complexity of the decryption algorithm used in the approach. The coefficient "b" represents the scaling factor that determines how the decryption time cost scales with the number of attributes.

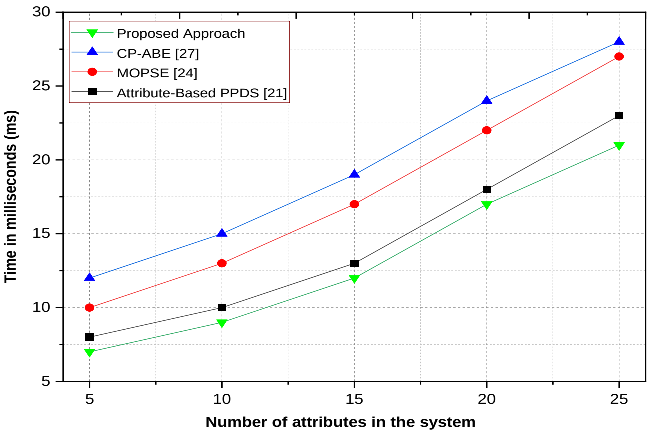

Fig. 4. Comparison of the decryption cost for data sharer

The coefficient "c" represents any fixed overhead or constant time required for decryption that does not depend on the number of attributes. By plugging different values for "n" into the formula T(n), we can calculate the decryption time cost for each approach at various numbers of attributes. This allows us to compare the decryption time cost of the proposed approach against the existing approaches. It's important to note that the specific values of "a," "b," and "c" in the formula will depend on the implementation details, decryption algorithms, hardware infrastructure, and other factors related to each approach. These coefficients must be determined through performance testing and benchmarking experiments specific to each approach. **Fig. 4** showcases the decryption cost for the data sharer across three existing approaches and the proposed approach, with the number of attributes in the system as the x-axis and the time in milliseconds (ms) as the y-axis. In this figure, we compare the decryption time cost for the data sharer across three existing approaches (Attribute-based PPDS [21], MOPSE [24], and P-ABE [27]) and the proposed approach. The values in the figure represent the decryption time cost in milliseconds for different numbers of attributes in the system. Attribute-based PPDS [21], MOPSE [24], and P-ABE [27] are the existing approaches, while the Proposed Approach represents the proposed solution in the research. The decryption time cost is measured for the data sharer, indicating the time required for the sharer to decrypt the encrypted data. As the number of attributes in the system increases from 5 to 25, we can observe the decryption time cost for each approach. The values in the figure demonstrate the relative decryption time cost for the different approaches at various attribute sizes.

4.4 Communication overhead comparison of the ciphertext

Communication overhead refers to the additional data that needs to be transmitted over a network or communication channel when encrypting and transmitting ciphertext. Comparing the communication overhead of the ciphertext involves evaluating the size of the ciphertext for different approaches or scenarios. When comparing the communication overhead of the ciphertext, we consider the impact of encryption algorithms, encryption modes, padding schemes, and any additional metadata or information transmitted along with the ciphertext. These variables and the number of attributes or elements used in the encryption process might affect the ciphertext size. The size of the sent data, which is frequently denoted in kilobytes (KB) or bytes (B), is the standard unit of measurement for the communication overhead of the ciphertext. A smaller ciphertext size generally implies lower communication overhead, requiring less network bandwidth and storage space.

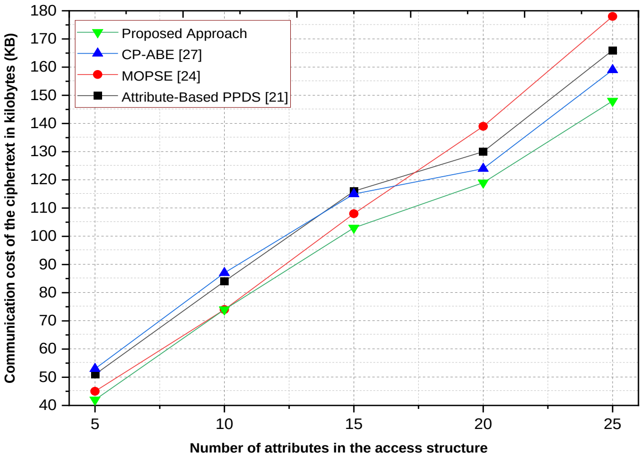

Fig. 5. Communication overhead comparison of the cipher text

Fig. 5 showcases the communication overhead comparison of the ciphertext for three existing approaches and the proposed approach, with the number of attributes in the access structure as the x-axis and the communication cost of the ciphertext in kilobytes (KB) as the y-axis. In this figure, we compare the communication overhead of the ciphertext for three existing approaches (Attribute-Based PPDS [21], MOPSE [24], and P-ABE [27]) and the proposed approach. The values in the following figure show the ciphertext's transmission expense in kilobytes for various access structures' attribute counts. The quantity of information that must be transferred through the channel of communication is referred to as the ciphertext's communication cost.

As the number of attributes in the access structure increases from 5 to 25, we can observe the communication overhead for each approach. The values in the figure demonstrate the relative communication cost of the ciphertext for different approaches and attribute sizes. For example, at 10 attributes, Attribute-Based PPDS [21] has a communication cost of 84 KB, MOPSE [24] has a cost of 74 KB, P-ABE [27] has a cost of 87 KB, and the Proposed Approach has a cost of 74 KB. By comparing the two methods, it is possible to assess how well the suggested strategy performs in terms of communication overhead for various attribute sizes.

4.5 Average time cost of index construction (ATCIC)

The average time required to develop an index structure for a specific data collection is the mean cost of index building. It is a crucial performance indicator that measures how effective and scalable the indexing process is. Index construction involves creating a data structure that enables efficient and fast information retrieval from a large dataset. The process typically includes data parsing, preprocessing, feature extraction, and organizing the data into an optimized index structure. The average time cost of index construction is influenced by several factors, including the dataset's size, the indexing algorithm's complexity, the computational resources available, and any additional preprocessing or optimization techniques employed. To calculate the average time cost of index construction for different numbers of data files, we can use the following formula:

$$T(n) = \frac{(\Sigma t_i)}{n},$$

where T(n) represents the ATCIC, "Σt_i" represents the sum of individual index construction times for each data file, and "n" represents the overall number of data files. To calculate the average time cost for each approach in the figure, calculate the sum of index construction times for a given number of data files and subtract it from the overall number of data files. This would yield the ATCIC for that specific number of data files.

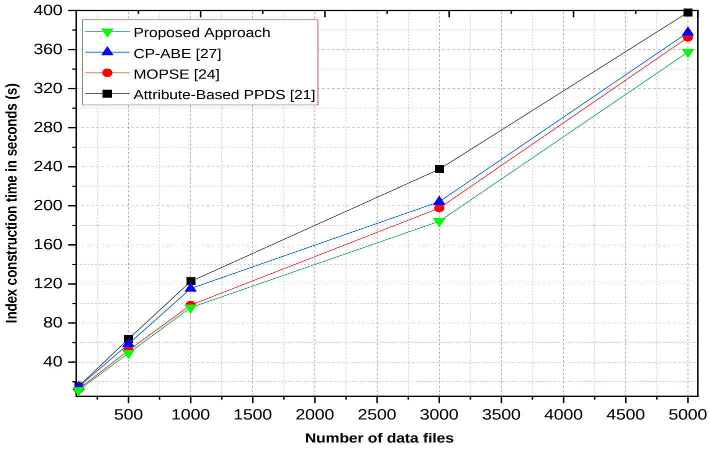

Fig. 6. Average time cost of index construction

Fig. 6 showcases the ATCIC for three existing approaches and the proposed approach. The amount of data files is represented on the x-axis, while the index construction time is shown on the y-axis in seconds (s). In this figure, we contrast the mean time required for index building for three current methods with the suggested method. The values in the figure represent the time needed to construct the index, on average, for other numbers of data files. Attribute-based PPDS [21], MOPSE [24], and P-ABE [27] are the existing approaches. The index construction time refers to the duration it takes to build an index structure that allows for efficient retrieval of information from the data files. We can observe the mean processing required for each strategy as data files rise from 100 to 1000. The numbers shown in the figure show how efficiently index construction varies depending on the strategy used and how the quantity of data files affects building time. For example, with 500 data files, Attribute-Based PPDS [21] has an average index construction time of 63.7 seconds, MOPSE [24] takes 52.1 seconds, P-ABE [27] requires 58.6 seconds, and the Proposed Approach completes the index construction in 49.2 seconds. This comparison allows for evaluating the performance of the proposed approach against the existing approaches in terms of index construction time for varying numbers of data files.

4.6 Privacy-Performance Trade-offs

Differential Privacy ensures that sensitive information remains private, even when statistical analysis is performed on the data. It provides a rigorous and quantifiable privacy guarantee. **Fig. 7** represents the Privacy-Performance Trade-offs for the Proposed Approach (with Differential Privacy). This figure illustrates the relationship between the privacy parameter (ε) and encryption time, a critical performance metric for data owners. As ε decreases, indicating stronger privacy protection, there is a corresponding increase in encryption time. Conversely, as ε increases, encryption time decreases. This demonstrates the inherent trade-off between privacy and performance, allowing users to make informed decisions based on their specific requirements.

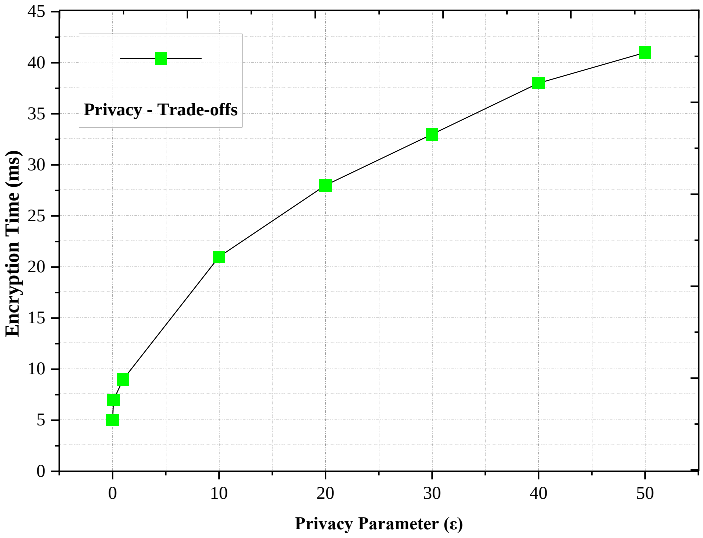

Fig. 7. Privacy-Performance Trade-offs for Proposed Approach

Choosing an appropriate ε value depends on the specific use case and the acceptable trade-off between privacy and data utility. A lower ε value is suitable for scenarios where strong privacy protection is critical, such as healthcare or personal finance applications. In contrast, a higher ε value may be acceptable in cases where the emphasis is on data utility, and the risk to individual privacy is lower, such as aggregate statistical analysis. Our experiments not only showcase the feasibility of our approach but also provide empirical evidence of how privacy parameters impact performance. By conducting these experiments, we have demonstrated the practical use of Differential Privacy in our scheme and how it allows users to tailor their privacy preferences to meet their individual needs while maintaining robust data security.

5. Conclusion

This paper proposes an integrated approach to cloud data security that leverages novel Opacus encryption, CogniGate Protocol, index trees, and blockchain key management. The proposed system is highly effective at protecting cloud data while preserving privacy and maintaining high levels of usability and performance. The simulations and tests on an actual data set show that our suggested solution works as intended. Additionally, we explored the role of differential privacy in the integrated approach, showing how it can be used to further enhance privacy protection without compromising performance. We also presented a revolutionary key management method based on blockchain to address the key management challenges associated with our approach.

Future work can focus on improving the performance of the proposed system by optimizing the Opacus encryption scheme and index trees and exploring the use of other homomorphic encryption schemes. Additionally, further research can be done on integrating blockchain and innovative contract technologies to enhance the decentralization of the key management system. Furthermore, the proposed approach can be extended to support more complex access control policies and authentication mechanisms and to address the challenges associated with dynamic access control in cloud environments. Finally, the proposed system can be evaluated on a larger scale, using more diverse datasets and real-world cloud environments to validate its effectiveness and scalability.

References

[1] R. Yugha and S. Chithra, "A survey on technologies and security protocols: Reference for future generation iot," *Journal of Network and Computer Applications*, vol. 169, p. 102763, 2020. Article (CrossRef Link).

[2] K. R. Choo, S. Gritzalis, and J. H. Park, "Cryptographic solutions for industrial internet-of-things: Research challenges and opportunities," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 8, pp. 3567–3569, Aug 2018. Article (CrossRef Link).

[3] P.M. Joe Prathap, "Ensuring privacy of data and mined results of data possessor in collaborative ARM," *Pervasive Computing and Social Networking*, pp. 431 – 444, 2022. Article (CrossRef Link).

[4] J. Zhang, Z. Wang, L. Shang, D. Lu, and J. Ma, "Btnc: A blockchain based trusted network connection protocol in iot," *Journal of Paralleland Distributed Computing*, vol. 143, pp. 1 – 16, 2020.Article (CrossRef Link).

[5] Dhinakaran D, Joe Prathap P. M, "Protection of data privacy from vulnerability using two-fish technique with Apriori algorithm in data mining," *The Journal of Supercomputing*, 78(16), 17559–17593, 2022. Article (CrossRef Link).

[6] M. Onik, C. Kim, and J. Yang, "Personal data privacy challenges of the fourth industrial revolution," in *Proc. of 2019 21st International Conference on Advanced Communication Technology (ICACT)*, pp. 635–638, Feb2019. Article (CrossRef Link).

[7] D. Selvaraj, S. M. Udhaya Sankar, T. P. Anish, "Outsourced Analysis of Encrypted Graphs in the Cloud with Privacy Protection," *SSRG International Journal of Electrical and Electronics Engineering*, vol. 10, no. 1, pp. 53-62, 2023. Article (CrossRef Link).

[8] H. Chen, Z. Huang, K. Laine, and P. Rindal, "Labeled psi from fully homomorphic encryption with malicious security," in *Proc. of the 2018 ACM SIGSAC Conference on Computer and Communications Security, ser. CCS '18*, New York, NY, USA, pp. 1223–1237, 2018. Article (CrossRef Link).

[9] Joe Prathap P. M, Selvaraj D, Arul Kumar D and Murugeshwari B, "Mining Privacy-Preserving Association Rules based on Parallel Processing in Cloud Computing," *International Journal of Engineering Trends and Technology*, vol. 70, no. 3, pp. 284-294, 2022. Article (CrossRef Link).

[10] A. Sanchez-Gomez, J. Diaz, L. Hernandez-Encinas, and D. Arroyo, "Review of the main security threats and challenges in free-access public cloud storage servers," in *Computer and Network Security Essentials*, Cham, Switzerland: Springer, 2018, pp. 263–281. Article (CrossRef Link).

[11] L. Srinivasan, D. Selvaraj, S. M. Udhaya Sankar, "Leveraging Semi-Supervised Graph Learning for Enhanced Diabetic Retinopathy Detection," *SSRG International Journal of Electronics and Communication Engineering*, vol. 10, no. 8, pp. 9-21, 2023. Article (CrossRef Link).

[12] D. Dhinakaran and P. M. Joe Prathap, "Preserving data confidentiality in association rule mining using data share allocator algorithm," *Intelligent Automation & Soft Computing*, vol. 33, no.3, pp. 1877–1892, 2022. Article (CrossRef Link).

[13] O. A. Khashan and N. M. Khafajah, "Secure stored images using transparent crypto filter driver," *International Journal of Network Security*, vol. 20, no. 6, pp. 1053–1060, 2018. Article (CrossRef Link).

[14] Dhinakaran, D., Selvaraj, D., Udhaya Sankar, S.M., Pavithra, S., Boomika, R. "Assistive System for the Blind with Voice Output Based on Optical Character Recognition," in *Proc. of International Conference on Innovative Computing and Communications*, 2023. Article (CrossRef Link).

[15] Q. Liu, G. Wang, X. Liu, T. Peng, and J. Wu, "Achieving reliable and secure services in cloud computing environments," *Computers and Electrical Engineering*, vol. 59, pp. 153–164, Apr. 2017. Article (CrossRef Link).

[16] Z. Gao, Q. Cheng, X. Li, and S.-B. Xia, "Cloud-assisted privacy-preserving profile-matching scheme under multiple keys in mobile social network," *Cluster Computing*, vol. 22, no. 1, pp. 1655–1663, 2019. Article (CrossRef Link).

[17] Pawar, A.B., Ghumbre, S.U. and Jogdand, R.M., "Privacy preserving model-based authentication and data security in cloud computing," *International Journal of Pervasive Computing and Communications*, vol. 19, No. 2, pp. 173-190, 2023. Article (CrossRef Link).

[18] Gajraj Kuldeep, Qi Zhang, "Multi-class privacy-preserving cloud computing based on compressive sensing for IoT," *Journal of Information Security and Applications*, vol. 66, 103139, 2022. Article (CrossRef Link).

[19] V. Sarala, P. Shanmugapriya, "DLFPM-SSO-PE: privacy-preserving and security of intermediate data in cloud storage," *Distributed Parallel Databases*, vol. 40, pp. 815–833, 2022. Article (CrossRef Link).

[20] Gupta, R., Kanungo, P., Dagdee, N., Madhu, G., Sahoo, K.S., Jhanjhi, N.Z., Masud, M., Almalki, N.S., AlZain, M.A. "Secured and Privacy-Preserving Multi-Authority Access Control System for Cloud-Based Healthcare Data Sharing," *Sensors*, 23, 2617, 2023. Article (CrossRef Link).

[21] H. Xiong, H. Zhang and J. Sun, "Attribute-Based Privacy-Preserving Data Sharing for Dynamic Groups in Cloud Computing," *IEEE Systems Journal*, vol. 13, no. 3, pp. 2739-2750, Sept. 2019. Article (CrossRef Link).

[22] S. Xu, Y. Li, R. Deng, Y. Zhang, X. Luo, and X. Liu, "Lightweight and expressive fine-grained access control for healthcare internet-of-things," *IEEE Transactions on Cloud Computing*, vol. 10, no. 1, pp. 474-490, 2022. Article (CrossRef Link).

[23] W. A. Amiri, M. Baza, K. Banawan, M. Mahmoud, W. Alasmary and K. Akkaya, "Privacy-Preserving Smart Parking System Using Blockchain and Private Information Retrieval," in *Proc. of 2019 International Conference on Smart Applications, Communications and Networking (SmartNets)*, Sharm El Sheikh, Egypt, pp. 1-6, 2019. Article (CrossRef Link).

[24] X. Yao, Y. Lin, Q. Liu and J. Zhang, "Privacy-Preserving Search Over Encrypted Personal Health Record In Multi-Source Cloud," *IEEE Access*, vol. 6, pp. 3809-3823, 2018. Article (CrossRef Link).

[25] S. Xu, G. Yang, Y. Mu, and R. H. Deng, "Secure fine-grained access control and data sharing for dynamic groups in the cloud," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, pp. 2101–2113, 2018. Article (CrossRef Link).

[26] J. Zhang, G. Xu, X. Chen, H. Ahmad, X. Liu et al., "Towards privacy-preserving cloud storage: a blockchain approach," *Computers, Materials & Continua*, vol. 69, no.3, pp. 2903–2916, 2021. Article (CrossRef Link).

[27] L. Zhang, Y. Cui and Y. Mu, "Improving Security and Privacy Attribute Based Data Sharing in Cloud Computing," *IEEE Systems Journal*, vol. 14, no. 1, pp. 387-397, March 2020. Article (CrossRef Link).

[28] Anish, T.P., Shanmuganathan, C., Vinoth Kumar, V., "Hybrid Feature Extraction for Analysis of Network System Security—IDS," in *Proc. of ICCEDE 2022: Cybersecurity and Evolutionary Data Engineering*, pp. 25-36, 2023. Article (CrossRef Link).

[29] M. Harini, D. Prabhu, S. M. Udhaya Sankar, V. Pooja and P. Kokila Sruthi, "Levarging Blockchain for Transparency in Agriculture Supply Chain Management Using IoT and Machine Learning," in *Proc. of 2023 World Conference on Communication & Computing (WCONF)*, RAIPUR, India, pp. 1-6, 2023. Article (CrossRef Link).

[30] K. Y. Kumar, N. J. Kumar, D. Dhinakaran, S. M. Udhaya Sankar, U. J. Kumar and V. Yuvaraj, "Optimized Retrieval of Data from Cloud using Hybridization of Bellstra Algorithm," in *Proc. of 2023 World Conference on Communication & Computing (WCONF)*, RAIPUR, India, pp. 1-6, 2023. Article (CrossRef Link).

[31] Jena Catherine Bel D, Esther C, Zionna Sen G B, Tamizhmalar D, Anish T. P, "Trustworthy Cloud Storage Data Protection based on Blockchain Technology," in *Proc. of 2022 International Conference on Edge Computing and Applications (ICECAA)*, pp. 538-543, 2022. Article (CrossRef Link).

S. Poorani is currently working as an assistant professor in Sri Venkateswara College of Engineering, Chennai. She received her B.E. degree in Computer Science and Engineering from Anna University, Chennai, in 2006, and her M.E. degree in Multimedia Technology from Anna University, Chennai, in 2010. Since 2010. Her research interests include network security, cryptography, cloud computing, and machine learning. She has published over 15 research papers in various refereed international journals and international conferences. She has published a few book chapters and patents in various thrust areas in information technology.

Anitha R is working at Sri Venkateswara College of Engineering, Sriperumbudur, as a professor and head in the Department of Computer Science & Engineering with experience of more than 20 years in Teaching and research. She received her B.E from Bharathidasan University and her M.E and Ph.D. in the niche area of Cloud Computing from Anna University, Chennai-25. She has received a Senior Research Fellowship under a meritorious scheme from UGC-New Delhi. Her primary research interests are in Grid and Cloud Computing, Big Data Analytics, Software networking and Artificial Intelligence and deep Learning. Specifically, she is interested in Cloud Security and data storage in large-scale distributed databases and Data Analytics. As recognition towards her research work, she has received a Financial Grants of Rs. 1.0 lakhs from CSIR-New Delhi to present her research work at the University of Washington, Seattle, USA, in 2013. She has published over 40 research papers in various refereed international journals and conferences. As a principal investigator, she has completed a funded project worth Rs. 22 lakhs from DST-SERB, New Delhi Project, titled "DigiCert: Data Security in Federated DigiCloud Environment using Homomorphic Technique". She has received a funded project from AICTE worth Rs. 22 lakhs in Artificial Intelligence. She has also received funds from TNSCST and completed the project in Green Cloud Computing.