STAP
Smart Technologies Academic Press

Journal of Cyber Security and Risk Auditing

https://www.jcsra.thestap.com/

Check for updates

Secure Access Control using Ciphertext Policy Attribute-based Encryption with Performance Optimization in Cloud Computing

Siti Dhalila Mohd Satar[1], Masnida Hussin[2], Mohamad Afendee Mohamed [1], Nazirah Abd Hamid[1], Mohd Fadzil Abd Kadir[1], Roslinda Muda[1], Joshua Samual[3]

[1]*Faculty Informatics and Computing, Universiti Sultan Zainal Abidin, Besut Campus, Terengganu, Malaysia.*

[2]*Faculty of Information Technology and Computer Science, Univerisiti Putra Malaysia, Selangor, Malaysia.*

[3]*Faculty of Computing, Engineering and Technology, Asia Pacific University of Technology and Innovation, Malaysia*

ARTICLE INFO

Article History

Received: 17-03-2025
Revised: 30-05-2025
Accepted: 23-08-2025

Vol.2025, No.4

DOI:
https://doi.org/10.63180/jcsra.thestap.2025.4.8

***Corresponding author.**
Email:
masnida@upm.edu.my

Orcid:

https://orcid.org/0000-0003-1063-8502

ABSTRACT

Cipher text-Policy Attribute-Based Encryption is an access control technique widely used in cloud computing for enforcing data access policies based on attributes. However, existing Cipher text-Policy Attribute-Based Encryption schemes suffer from an issue of user's privacy leakage and increasing cipher text size as the number of attributes in the access policy grows, leading to computational overheads and security vulnerabilities. In this research, we propose a modified Cipher text-Policy Attribute-Based Encryption scheme that addresses both privacy preservation and the problem of increasing cipher text size. Our system achieves a significant reduction in cipher text size, regardless of the number of user-given attributes, thereby ensuring efficiency and enhancing data privacy. We accomplish this by implementing an access policy hiding mechanism that conceals the attribute location and adapting OptiSize Text to eliminates redundant text in input files. Experimental results demonstrate the effectiveness of our proposed system in overcoming challenges related to data privacy and computational overheads. By significantly reducing encryption time and cipher text size, our scheme improves efficiency and enhances the security of cloud computing applications.

Keywords: Access Control, CP-ABE, Performance Optimization, Cloud Computing

How to cite the article
Satar, S. D. M., Hussin, M., Afendee Mohamed, M., Abd Hamid, N., Abd Kadir, M. F., Muda, R., & Samual, J. (2025). Secure Access Control using Ciphertext Policy Attribute-based Encryption with Performance Optimization in Cloud Computing. Journal of Cyber Security and Risk Auditing, 2025(4), 287–305.

1. Introduction

The integration of cloud computing has revolutionized data storage and sharing by offering scalability and convenience. Cloud storage enables organizations and individuals to store vast amounts of data on remote servers, ensuring accessibility from any device, anywhere. However, this convenience comes with significant risks. Relying on cloud service providers (CSPs) raises concerns about data security, privacy breaches, and unauthorized access—particularly in multi-tenant environments where resources are shared among multiple users. Ensuring data confidentiality and integrity is essential to maintaining trust in cloud services.

Access control serves as a fundamental defense mechanism in cloud computing by regulating user privileges. An access control system defines rights, permissions, and privileges for authorized users based on predefined security policies (Kahani et al., 2016; Vijayalakshmi & Jayalakshmi, 2021; Younis et al., 2014). Its primary objective is to prevent unauthorized access and restrict users to permitted tasks. Various access control models achieve this, as illustrated in Figure 1.

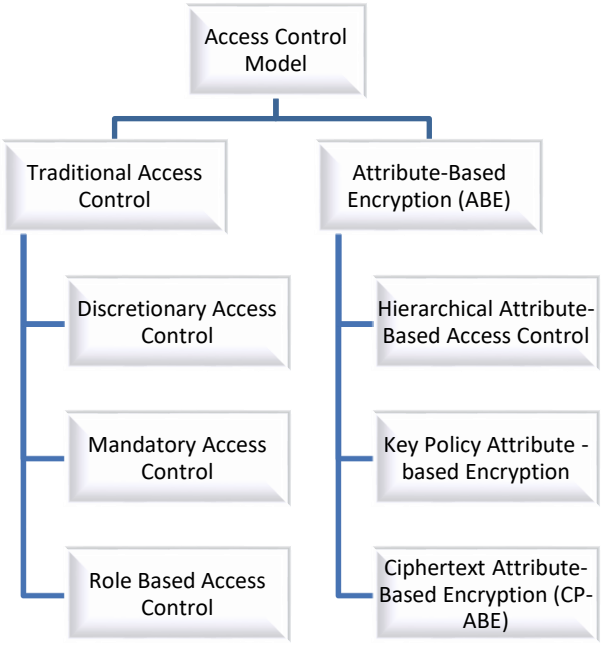


Figure 1. Access Control

Traditional access control models such as Role-Based Access Control (RBAC), Discretionary Access Control (DAC), and Mandatory Access Control (MAC) have been widely implemented (Vijayalakshmi & Jayalakshmi, 2021; Younis et al., 2014). However, these models often fail to address the diverse security challenges of cloud computing, including multi-tenant hosting and heterogeneous security policies. As a result, modern access control mechanisms are needed to effectively mitigate cloud security risks (Lopez & Rubio, 2018).

As highlighted by Narasingapuram & Ponnavaikko (2021), cloud access control is a critical decision for organizations, as it heavily depends on infrastructure and data security. However, access control alone is insufficient for comprehensive data protection. A combination of security mechanisms is required to create a secure data-sharing environment. Ciphertext-Policy Attribute-Based Encryption (CP-ABE) stands out as a cryptographic technique that ensures both privacy and fine-grained access control in cloud computing.

CP-ABE associates ciphertexts with attribute-based access policies, while user private keys correspond to attribute sets. Initially introduced by Goyal et al. (2006) for managing cloud data access, CP-ABE was later refined by Bethencourt et al. (2007) to improve data-sharing efficiency between owners and users. This approach enables data owners to define access policies for encrypted data, ensuring that only authorized users can decrypt it. However, a major drawback of CP-ABE is

that access policies are typically transmitted in plaintext, creating security risks. Malicious users could exploit this information to infer attributes and potentially gain unauthorized access. To address this vulnerability, researchers have proposed enhanced CP-ABE schemes (Edemacu et al., 2020; Khuntia & Kumar, 2018; R. Zhang et al., 2017; Z. Zhang et al., 2021; Zhao et al., 2022).

Yan et al. (2017) emphasize that while CP-ABE has significantly improved secure data sharing, privacy concerns persist due to plaintext access policies. These policies expose sensitive details to authorized entities, potentially leading to misuse. To mitigate this issue, encryption schemes must conceal both the message and access policy details, ensuring comprehensive privacy protection for data owners and users.

However, enhanced security often comes at the expense of performance. As noted by P et al. (2018), CP-ABE faces efficiency challenges, including storage, communication, and computational overheads. Key factors affecting efficiency include the size of public and secret keys, as well as ciphertext size. Susilo et al. (2018) explain that ciphertext size in CP-ABE depends on the number of attributes in the access policy. As policies grow, ciphertext size increases, leading to higher transmission overhead and storage requirements. This research aims to design a CP-ABE scheme that reduces storage costs by generating shorter ciphertexts.

Ensuring the security of cloud storage requires robust encryption mechanisms and effective access control measures to prevent unauthorized access and mitigate data breaches. CP-ABE has emerged as a promising cryptographic solution that enables fine-grained access control by associating data access policies with ciphertext. This approach ensures that only users with attributes satisfying the specified policy can decrypt the data. However, existing CP-ABE schemes often expose access policies in plaintext, leading to privacy vulnerabilities. Additionally, inefficiencies in storage and computational overheads limit the scalability of CP-ABE implementations.

The key contributions of this paper are as follows

1. Introducing an access policy hiding scheme that leverages logical connective operations to enhance user privacy while improving encryption efficiency, even in the presence of security attacks.

2. Proposing a scheme called OptiSize to reduce ciphertext size and eliminate excessive data inconsistencies in files. By optimizing data structures, this scheme minimizes storage costs and transmission overhead, thereby improving the overall effectiveness of CP-ABE.

The paper is structured as follows: Section 2 reviews related work, while Section 3 presents preliminary studies. Section 4 discusses the CP-ABE system model, followed by Section 5, which details the proposed CP-ABE scheme. Section 6 describes the experimental configuration, and Section 7 provides evaluation and implementation details. Finally, Section 8 concludes the paper and outlines future research directions.

2. Related Works

In classical Attribute-Based Encryption (ABE), the access policy associated with an encrypted message is often transmitted in an unencrypted format, making it vulnerable to unauthorized access. This exposure allows malicious parties to extract attribute details from the access policy and potentially disclose sensitive information. To address this issue, several researchers (Bethencourt, Sahai, et al., 2007; X. Liu et al., 2018; L. Zhang et al., 2020) have developed novel ABE systems. One such system is Ciphertext-Policy Attribute-Based Encryption (CP-ABE), introduced by X. Liu et al. (2018), which enables fine-grained data access control in cloud environments. CP-ABE has also been proposed by Sabitha & Rajasree (2017) and L. Zhang et al. (2020) as a mechanism to improve data-sharing efficiency, allowing data owners to define encryption policies that restrict access to authorized users only.

However, as noted by Ramachandra et al. (2017), while many CP-ABE schemes successfully facilitate secure and efficient data sharing, they often overlook the privacy concerns of data owners and consumers. To mitigate these issues, several researchers have introduced enhancements to CP-ABE. Nishide et al. (2008) proposed a policy-hiding CP-ABE solution that achieves security under the Random Oracle Model (ROM), based on the Decisional Bilinear Diffie-Hellman (DBDH) and Decisional Linear assumptions. However, this scheme only provides selective security, limiting its applicability.

To achieve full security against adaptive adversaries, Phuong et al. (2016a), Y et al. (2016), L. Zhang et al. (2020), and Y. Zhang et al. (2018) designed CP-ABE solutions based on composite-order groups, ensuring stronger security guarantees

under new cryptographic assumptions. Nevertheless, the schemes proposed by Phuong et al. (2016a) and Y et al. (2016) support only AND-gate policies, restricting their expressiveness. Additionally, these schemes employ a small-universe construction, where ciphertext size scales linearly with the total number of attributes. To enhance policy expressiveness, Cui et al. (2018) developed a CP-ABE scheme incorporating Linear Secret Sharing Schemes (LSSS). They claimed that their scheme is both secure and practical. However, their analysis lacks a thorough comparative evaluation with existing works, making it difficult to assess its true effectiveness.

A critical aspect of CP-ABE is the efficiency of the decryption process, which has become a major research focus. To address this, various schemes have been proposed to accelerate decryption. H. Li et al. (2017) introduced a CP-ABE scheme that prioritizes attribute privacy protection while maintaining decryption efficiency. Their approach implements a policy-hiding mechanism, where attribute matching occurs before decryption, reducing computational overhead. However, their scheme does not provide fully secure anonymous CP-ABE, leaving room for potential privacy risks.

In contrast, Wang et al. (2018) proposed a CP-ABE scheme optimized for lightweight decryption on mobile devices. Their model ensures that neither the data server nor the data management server can decrypt ciphertext unless a user meets the access control conditions. This approach enhances security by preventing unauthorized access while ensuring decryption remains computationally feasible for resource-constrained devices. Meanwhile, Y. Zhang et al. (2018) introduced a large-universe CP-ABE scheme with policy hiding, improving both expressiveness and computational efficiency. Their model supports LSSS policies and reduces computational overhead by requiring only two bilinear pairings for attribute matching. The security of their scheme has been rigorously demonstrated in standard cryptographic models.

Beyond traditional CP-ABE applications, researchers have explored policy-hiding CP-ABE for cloud-based IoT systems. Hao et al. (2019) developed a scheme aimed at providing fine-grained access control with fully concealed attributes. They introduced a fuzzy attribute positioning mechanism based on garbled Bloom filters, enabling efficient attribute retrieval and ciphertext decryption for authorized users. However, their performance evaluation was limited, relying only on simulations of encryption and decryption algorithms using four elliptic curves.

Further extending CP-ABE for IoT, Yin et al. (2022) designed a policy-hiding attribute-based encryption scheme with keyword search for Cloud-assisted IoT systems. Their scheme, constructed using prime order groups, offers a practical and secure solution suitable for real-world implementation. However, it is limited to static data handling, which may constrain its applicability in dynamic IoT environments.

Despite these advancements, existing CP-ABE schemes with policy hiding still face performance challenges. While they provide strong data security and privacy, many suffer from high storage and computational overheads, limiting scalability. As a result, further research is needed to optimize CP-ABE implementations by reducing ciphertext size and computational complexity. Achieving an optimal balance between security, storage efficiency, and computational performance remains a crucial challenge for CP-ABE, particularly in large-scale cloud environments.

3. Preliminaries

In this section, basic knowledge in CP-ABE is presented.

3.1 Bilinear Pairing

In the CP-ABE, a composite order bilinear group with a distinct prime N is utilized to generate the public key. This approach is adopted from the work of L. Zhang et al. (2019). The generation of the public key involves the use of bilinear pairings. The algorithm takes an input of 1^{λ} where λ represents the security parameter, and produces a tuple denoted as ($\mathbb{G}$, $\mathbb{G}_T$, e, p_1, p_2, p_3, p_4). The cyclic group $\mathbb{G}$ and $\mathbb{G}_T$ have an order of $N= p_1p_2p_3p_4$, and the mapping function e: $\mathbb{G}$ x $\mathbb{G} \rightarrow \mathbb{G}_T$, satisfies properties:

- Bilinearity: $\forall g, y \in \mathbb{G}$ and $d, w \in \mathbb{Z}_N$ where $e(g^d, y^w) = e(g, y)^{dw}$
- Non-degenerate: $\exists g \in \mathbb{G}$ such that $e(g, g)$ has order N in $\mathbb{G}$
- Computable: the value of e can be efficiently computed.

It is crucial for the computable e pairing function to be effective, as this ensures that the implementation of a device utilizing e is fast enough to avoid any noticeable delays for system users (Lynn, 2007).

3.2 Linear Secret Sharing Scheme

In CP-ABE, the access policy for encrypted data is defined using Linear Secret Sharing Scheme (LSSS). LSSS is a scheme that divides a secret into multiple shares, allowing the original secret to be reconstructed using a specific combination of shares. In the context of CP-ABE, the access policy is presented as a Boolean expression of attributes. LSSS is employed to convert this expression into a set of access structures, each associated with a set of shares. These shares are then used to encrypt the data, such that a user possessing the shares associated with an access structure can decrypt the data. The use of LSSS in CP-ABE enables fine-grained access control over encrypted data, allowing data owners to specify complex access policies based on a user's attributes. Specifically, the access policy is defined using a policy matrix, where each row in the matrix is mapped to an attribute using a function denoted by (Y. Zhang et al., 2018). In this scheme, the presence of an attribute universe was denoted as AU, which has n categories of attributes. Below is the LSSS concept:

Definition 1 (LSSS): Let $AU = (Att_1, Att_2, Att_3, \dots, Att_n)$, where each attribute Att_n consists of an attribute name and attribute values. The attribute values can be represented as $AVx = \{\xi_{x,1}, \xi_{x,2}, \xi_{x,3}, \dots, \xi_{x,nx}\}$. Additionally, let $\boldsymbol{A} = \frac{l \text{ x } n}{\mathbb{Z}p}$ denote the share-generating matrix, where each row in A corresponds to a mapping of an attribute name index denoted as ρ. LSSS comprises the following two algorithms:

Secret share: The secret shared, $s \ni \mathbb{Z}p$ and the value λ_x is computed for each row A_x of $\boldsymbol{A}$, where $V = (s, y_2, y_3, \dots, y_{n)} - \in R\mathbb{Z}_p^n$ and $y_2, y_3, \dots, y_n$ are chosen randomly from $\mathbb{Z}p$. Hence, the secret share value is given $\lambda_x = A_x \times$ v.

Secret Construction: This algorithm takes as input the secret shares $\{\lambda_x\}$ and a set P containing the authorized attribute name index. It sets $I = \{x|\rho(x) \in P\} \subseteq \{1, 2, \dots, l\}$, where $\rho(x)$ denotes the mapping of attribute name index x. The algorithm futher computes the constant $\{\omega_x\}_{x \in I}$ such that $\sum_{x \in I} \omega_x A_x = (1, 0, 0, \dots, 0)$. Finally, the secret s is reconstructed by $s = \sum_{x \in I} \omega_x \lambda_x$.

Similar to the approach proposed by L. Zhang et al., (2019), our scheme utilizes LSSS matrices constructed over $\mathbb{Z}_N$. In our proposed scheme, the user's attribute is denoted as $S = (B_s, J_s)$, where $B_s, \subseteq \mathbb{Z}_N$ representing the attribute name index, and $J_s = \{l_x, i\}_x \subseteq B_s$ represent the attribute value set. We denote the access policy $\mathbb{A} = (\boldsymbol{A}, \rho, \mathcal{T})$ where $\mathcal{T}$ represent the attribute value for each row of A. Specifically, $\mathcal{T} = \left(t_{\rho(1)}, t_{\rho(2)}, t_{\rho(3)}, \dots, t_{\rho(l)}\right)\left(t_{\rho(x)} \in AV_{\rho(x)}\right)$. For S satisfies $\mathbb{A}$, there a must exits a $I \subseteq \{1, 2, \dots, l\}$ satisfying $(A, \rho), \{\rho(x)|x \in I\} \subseteq B_s$ and $l_{\rho(x)} = t_{\rho(x)} \forall x \in I$.

4. The System Model and Security Assumption of Ciphertext Policy Attribute based Encryption

In this section,we present the system model, and security requirement and assumption for CP-ABE proposed in this paper.

4.1 System Model

The system model as in Figure 3 includes the following entities: Data Owner, Data User, Attribute Authority and Cloud Service Provider.

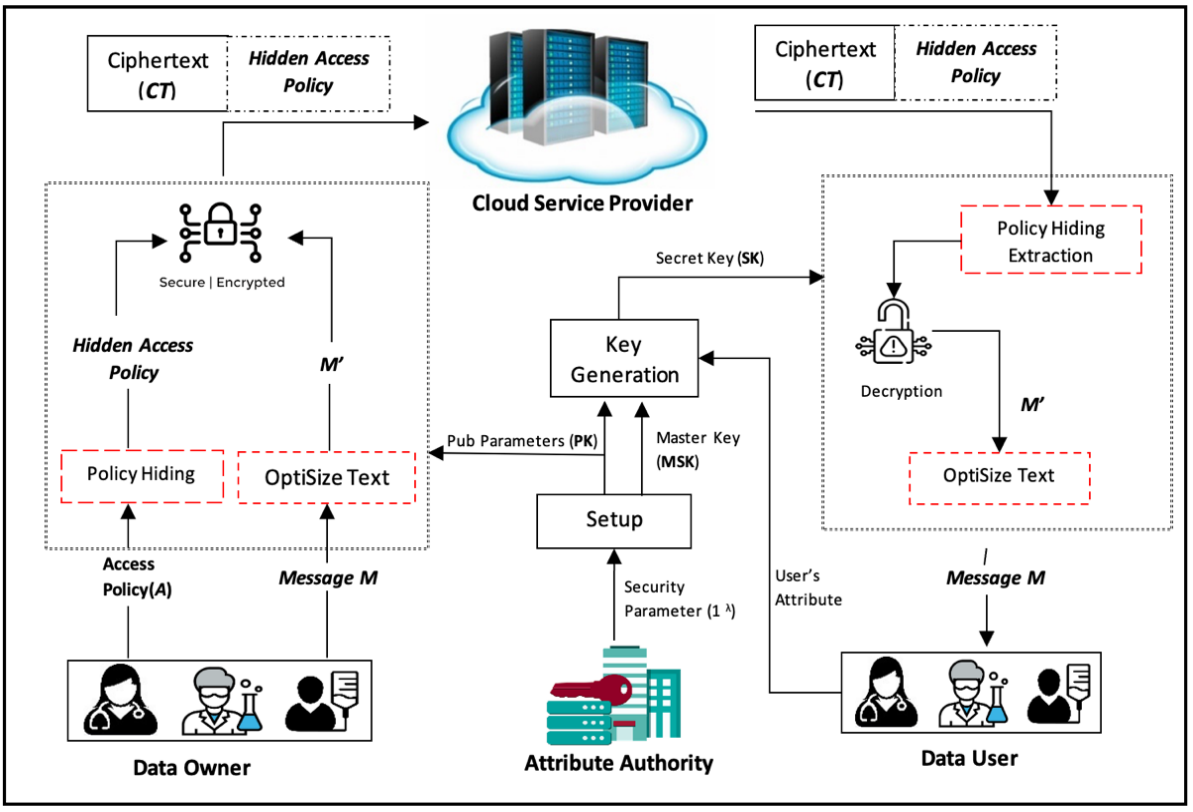


Figure 3. System Model

4.1.1 Data Owners

The Data Owner is responsible for uploading files into the system for encryption. Before encryption, the files undergo an optimization process (OptiSize) to minimize their size, reducing storage and transmission overhead. The ciphertext and the access policy, defined by the Data Owner, are then stored in the Cloud for controlled access.

4.1.2 Data User

A Data User requests access to the encrypted data stored on the Cloud. Each user possesses a secret key, issued by the Attribute Authority (AA), containing their attributes. Decryption is permitted only if the user's attributes satisfy the access policy defined by the Data Owner. Otherwise, the ciphertext remains inaccessible.

4.1.3 Attribute Authority

The Attribute Authority (AA) is responsible for issuing encryption and decryption credentials (secret keys) to both Data Owners and Data Users. The Data Owner submits a set of attributes to the AA, which then generates a secret key for encrypting the data. The ciphertext, along with the access policy, is stored in the Cloud after encryption.

For decryption, Data Users must present their attributes to the AA for verification. If the attributes match the policy set by the Data Owner, the AA provides the necessary decryption credentials, enabling data access.

4.1.4 Cloud Server

The Cloud offers scalable storage and facilitates data sharing among authorized users. However, as an untrusted third-party service provider, it introduces security risks, including potential unauthorized access or data exploitation. This necessitates robust encryption mechanisms to ensure data confidentiality and integrity.

4.2 Security Requirement and Assumption

Within this section, we delve into the security requirements and underlying assumptions of the CP-ABE scheme. Our exploration encompasses a comprehensive discussion of various assumptions and security needs. Additionally, we provide an in-depth examination of the proposed Access Policy Hiding (APH) in CP-ABE, emphasizing its security model and analysis

- Security Requirement of the proposed CP-ABE are explained below:

Confidentiality: To achieve this goal, measures are implemented to prevent unauthorized users from accessing the encrypted data, allowing only those who meet the access policy criteria to utilize the encryption module. Furthermore, data confidentiality is maintained by implementing safeguards that prohibit entities, including CSPs, from gaining access to or reading any information from the encrypted data.

Fine-grained access control : The privilege of retrieving data on the Cloud is not uniformly granted to all users. Instead, it is contingent upon their level of involvement or responsibility. Consequently, in this study's security solution, users are assigned distinct access privileges that align with the access policy defined by the Attribute Authority (AA). As a result, all attributes must align with the user's access policy structure in order to access the required information.

- There are different assumptions for the proposed CP-ABE scheme, as explained below:

1.The Cloud server is primarily responsible for executing assigned tasks; however, it possesses an inherent curiosity regarding confidential information associated with the data and its users. However, the system incorporates cryptographic access control mechanisms that enforce access policies directly within the ciphertext. This ensures that the Cloud server is prevented from intervening or accessing the data, maintaining the integrity and privacy of the system.

2. The Attribute Authority (AA) is regarded as a semi-trusted entity, meaning there is a level of trust placed in its operations. However, it is important to acknowledge that the AA can be targeted by adversaries. The AA acts as a key generation center, responsible for generating users' secret keys. Additionally, the AA plays a crucial role in authenticating the attributes of users before granting them access, ensuring that only authorized individuals are allowed entry.

3. While accessing and storing encrypted data on a Cloud server system is open to all, the capability to successfully decrypt the corresponding ciphertexts is restricted to users who possess attributes that comply with the access policy and are not included in the revocation list. It is crucial to acknowledge that the system takes into account the potential for user misconduct, as well as collusion among entities within the system or between users, excluding the data owner, in order to illicitly obtain unauthorized access to the data.

4. The owner of the data is a trusted entity prevent malicious entities from detecting individual users.

5. Ciphertext Policy Attribute based Encryption

The traditional CP-ABE scheme consists of four algorithms: setup, key generation, encryption, and decryption. However, as discussed earlier, existing CP-ABE schemes suffer from privacy leakage and performance issues. In order to address these issues, this paper proposes several enhancements to the existing CP-ABE scheme. These enhancements include access policy hiding to prevent privacy leakage and OptiSize Text, which aims to optimize the ciphertext size for improved efficiency. Figure 4 provides an overview of the proposed algorithm. Section 5.1 focuses on the details of the proposed access policy hiding, while Section 5.2 delves into OptiSize and its role in optimizing the ciphertext size.

1. Setup

The Attribute Authority (AA) begins the setup process by executing the setup algorithm with a security parameter of 1^{λ}. This algorithm generates a tuple $N = (p_1p_2p_3p_4; \mathbb{G}; \mathbb{G}T; e)$ which consists of four prime numbers, p_1, p_2, p_3, p_4. Additionally, based on these prime numbers, four distinct ordered subgroups denoted as $\mathbb{G}p_1$, $\mathbb{G}p_2$, $\mathbb{G}p_3, \mathbb{G}p_4$ are constructed. Let $\mathbb{G}$ and $\mathbb{G}T$; be a cyclic group with order N. The Attribute Authority then uniformly selects $a, \alpha, \alpha 1, \beta \in R$ $\mathbb{Z}_N$ and $g, g1 \in \mathbb{G}p$. Two publics hash functions, H and H_1, are set up with H mapped the attribute value, $AV_x$to an element in $\mathbb{Z}_N$, while H_1 was a pseudo-random function that maps elements in $\mathbb{G}$ and $\mathcal{M}$ to elements in $\mathcal{M}$. The bilinear map e is computed, resulting in the values Y, Public Parameters PK, and Master Key MSK. These values are determined as part of the setup process.

$$Y = e(g, g_1)^{\alpha\alpha 1},$$

$$PK = \{N, g, g^a, g^{\alpha 1}, g^{\beta}, Y\},$$

$$MSK = \{a, \alpha, \alpha_1, \beta, g_1\}$$

2. $Keygen$ $(PK, MSK, S) \rightarrow SK$

To maintain the integrity of the file system and prevent unauthorized access, the Attribute Authority (AA) performs verification to ensure the legitimacy of users. In order to generate a secret key (SK) that grants access to authorized individuals, the AA employs the KeyGen module. This module takes several input parameters, including public parametersPK, master key MSK, users' attributes $S = (B_s, J_s)$. Given B_s represents the attribute name index set and J_s denotes the attribute value set for the user. The AA executes the KeyGen module as outlined below:

AA chose $t \in R\mathbb{Z}_N$ and $R, R_1, R_i \in R\ \mathbb{G}p_3$ for $i \in B_s$. It computed K_1, K_2, K_i as

$$K_1 = g_1^{\alpha} {g_1}^{dt}.R,$$

$$K_2 = {g_1}^{t\alpha 1}.R_1,$$

$$K_i = ({g_1}^{H(L_i)} {g_1}^{\beta})^t.R_i.$$

Then, the secret keys associated with attribute set $S = (B_s, J_s)$ were calculated as

$$SK = (K_1, K_2, \{K_i\}_{i \in B_s}.$$

3. *Access Policy Hiding*

The module takes the access structure $\mathbb{A} = ((A, \rho)\mathcal{T}))$ input from the data owner. The module extracts the location $(-x, -y)$ information from the access matrix A of the policy. The $(-x, -y)$ location is then obfuscated or concealed within another value before being embedded together with the ciphertext. A comprehensive discussion of this scheme will be presented in Section 5.1, providing in-depth insights into its workings and components.

4. OptiSize TextThe

OptiSize module accepts file inputs m from the data owner. It analyzes the file contents and scans for any data redundancy or inconsistencies. In order to eliminate discrepancies, redundant data within the file is removed and it will produce m'. Section 5.2 will provide a thorough analysis of this scheme, offering a comprehensive exploration of its mechanisms.

5.Encrypt $(PK, M, \mathbb{A}) \rightarrow CT$

In this module, we provide the following inputs: public parameter PK, Message M and access structure $\mathbb{A} = ((A, \rho)\mathcal{T})$. These inputs are utilized to generate the ciphertext, CT. The access structure $\mathbb{A}$ comprises an access matrix A with dimension $l \times n$ where each row A_x is mapped by ρ to an attribute name index. Additionally, $\mathcal{T} = (t_{\rho(1)}, t_{\rho(2)}, t_{\rho(3)}, \dots, t_{\rho(l)}) \in \mathbb{Z}_N^l (t_{\rho(x)} \in AV_{\rho(x)})$ represent a set of attribute-value related to the access policy (A, ρ). The encryption algorithm proceeds by selecting a random vector $V = (s, y_2, y_3, \dots, y_n)$ where $s, y_2, y_3, \dots, y_n$ are chosen randomly from $\mathbb{Z}_N$ with s is a shared value. For $x = 1$ to l, it computed $\lambda_x = A_x \times V$, where A_x corresponded to the x^{th} row of A and calculated $X = \mathcal{E}_{Enc}(k, M'), \mathcal{F} = H_1(k \parallel M')$. Additionally, it also randomly took $Q_0, \{Q_x\}_{1 \le x \le 1 \in R} \mathbb{G}_{p4}$. Finally, it calculated the entire ciphertext components $C_0, C_1\{C_x\}_{1 \le x \le 1}$ as follows:

$$C_0 = k e(g, g_1)^{\alpha\alpha 1 s},$$

$$C_1 = g^{s\alpha 1}. Q_0,$$

$$C_x = g^{a\lambda x}\left(g^{H(t\rho(x))} g^{\beta}\right)s. Q_x.$$

Decrypt $(PK, SK, CT, S) \rightarrow M$

Similar to the approach used by L. Zhang et al. (2019), the decryption algorithm begins by verifying whether the hash value of $H(J_s) = H(t_{\rho(x)})$. If these value match, the system authorized the Data User (DU) to decrypt the CT using following steps:

$$E = e(g, g_1)^{\alpha\alpha_1 s}$$

$$k = C_o \,/\, E$$

5.1 Access Policy Hiding (APH)

In existing CP-ABE, the access policy $((A, \rho), \mathcal{T})$ is appended to the ciphertext CT then stored into the Cloud storage. However, this readable access policy format poses a risk of exposing sensitive user information. To address this concern, researchers in Ying et al., (2018) have emphasized the potential attribute leakage caused by the attribute mapping function $\boldsymbol{\rho}$. Therefore, the APH scheme has been enhanced to mitigate privacy leakage by eliminating the attribute mapping function. In this improved scheme, attribute value in access structure $\mathbb{A} = ((A, \rho)\mathcal{T}))$ is replaced with attribute location in the form of (x, y). Additionally, XOR-based logical connectives are employed in the policy hiding strategy to enhance the privacy of the access policy. Within the APH scheme, XOR-based logical connection conceals location attributes by transforming them into meaningless location values. This approach enhances the robustness of the APH scheme since even if an attacker intercepts the access policy, they cannot determine the precise location of the attribute values in the access matrix.

5.1.1 Policy Hiding $((A, \rho), \mathcal{T}) \rightarrow$

HVDuring this process, the Policy Hiding algorithm aims to derive the hidden value HV, which represents the location of the encrypted attribute value. The algorithm takes an access policy $((A, \rho), \mathcal{T})$ as input. First, it extracts the set of attribute values associated with the access policy $(\mathcal{T})$ and determines the exact location (x, y) of each attribute value. These locations are then transformed into ciphertext using operations such as $\oplus$ and $\odot$. As a result, the Policy Hiding solution generates the output of ciphertext and hidden value (CT, HV) which is subsequently outsourced to Cloud servers. The algorithm of the Policy Hiding process is illustrated in Figure 4.

Algorithm 4.1: Hiding Access Policy

INPUT Access policy $((A, \rho), \mathcal{T})$

OUTPUT Hidden Access Policy (HV)

BEGIN

1. // Access policy contains all the attribute values chosen by DO

1. //AV referring to Attribute Value

3. // ($\mathcal{T}$) - attribute value

4. **foreach** AV_x of $\mathcal{T}$, $\forall_x =$ index of AV

5. Extract location (X_x, Y_x)

6. //location attribute value in access matrix

7. Convert location (X_x, Y_x)String to Binary

8. Compute $\alpha_x = X_x \oplus Y_x$

9. Compute $\beta_x = \alpha_x \odot Y_x$

10. Convert Binary to Hexadecimal for each β_x, store as = HV_x

11. // HV_x will be bind with CT

12. **End the process**

5.1.2 $ExtHiddenPolicy\ CT, HV) \rightarrow (\mathcal{T})$

In the decryption process, the hidden value, HV retrieved from the Cloud storage is utilized as an input. HV is first converted into a binary set, and then operations such as $\oplus$ and $\odot$ are applied to obtain the original location of the attribute. The details of this process are described in Algorithm 2.

Algorithm 2: Extracting Policy Hiding Logical Connective

INPUT Hidden Access Policy (HV)

OUTPUT Access policy $((A, \rho), \mathcal{T})$

BEGIN

1. // HV referring to hidden values of access policy

2. **foreach** HV_x , $\forall_x =$ index of HV

3. Convert hexadecimal HV_x to binary; store as (X_x, Y_x)

4. Compute $\delta_x = X_x \oplus Y_x$

5. Compute $\Omega_x = \delta_x \odot Y_x$

6. Convert $\Omega_x$into Unicode Text

7. **End** the process

5.2 OptiSiz

Ciphertext-Policy Attribute-Based Encryption (CP-ABE) provides fine-grained access control, yet its scalability is hindered by the expanding ciphertext size, which directly correlates with the number of attributes in the access policy. As access policies grow, ciphertexts become increasingly large and inefficient, imposing high storage costs and performance bottlenecks in cloud environments. Prior research, including work by C. Jin et al. (2016b) and L. Zhang et al. (2019), has attempted to mitigate this issue by employing data compression techniques to remove redundant attributes. However, since redundancy in access policies is inherently low, these methods yield minimal reductions in ciphertext size. Additionally, data inconsistencies in encrypted files further exacerbate storage inefficiencies and increase transmission overhead, limiting the practicality of CP-ABE in large-scale applications.

To overcome these limitations, this research introduces OptiSize, an optimization scheme designed to reduce ciphertext size while maintaining CP-ABE's security guarantees. Unlike conventional compression approaches, OptiSize eliminates redundant data directly from the encrypted files, thereby reducing overall ciphertext size beyond just access policy simplifications. By integrating a dual optimization approach, OptiSize effectively minimizes storage overhead and improves transmission efficiency, making encrypted data more manageable in cloud environments. The scheme ensures that encrypted files remain lightweight, reducing the computational burden on cloud storage while optimizing data-sharing processes.

The key contribution of this work is the development of a lightweight yet effective enhancement to CP-ABE that significantly improves its scalability and efficiency. OptiSize reduces ciphertext expansion, lowers cloud storage costs, and enhances transmission efficiency without compromising security. By balancing performance optimization and robust encryption, this approach addresses a critical gap in CP-ABE research, ensuring practical deployment in real-world cloud systems. The proposed solution outperforms existing methods by targeting not only access policy redundancy but also structural inefficiencies in encrypted files, making it a transformative advancement in secure cloud-based data sharing.

5.2.1 OptiSize Component

In the OptiSize, each word in a file is treated as a separate block. Each block is compared with the other block to find similarities. When this resemblance occurs, it indicates that the file contains repetitive words, which will be removed. There are several components in OptiSize as in Figure 5. This component consists of tokenizer, token identifier, filtering and concatenate identifier.

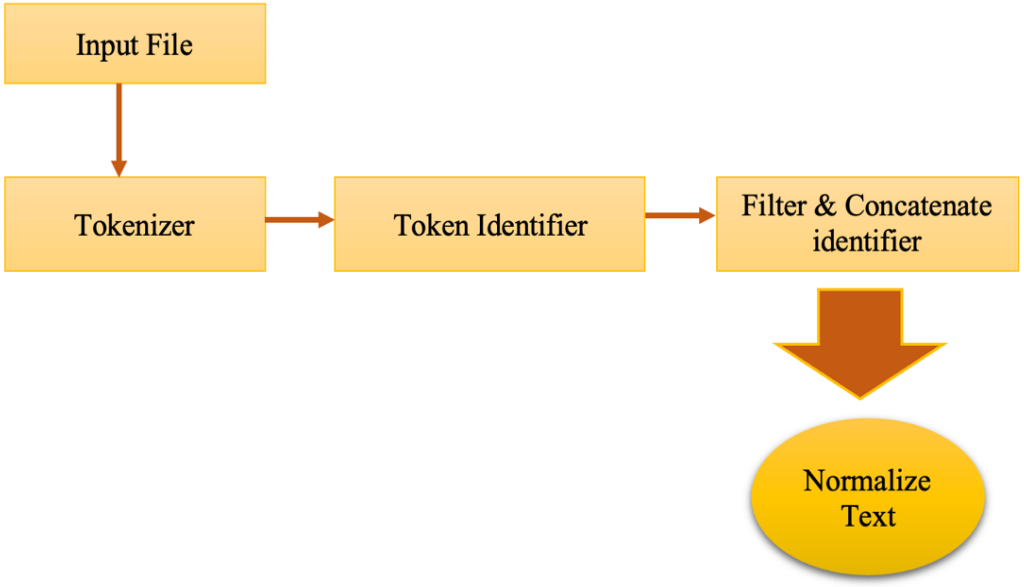


Figure 5. OptiSize Components

- Input File

For process of breaking down a text document into individual words or tokens, an input file must contain only textual data. The format of the text document does not necessarily have to be a text file, but it should be a format that can be read and processed by proposed technique.

- Tokenization

Tokenization is the process of breaking a piece of text into a series of meaningful blocks or chunks. The main function of this component is to identify and separate words in a file using tokens. In this process, each individual word as well as each punctuation mark, numbers with decimals or dates in the format of numbers, periods, commas or slashes or any element present in the file will be a different token. In the tokenization process, the white space in the text will be used as the word "delimiter". In addition, this white space will also be assigned a token and will be used in the formation of the text file during the decryption process.

- Token Identifier

After each element in the text file has been tokenized, a process of determining the identifier for each token is performed. The identifier for each token will be provided in a sequential manner which refers to the position of each token in the file or message. Each token with their respective identifier is sorted lexicographically. Lexicographically means each word with index number totally ordered set. Then, the token has been sorted which tokens that have the same word/element are gathered with its identifier.

- Filtering & Concatenate Identifiers

Filtering and concatenate identifiers is the last component in OptiSize Text. In this component, the token together with the identifier that were gathered in the preceding component will be filtered. In this filtering process, tokens that have the same word will be dropped and only single word tokens are stored. The discarded token's identifier will then be combined with the single token that was previously saved. Afterward, OptiSize Text technique will save token and its identifier as a file and the file will then go through an encryption process before being saved to the Cloud storage.

5.2.2 OptiSize Construction

In OptiSize scheme, the data owner uploaded a file to be encrypted and stored in cloud storage. Prior to that, the file underwent the OptiSize to eliminate extraneous data and reduce the file size. Based on the flowchart as in Figure 6, Data Owner submitted a text file to be shared with data users who satisfy the attribute value requirements in the access policy. The OptiSize Text is applied to the text file before encryption to remove duplicate data. This procedure is significant as if the file size increases, it will have an impact on the ciphertext's size and the efficiency of the CP-ABE scheme. After the file has been uploaded, text cleaning is carried out to add a white space before any punctuation or symbols that were identified in the text. As a result, in the tokenization proses, any element discovered before white space will be tokenized by the tokenizer.

After the completion of the tokenization process, OptiSize performs the process of assigning an identifier to each token. When two tokens contain the same word, their identifiers will be concatenated. Therefore, the file containing the identifier and single token for same word will be encrypted and stored in the cloud. Meanwhile, when the data user wants to decrypt the file, each concatenated identifier is reused to build the original text in the file.

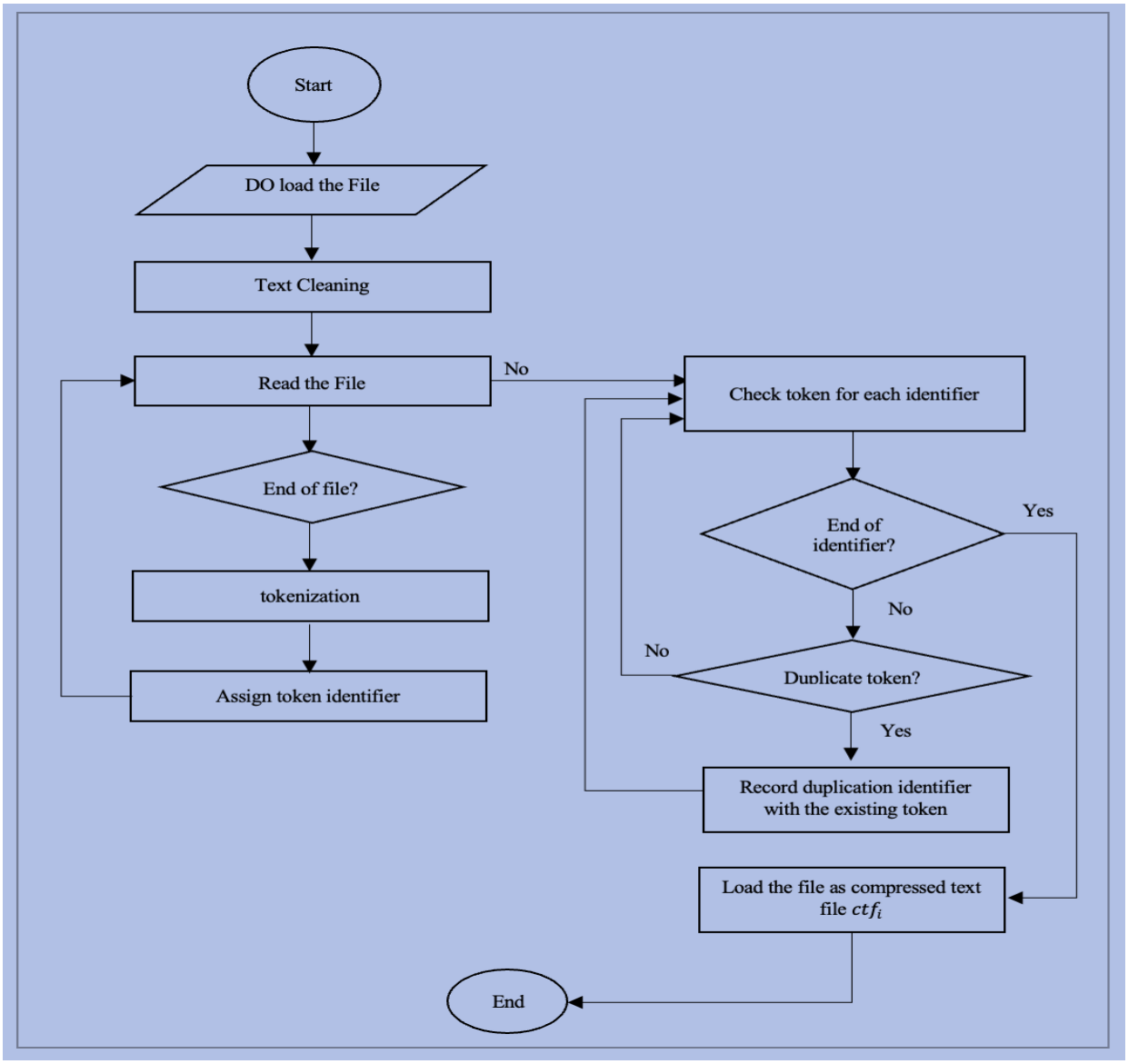


Figure 6. OptiSize Text Flowchart

The detailed steps of the OptiSize Text algorithm are presented below.

Algorithm 3: OptiSize Text

Input: file f_i

Output: compressed text file cf_i

//Data Owner load f_i

char sym ={!, @, #, $, %, ^, &, *, (,), _, -, +, {, }, [,], ;, :, ', ",<, >, /, ?, \, |, `, ~, ., ,}

array token identifier [i]

```
6     foreach f_i
7       Search sym in f_i
8       Add whitespace before sym
9       Read f_i
10      if f_i≠ eof
11        Do word tokenize
12        Assign token identifier [i]
13        else
14        foreach token identifier[i], check token
15          if token ≠ eof
16            if token= duplicate
17              Record duplication token identifier with the
                existing token
18            else
19              check duplication token
20            end if
21          else
22            Load the file as compressed text file cf_i
23          end if
24          end for
25        end if
26    end for
```

6. Experiment Configuration

The experiment is carried out on a machine with 3.40 GHz Intel ® Core ™ i3-4130 CPU, a running speed of 4.0 GB RAM on Windows 10 operating system, and an Eclipse IDE for Java developer 2019-03 to analyse the performance of construction. Moreover, numerous experiments were performed with the increasing user attributes that vary between 2 and 14. During the experiment, we first test the performance of the proposed scheme by the time cost. Then, we also analyse the performance by evaluating the storage cost. Finally, we analyse the advantages of proposed scheme by comparison with other works.

7. Result and Discussion

The Access Policy Hiding (APH) evaluation focuses on two key performance metrics: encryption time and ciphertext size. Meanwhile, for OptiSize, the performance metrics studied are ciphertext size and computation cost. To evaluate the effectiveness of the proposed scheme, a benchmark was established based on related research conducted by Zhang et al. (2019). Zhang's work shares the same objective of ensuring user data and privacy in the Cloud while striving for optimal performance. Like our proposed scheme, Zhang's research uses CP-ABE with a hidden policy and implements a Linear Secret Sharing Scheme (LSSS) for the access structure. Additionally, both approaches use a single-attribute authority architecture. By comparing the performance of the proposed scheme with the benchmark set by Zhang's work, the effectiveness and efficiency of APH and OptiSize text can be accurately evaluated.

The APH scheme involves the encryption process, which includes both access policy hiding and data encryption. In the experiment, a 10KB file size was used as input, with varying numbers of attributes in the access policy. The experiment compared the performance of the APH scheme with a Zhang, as shown in Figure 7.

Based on Figure 7, the encryption time in milliseconds (y-axis) against the number of attributes in the access policy (x-axis). The bar chart demonstrates that the APH scheme achieved lower encryption times compared to the work by L. Zhang et al. (2019). For the number of attributes ranging from 2 to 14, the encryption time gradually increased for both schemes. However, the APH scheme consistently outperformed the Zhang work, exhibiting an average improvement of 10.6% in encryption time. Notably, for the number of attributes 12 to 14, the APH scheme achieved an average decrease in encryption time of 35.7% compared to the benchmark work, indicating higher efficiency in terms of encryption time.

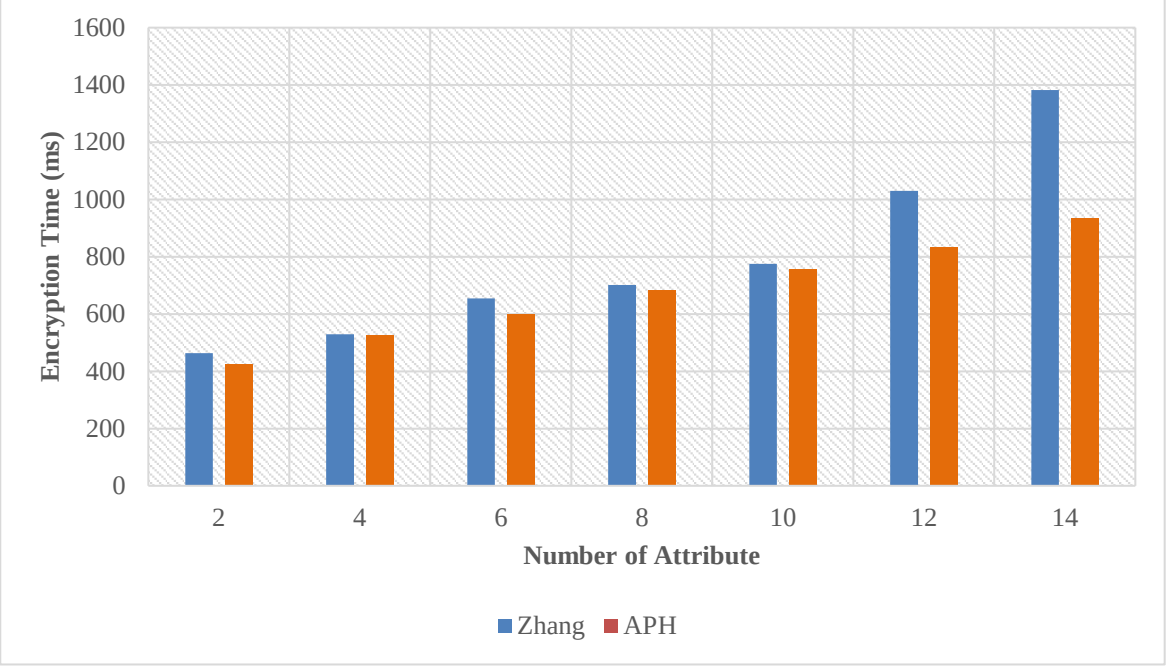


Figure 7. Encryption Time

Figure 8 illustrates the encryption time comparison between the APH scheme and the Zhang for files of different sizes. The experiment was conducted using two attributes in the access policy and file sizes ranging from 200Kb to 1000Kb. The graph reveals a similar pattern in the encryption time for both schemes. However, the APH scheme demonstrates an average improvement of 9.6% in encryption time compared to the benchmark algorithm. Although the improvement percentage may not be significant, APH achieves a noteworthy average encryption time advantage over the benchmark work without compromising the overall encryption process.

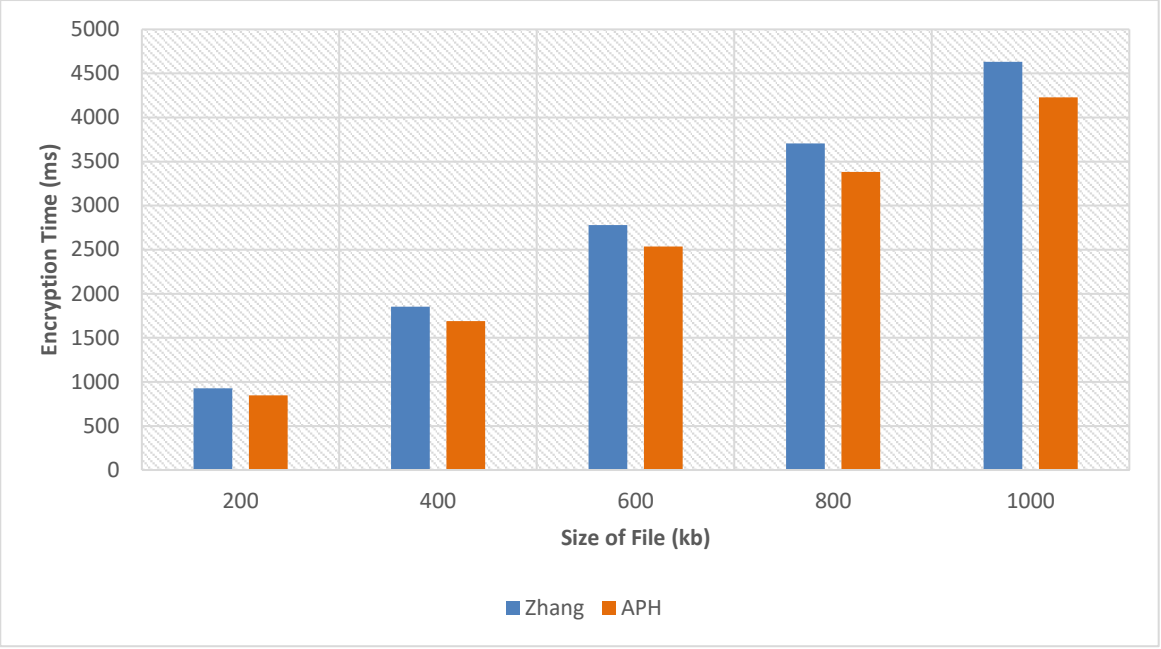
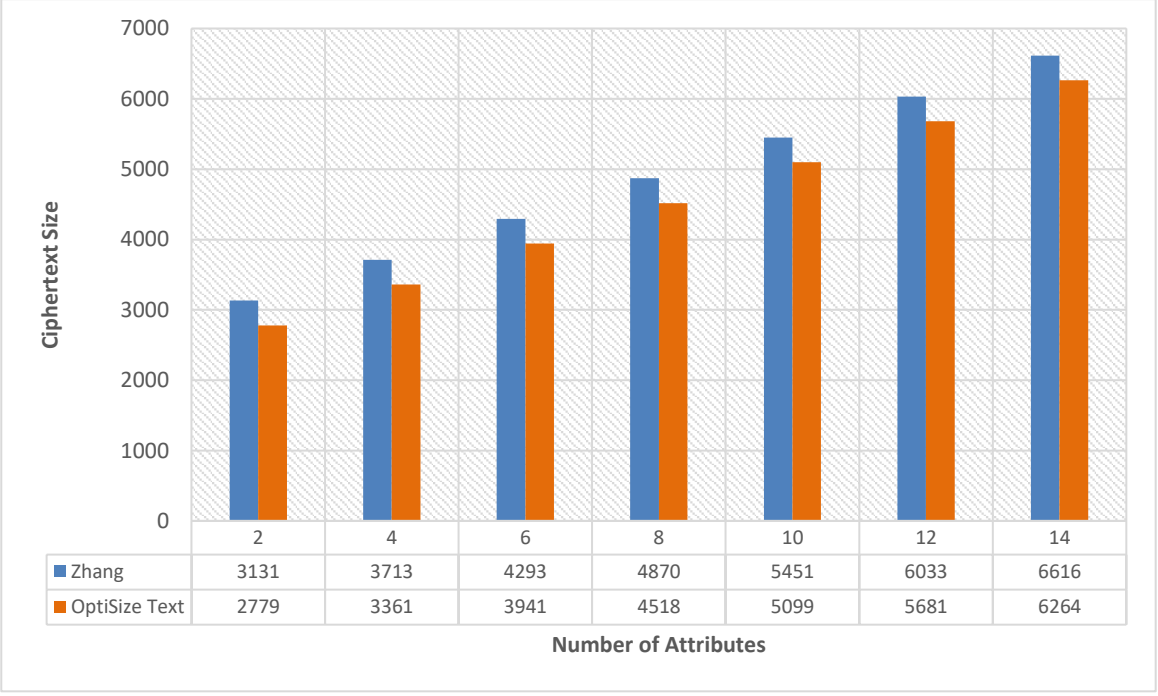


Figure 8. Encryption Time for Different Size File

The graph presented in Figure 9 demonstrates a significant reduction in ciphertext size achieved by the proposed OptiSize scheme. An experiment was conducted to compare the ciphertext size between the Zhang and OptiSize scheme. The experiment utilized a fixed file size of 3200Kb and varied the number of attributes. According to existing study, the number of attributes directly impacts the file size. From the graph, it is evident that the file size increases proportionally with the number of attributes. However, the proposed OptiSize scheme solution consistently achieved a smaller ciphertext size compared to the Zhang for every number of attributes by demonstrating a 5% reduction in size.



	2	4	6	8	10	12	14
Zhang	3131	3713	4293	4870	5451	6033	6616
OptiSize Text	2779	3361	3941	4518	5099	5681	6264

Figure 9. Ciphertext Size

The computation cost in this experiment is determined by measuring the time required to execute the OptiSize scheme. The computation time includes various detailed processes such as text cleaning, tokenization, assigning identifiers, and filtering and concatenating identifiers. The experiment aimed to compare the computation cost between the OptiSize scheme approach and the Zhang work.

Figure 10 displays the number of access policy attributes on the x-axis and the computation time on the y-axis. According to the graph, as the number of attributes in the access policy increases, the computational cost also increases. However, the OptiSize scheme exhibits lower computational cost compared to the benchmark work. This indicates that even with the addition of a new phase to the CP-ABE method, the OptiSize scheme offers efficient computation cost.

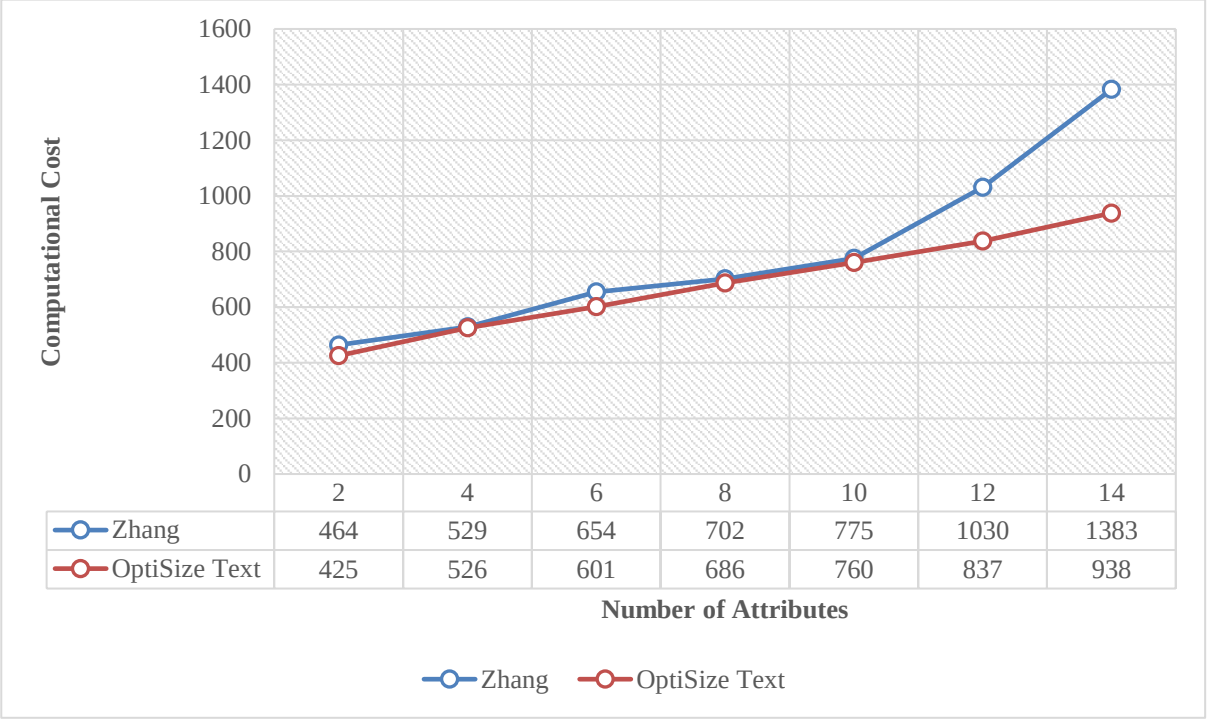


Figure 10. Computational Cost for OptiSize scheme

8. Conclusion

This study addresses critical challenges in securing confidential data within an untrusted cloud environment by proposing a modified Ciphertext-Policy Attribute-Based Encryption (CP-ABE) scheme with fine-grained access control. Existing CP-ABE implementations suffer from data privacy leakage and expanding ciphertext size, leading to computational inefficiencies and security vulnerabilities as the number of attributes in the access policy increases. To mitigate these issues, we introduce an enhanced CP-ABE framework that integrates a two-layer policy concealment approach and text size optimization to ensure stronger privacy protection, reduced ciphertext expansion, and lower storage overhead. These improvements contribute to enhancing data security, optimizing cloud storage utilization, and increasing the feasibility of CP-ABE in large-scale environments.

Despite its advancements, the proposed scheme has certain limitations. While ciphertext size reduction and privacy enhancements are achieved, the model still incurs communication overhead during data transmission to the cloud, which could impact system efficiency in highly dynamic environments. Additionally, the current scheme is designed for static access policies, limiting its adaptability in scenarios requiring frequent attribute updates. These constraints highlight the need for further refinements to enhance the scalability and real-time efficiency of CP-ABE in practical cloud applications.

For future research, we aim to explore strategies for minimizing communication overhead during data transmission and enhancing computational efficiency to further reduce processing delays. Additionally, incorporating dynamic attributes into CP-ABE will improve real-time adaptability, ensuring that access control mechanisms remain flexible and responsive to evolving security requirements. By addressing these challenges, CP-ABE can be further refined to offer a highly efficient, scalable, and privacy-preserving encryption framework suitable for real-world cloud environments.

Corresponding author

Masnida Hussin
masnida@upm.edu.my

Acknowledgements
This research was funded by a grant from Universiti Sultan Zainal Abidin through Center for Research Excellence and Incubation Management (CREIM) (UniSZA/2023/DPU1.0/07).

Funding
No funding.

Contributions
S.D.M.S; M.H; M.A.M; N.A.H; M.F.A.K; R.M; J.S; Conceptualization, S.D.M.S; M.H; M.A.M; N.A.H; M.F.A.K; R.M; J.S; Investigation, S.D.M.S; M.H; M.A.M; N.A.H; M.F.A.K; R.M; J.S; Writing (Original Draft), S.D.M.S; M.H; M.A.M; N.A.H; M.F.A.K; R.M; J.S; Writing (Review and Editing) Supervision, S.D.M.S; M.H; M.A.M; N.A.H; M.F.A.K; R.M; J.S; Project Administration.

Ethics declarations
This article does not contain any studies with human participants or animals performed by any of the authors.

Consent for publication
Not applicable.

Competing interests
All authors declare no competing interests.

References

[1] Bethencourt, J., & Waters, B. (2007). Ciphertext-Policy Attribute-Based Encryption. IEEE Computer Society.
[2] Bethencourt, J., Waters, B., Sahai, A., & Waters, B. (2007). Ciphertext-Policy Attribute-Based Encryption. *2007 IEEE Symposium on Security and Privacy (SP '07)*, 321–334. https://doi.org/10.1109/SP.2007.11
[3] Cui, H., Deng, R. H., & Li, Y. (2018). Attribute-based cloud storage with secure provenance over encrypted data. *Future Generation Computer Systems*, *79*, 461–472. https://doi.org/10.1016/j.future.2017.10.010
[4] Edemacu, K., Jang, B., & Kim, J. W. (2020). Efficient and Expressive Access Control with Revocation for Privacy of PHR Based on OBDD Access Structure. *IEEE Access*, *8*, 18546–18557. https://doi.org/10.1109/ACCESS.2020.2968078
[5] Goyal, V., Pandey, O., Sahai, A., & Waters, B. (2006). Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. *CCS '06: Proceedings of the 13th ACM Conference on Computer and Communications Security*, 89–98.
[6] Hao, J., Huang, C., Ni, J., Rong, H., Xian, M., & Shen, X. (Sherman). (2019). Fine-grained data access control with attribute-hiding policy for cloud-based IoT. *Computer Networks*, *153*, 1–10. https://doi.org/10.1016/j.comnet.2019.02.008
[7] Kahani, N., Elgazzar, K., & Cordy, J. R. (2016). Authentication and Access Control in e-Health Systems in the Cloud. *2016 IEEE 2nd International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing (HPSC), and IEEE International Conference on Intelligent Data and Security (IDS)*, 13–23. https://doi.org/10.1109/BigDataSecurity-HPSC-IDS.2016.43
[8] Khuntia, S., & Kumar, P. S. (2018). New Hidden Policy CP-ABE for Big Data Access Control with Privacy-preserving Policy in Cloud Computing. *2018 9th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2018*, 1–7. https://doi.org/10.1109/ICCCNT.2018.8493698
[9] Li, H., Li, J., Zhang, Y., Chen, X., You, I., & Wong, D. S. (2017). Ensuring attribute privacy protection and fast decryption for outsourced data security in mobile cloud computing. *Information Sciences*, *379*, 42–61. https://doi.org/10.1016/j.ins.2016.04.015
[10] Lopez, J., & Rubio, J. E. (2018). Access control for cyber-physical systems interconnected to the cloud. *Computer Networks*, *134*, 46–54. https://doi.org/10.1016/j.comnet.2018.01.037

[11] Mohamed, A. K. Y. S., Auer, D., Hofer, D., & Küng, J. (2022). A systematic literature review for authorization and access control: definitions, strategies and models. *International Journal of Web Information Systems*. https://doi.org/10.1108/IJWIS-04-2022-0077
[12] Narasingapuram, P. B., & Ponnavaikko, M. (2021). A Secure Cloud Authentication and Access Control System for Cloud Infrastructure. *IT in Industry*, *9*(2).
[13] P, P. K., P, S. K., & Alphonse, P. J. A. (2018). Attribute based encryption in cloud computing: A survey, gap analysis, and future directions. *Journal of Network and Computer Applications*, *108*(December 2017), 37–52. https://doi.org/10.1016/j.jnca.2018.02.009
[14] Phuong, T. V. X., Yang, G., & Susilo, W. (2016a). Hidden ciphertext policy attribute-based encryption under standard assumptions. *IEEE Transactions on Information Forensics and Security*, *11*(1), 35–45. https://doi.org/10.1109/TIFS.2015.2475723
[15] Ramachandra, G., Iftikhar, M., & Khan, F. A. (2017). A Comprehensive Survey on Security in Cloud Computing. *Procedia Computer Science*, *110*(2012), 465–472. https://doi.org/10.1016/j.procs.2017.06.124
[16] Sabitha, S., & Rajasree, M. S. (2017). Access control based privacy preserving secure data sharing with hidden access policies in cloud. *Journal of Systems Architecture*, *75*, 50–58. https://doi.org/10.1016/j.sysarc.2017.03.002
[17] Susilo, W., Yang, G., Guo, F., & Huang, Q. (2018). Constant-size ciphertexts in threshold attribute-based encryption without dummy attributes. *Information Sciences*, *429*, 349–360. https://doi.org/10.1016/j.ins.2017.11.037
[18] Vijayalakshmi, K., & Jayalakshmi, V. (2021). Shared Access Control Models for Big Data: A Perspective Study and Analysis (pp. 397–410). https://doi.org/10.1007/978-981-15-8443-5_33
[19] Xue, L., Yu, Y., Li, Y., Au, M. H., Du, X., & Yang, B. (2018). Efficient attribute-based encryption with attribute revocation for assured data deletion. *Information Sciences*, *0*, 1–11. https://doi.org/10.1016/j.ins.2018.02.015
[20] Y, J., S, W., M, Y., & G, F. (2016). Ciphertext-Policy Attribute Based Encryption Supporting Access Policy Update. *Provable Security*, *10005*, 39–60. https://doi.org/10.1007/978-3-319-47422-9
[21] Yin, H., Li, Y., Li, F., Deng, H., Zhang, W., & Li, K. (2022). An efficient and access policy-hiding keyword search and data sharing scheme in cloud-assisted IoT. *Journal of Systems Architecture*, *128*. https://doi.org/10.1016/j.sysarc.2022.102533
[22] Younis, Y. A., Kifayat, K., & Merabti, M. (2014). An access control model for cloud computing. *Journal of Information Security and Applications*, *19*(1), 45–60. https://doi.org/10.1016/j.jisa.2014.04.003
[23] Younis, Y. A., Kifayat, K., & Merabti, M. (2016). A novel evaluation criteria to cloud based access control models. *Proceedings - 2015 11th International Conference on Innovations in Information Technology, IIT 2015* (pp. 68–73). https://doi.org/10.1109/INNOVATIONS.2015.7381517
[24] Zhang, L., Cui, Y., Mu, Y., & Member, S. (2020). Improving Security and Privacy Attribute Based Data Sharing in Cloud Computing. *IEEE Systems Journal*, *14*(1), 1–11. https://doi.org/10.1109/JSYST.2019.2911391
[25] Zhang, L., Hu, G., Mu, Y., & Rezaeibagha, F. (2019). Hidden ciphertext policy attribute-based encryption with fast decryption for personal health record system. *IEEE Access*, *7*, 33202–33213. https://doi.org/10.1109/ACCESS.2019.2902040
[26] Zhang, R., Ma, H., & Lu, Y. (2017). Fine-grained access control system based on fully outsourced attribute-based encryption. *Journal of Systems and Software*, *125*, 344–353. https://doi.org/10.1016/j.jss.2016.12.018
[27] Zhang, Y., Zheng, D., & Deng, R. H. (2018). Security and Privacy in Smart Health: Efficient Access Control. *IEEE Internet of Things Journal*, *5*(3), 2130–2145. https://doi.org/10.1109/JIOT.2018.2825289
[28] Zhang, Z., Zhang, W., & Qin, Z. (2021). A partially hidden policy CP-ABE scheme against attribute values guessing attacks with online privacy-protective decryption testing in IoT assisted cloud computing. *Future Generation Computer Systems*, *123*, 181–195. https://doi.org/10.1016/j.future.2021.04.022
[29] Zhao, C., Xu, L., Li, J., Fang, H., & Zhang, Y. (2022). Toward Secure and Privacy-Preserving Cloud Data Sharing: Online/Offline Multiauthority CP-ABE With Hidden Policy. *IEEE Systems Journal*. https://doi.org/10.1109/JSYST.2022.3169601