Dynamic Access Control and Fine-Grained Searchable Encryption for Cloud Data Using Trust Evaluation and B-Tree Indexing

Dapeng Zhao[1], Yapeng Zhao[2], Xuexia Dou[1*]
[1]Henan Polytechnic, Zhengzhou 450046, China
[2]Henan Polytechnic University, Jiaozuo 454003, China
E-mail: zhengzhou16886@163.com
[*]Corresponding author

Keywords: SE, B-tree, cloud servers, user permissions, safety management, trust value, access control

Received: July 2, 2025

The security management of data stored on cloud servers is of great significance, as it not only prevents data leakage but also ensures data integrity. Thus, to achieve secure management of data stored on cloud servers, the research starts with existing searchable encryption technologies and attribute-based searchable encryption technologies, and improved designs are made for both to form a complete encryption scheme. Firstly, in terms of searchable encryption technology, the study considers the dynamic changes in user permissions and constructs an encryption scheme that includes a trust value evaluation model and a dual dynamic access control mechanism. Among them, the trust value evaluation model needs to consider historical trust values, recommended trust values, and attribute related trust values, and then generate a comprehensive trust value to evaluate user permissions. Secondly, in terms of attribute-based searchable encryption technology, the research adopts a multi-branch balanced tree and linear secret sharing scheme to construct a fine-grained access control scheme. Among them, the B-Tree index structure is used to optimize search efficiency, and the linear secret sharing scheme is used to achieve access control for users. In the experimental part, the study uses Python programming language combined with PyCrypto and OpenSSL cryptographic libraries for testing to ensure the feasibility and performance of the solution, and the operating system is Windows 10. The results show that the maximum trust value calculation time considering the user permission scheme is 32 ms, which is 46 ms, 33 ms, 22 ms, and 19 ms lower than the maximum values of the four comparison schemes, respectively. In addition, the maximum access control determination time, CPU utilization, and memory occupancy of this scheme are 56 ms, 12.17%, and 13.95%, respectively. The maximum key generation time and communication volume for supporting fine-grained access control schemes are 180 ms and 3.257 Byte, respectively, and the average storage overhead for user keys and ciphertext is 3.94 KB and 5.37 KB, respectively. The encryption schemes designed by the research have good performance and can provide technical support for secure management of cloud server storage data without decryption for data queries.

Povzetek: Predlagajo dinamičen nadzor dostopa s celovitim zaupanjem ter ABSE z B-drevesnim indeksiranjem in LSSS za fino zrnat, učinkovit iskalni dostop v oblaku.

1 Introduction

As cloud computing technology advances at a rapid pace, cloud servers have become an important platform for data storage and processing. However, data storage in cloud environments faces many security challenges, such as data leakage risks, complex access control permission management, and data privacy protection issues [1-2]. To ensure the security and integrity of data in cloud servers, while meeting the needs of users for efficient data retrieval and access, it holds particular significance in studying an efficient, secure, and fine-grained access control supported data storage management solution. The commonly-used methods for addressing this issue include Searchable Encryption (SE) and Attribute-Based Searchable Encryption (ABSE) [3]. In addition, many scholars have conducted research on SE and ABSE technologies.

Liu et al. designed an SE data sharing scheme based on inverted indexing to address issues such as unreliable data sharing, data attacks, and low efficiency in ciphertext retrieval. This scheme adopted a dual chain structure to store and share data, and constructed an inverted index structure and a ciphertext search algorithm based on this index structure. The results showed that this scheme could ensure datan SEcurity and improve retrieval efficiency [4]. Ng et al. proposed a blockchain-based multi-keyword SE scheme to protect the tracking data of contacts with a certain disease. Meanwhile, the scheme adopted the advanced encryption standard Galois/Counter mode to encrypt data and supported dynamic updates of search indexes. The results showed that the scheme could work effectively without affecting security objectives, and could also maintain efficiency when using larger search indexes [5]. Liu et al. designed an efficient multi-authority ABSE solution assisted by blockchain technology to

address issues such as single attribute authorization failure, privacy leakage during the search process, and high decryption overhead in ABSE technology. This framework leverages consortium blockchain technology for global public parameter management, integrating smart contracts, hybrid online/offline mechanisms, and verifiable edge-assisted decryption protocols to enhance system security and efficiency. The results showed that compared with existing solutions, this approach significantly improved computational efficiency [6]. Lu et al. designed a retrieval strategy ABSE scheme based on key policy attribute encryption framework to provide encryption support for privacy preserving database architectures. This scheme bound security query policies to query credentials and used a policy matrix to optimize the generation of security query policies. The results showed that the scheme achieved semantic security under the selection plaintext attack that included policy and identity queries [7].

Nevertheless, these studies and methodologies are not without their drawbacks, including the inability of SE technology to effectively respond to dynamic changes in user permissions, and the need to improve the search efficiency of ABSE technology. In order to better demonstrate the shortcomings of existing technologies, a detailed comparison of related works was conducted, as shown in Table 1.

Table 1: Summary table of related work

Comparative method	Features	Limitations	Quantitative results
Liu et al. [4]	SE Data Sharing Scheme Based on Inverted Index	Not involving dynamic changes in user permissions, ignoring static access permission issues	Reduce computing costs by 20% and storage expenses by 10%
Ng et al. [5]	Multi keyword SE scheme based on blockchain	Not explicitly mentioning the efficiency of dynamic management of user permissions and access control	Efficiency increased by 30%, computational cost increased by 8%
Liu et al. [6]	Efficient and Multi Authoritative ABSE Scheme Assisted by Blockchain	Unresolved problem of low search efficiency in ABSE technology	40% increase in computational efficiency and 25% increase in storage overhead
Lu et al. [7]	ABSE scheme based on key policy attribute encryption framework	Handling of dynamic changes in user permissions not mentioned	Efficiency increased by 35%, storage overhead increased by 20%

From Table 1, although existing research has made some progress in SE and ABSE technologies, there are still some common limitations. Some solutions did not fully consider the dynamic changes in user permissions, resulting in insufficient flexibility in practical application scenarios. In addition, the search efficiency of ABSE technology still needs to be further improved to meet the needs of large-scale data processing. These shortcomings provide directions for improvement in this study, prompting us to design encryption schemes that are more adaptable and efficient.

Therefore, the research questions are as follows: how to achieve secure storage and efficient management of data in cloud servers, while supporting fine-grained access control; How to dynamically manage user permissions to adapt to the dynamic changes in user permissions; How to improve the search efficiency of ABSE technology. To address the aforementioned research issues, a cryptographic scheme combining SE technology and ABSE technology has been proposed. Specifically, a trust value (TV) evaluation model and a dual dynamic access control mechanism were designed to achieve dynamic management of user permissions and solve the problem of dynamic changes in user permissions. This model evaluates the user's comprehensive TV in real time by comprehensively considering the historical TV, recommended TV and attribute related TV, so as to dynamically adjust user permissions and ensure that only trusted users can access sensitive data. At the same time, the study introduced a Balanced Multi Way Tree (B-Tree) index structure and a Linear Secret Sharing Scheme (LSSS) to optimize the search efficiency of ABSE technology and achieve fine-grained access control. The goal of the research is to design a cloud server data storage management scheme that can adapt to dynamic changes in user permissions by improving SE and ABSE, to achieve secure storage, efficient retrieval, and fine-grained access control of data. The novelty of the present research is embodied in several aspects. It involves the creation of a TV evaluation model, the application of B-Tree to optimize ABSE technology, the attainment of efficient data retrieval capabilities, and the offering of a brand-new technological route for the secure storage and effective management of data on cloud servers.

2 Methods and materials

To address the security management issues of data storage in cloud servers, an SE-oriented user permission dynamic management method was first developed to consider the dynamic changes in user permissions. Meanwhile, the research also designed an ABSE scheme based on B-Tree index structure and combined it with user permission dynamic management method to form a complete encryption method.

2.1 Design of user permission dynamic management method for SE

To manage the security of data stored in cloud servers, the research starts with existing encryption technologies, namely SE technology and ABSE technology. Research has improved the shortcomings of these two technologies and developed new encryption schemes for each. Finally, these two encryption schemes are deployed on the encryption system to form a complete encryption scheme. SE technology is a technique that allows users to perform search operations in encrypted data, while ensuring the privacy and security of the data in an unencrypted state [8-9]. The advantages of this technology are strong retrieval capability, high flexibility, strong security, support for multiple application scenarios, and effective prevention of data leakage. The schematic of SE technology is shown in Figure 1.

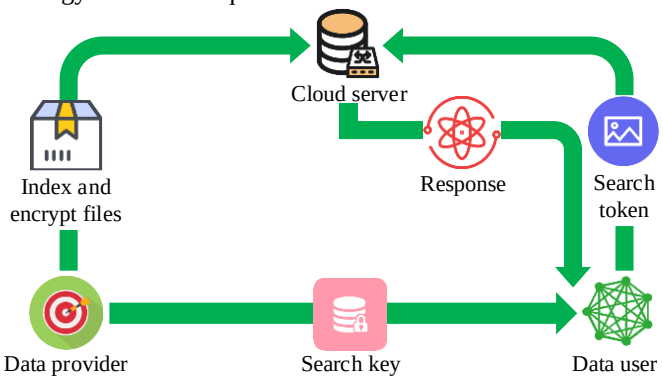


Figure 1: The schematic of SE technology

From Figure 1, SE technology involves data providers, data consumers, cloud servers, search keys, indexes and encrypted files, search tokens, and responses. However, a large portion of SE solutions currently do not involve dynamic changes in user permissions and ignore potential issues with static user access permissions. Therefore, to solve this problem, an SE-oriented user permission dynamic management method has been studied and designed. Figure 2 illustrates the comprehensive structure of this method.

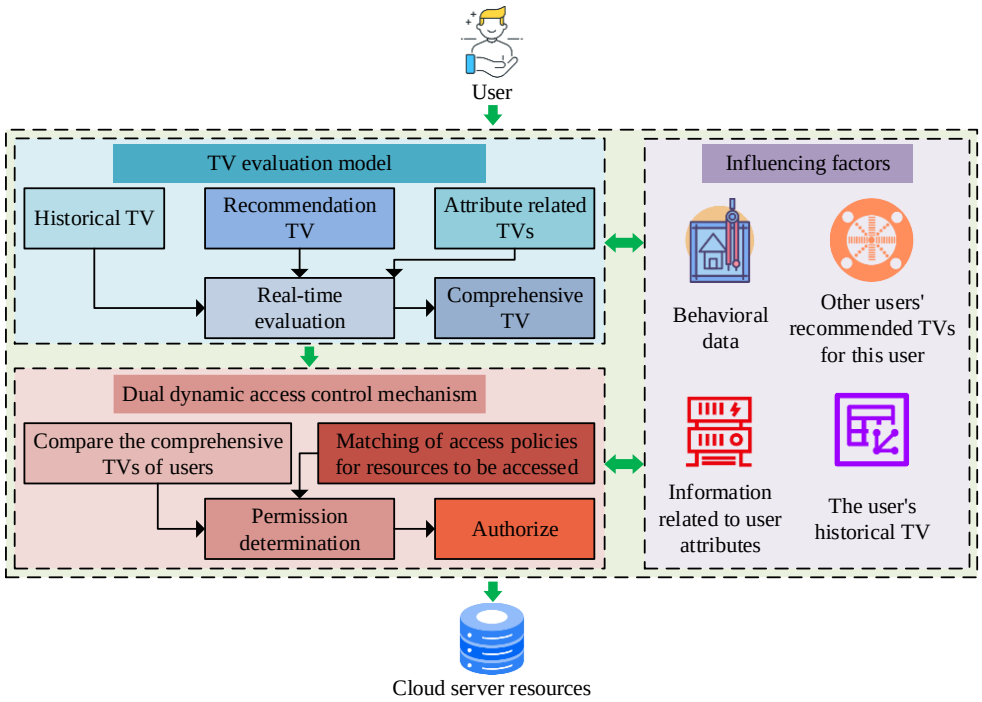


Figure 2: The overall framework of SE oriented user permission dynamic management method

From Figure 2, the user permission dynamic management method mainly includes two key parts, namely the TV evaluation model and the dual dynamic access control mechanism. In the calculation of TVs, including weights or function forms, innovative designs and improvements are made based on previous relevant research [10-11]. The recommended TV d of the system and other registered users h for registered user c is solved as shown in Equation (1) [12].

$$AB_c(D) = \zeta AB_c^g(D) + \psi \sum_h \sin\left(B_h(d-D)\right) \cdot AB_c^h(D) \quad (1)$$

In Equation (1), both ζ and ψ are influence weights, and the sum of the two is 1. $AB_c^g(D)$ represents the recommended TV of the system for registered user c

, D is the time interval, and d represents the specific time. The weight is $\sin\left(B_h\left(d-D\right)\right)$, and the value range is $[0,1]$. B is a TV, and $B_h\left(d-D\right)$ represents the TV of h under the previous evaluation D. $AB_c^h\left(D\right)$ represents the recommended TV of h towards registered user c. In addition, in the calculation of user recommendation TVs, normalization methods need to be used. The reason for using normalization method is to eliminate dimensional differences and dimensional influences between different data sources or indicators. There is a theoretical basis for using Z-score normalization on the minimum maximum scale. Z-score normalization can make data from different sources comparable [13]. The conditions for using this method include normal distribution of data, elimination of dimensional influence, and standardization of data range. The expression of Z-score normalization is in Equation (2) [14].

$$I=\frac{a-b}{G} \tag{2}$$

In Equation (2), I represents the processed data and a is the actual data. b and G are the mean and standard deviations of the dataset, respectively. The expression of range normalization is in Equation (3).

$$I=\frac{a-\min}{\max-\min} \tag{3}$$

In Equation (3), $\max$ and $\min$ represent the maximum and minimum values in the dataset, respectively. In the solution of system recommendation TV, the behavior feature vector of c is expressed as shown in Equation (4) [15].

$$J_c=\left(j_{c,1},j_{c,2},.....,j_{c,k}\right) \tag{4}$$

In Equation (4), $j_{c,1}$ represents the behavior feature of the first row and k is the total number of user behavior features. Therefore, the expression of $AB_c^g\left(D\right)$ is in Equation (5).

$$AB_c^g\left(D\right)=f\cdot\left(L\square J_c\right) \tag{5}$$

In Equation (5), $f\left(\cdot\right)$ represents the activation function, L represents the weight of behavioral features, and $L\square J_c$ is the inner product of vectors. The expression of $L\square J_c$ is in Equation (6).

$$L\square J_c=L_1\square j_{c,1}+L_2\square j_{c,2}+.....+L_k\square j_{c,k} \tag{6}$$

In equation (6), L_k represents the weight of the k th behavioral feature. The solution of $AB_c^h\left(D\right)$ is in Equation (7).

$$AB_c^h\left(D\right)=\frac{M_n}{M_n+P_n} \tag{7}$$

In Equation (7), n represents the interaction process, M_n and P_n respectively represent the number of times the operation of c in n is judged as honest or dishonest. In addition, the solution for the TV $NDD_c\left(D\right)$ related to the c attribute information is in Equation (8).

$$NDD_c\left(D\right)=\upsilon_1\cdot F_1\left(N_{1,s_1}\right)+\upsilon_2\cdot F_2\left(N_{2,s_2}\right)+...+\upsilon_R\cdot F_R\left(N_{R,s_R}\right) \tag{8}$$

In Equation (8), R represents the total number of categories of attributes, υ_1, υ_2, and υ_R all represent influence weights, and their sum is 1. N is the attribute value, and N_{R,s_R} represents the s_R th attribute value in the R th class attribute. F_1, F_2, and F_R represent conversion functions. s is the number of attributes of a certain type. In addition, different conversion functions are expressed uniformly, as shown in Equation (9).

$$F_1\left(N\right)=F_2\left(N\right)=...=F_R\left(N\right)=e^N-1 \tag{9}$$

Therefore, the solution for the comprehensive TV $B_c\left(D\right)$ is in Equation (10).

$$B_c\left(D\right)=\omega_1\cdot B_h\left(d-D\right)+\omega_2\cdot AB_c\left(D\right)+\omega_3\cdot NDD_c\left(D\right) \tag{10}$$

In Equation (10), ω_1, ω_2, and ω_3 represent the influence weights, and the sum of the three is 1. The process of calculating the comprehensive TV is as follows: first initialize the user TV and assign a set of attributes, then collect user behavior data in real time, weight and sum historical TVs (calculated based on behavior records), normalize recommended TVs (system and other user recommended inputs), and attribute-related TVs (calculated based on the correlation between attributes and data) to obtain the comprehensive TV. Finally, dynamically adjust user permissions based on this, and compare TVs with thresholds to determine authorization during resource access. The pseudocode of the TV evaluation model is shown in Figure 3.

```
function calculateCompositeTrust(user):
    historicalTrust = user.getHistoricalTrust()
    recommendedTrust = calculateRecommendedTrust(user.getRecommendations())
    attributeTrust = calculateAttributeTrust(user.Attributes)
    weight1 = 0.4
    weight2 = 0.3
    weight3 = 0.3
    compositeTrust = weight1 * historicalTrust + weight2 * recommendedTrust +
weight3 * attributeTrust
    return compositeTrust

function calculateRecommendedTrust(recommendations):
    normalizedRecommendations = normalize(recommendations)
    return normalizedRecommendations

function calculateAttributeTrust(attributes):
    attributeTrust = 0.0
    for attribute in attributes:
        attributeTrust += attribute.value
    return attributeTrust / len(attributes)
```

Figure 3: Pseudo code for TV evaluation model

From Figure 3, it can be seen that the pseudocode defines how to calculate the user's comprehensive TV. Subsequently, the generated comprehensive TV is input into the dual dynamic access control mechanism section. In the dual dynamic access control mechanism, the core is permission determination, and the determination content is to compare the comprehensive TVs of users and match the access policies of the resources to be accessed. The

pseudocode of the dual access control mechanism is shown in Figure 4.

```
function dualDynamicAccessControl(userRequest, resourcePolicy):
    compositeTrust = calculateCompositeTrust(userRequest.user)
    if compositeTrust < trustThreshold:
        return ACCESS_DENIED
    if not userRequest.attributes.match(resourcePolicy):
        return ACCESS_DENIED
    return ACCESS_GRANTED
```

Figure 4: Pseudo code for dual access control mechanism

From Figure 4, it can be seen that the pseudocode describes the core logic of the dual dynamic access control mechanism. In addition, the dynamic management method of user permissions also involves factors that affect the overall TV, and this part is the foundation of two key parts that need to be provided with relevant user information. Therefore, the overall process of the user permission dynamic management method is in Figure 5.

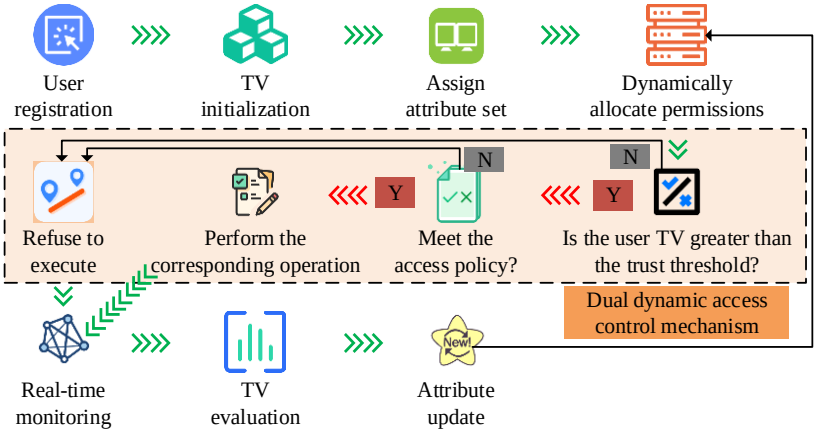


Figure 5: The overall process of dynamic management method for user permissions

From Figure 5, the first step of the user permission dynamic management method is to register the user, and the second step is to initialize the TV. The third step is to allocate attribute sets, and the fourth step is to dynamically allocate permissions. The fifth step is to enter the dual dynamic access control mechanism. In this mechanism, the first step is to determine whether the user's TV is greater than the trust threshold. If it is greater than the trust threshold, the next step is to proceed. Otherwise, execution is rejected and real-time monitoring is initiated. Secondly, it is necessary to determine whether the access policy is met. If it is met, the corresponding operation should be executed before proceeding to real-time monitoring. Otherwise, it will directly proceed to real-time monitoring. The sixth step is to evaluate the TV, the seventh step is to update the attributes, and then return to dynamically assigning permissions.

In addition, in the SE-based user permission dynamic management method, the dual dynamic access control mechanism closely interacts with the underlying SE operations. Specifically, when a user initiates a search request, the system first performs a TV check on the client side, compares the user's comprehensive TV with a preset trust threshold, and verifies whether it complies with the resource access policy. This process occurs before generating the search token, ensuring that only trusted users can create valid search tokens. If the user passes the TV check, the client will generate a search token based on the user's attributes and permissions, and send it to the cloud server. After receiving the search token, the cloud server performs a search operation and returns a list of encrypted files that match the token. After obtaining the encrypted file, the user performs a TV check again to ensure that they still have the authority to decrypt the data. This TV-based dynamic access control mechanism runs through the entire search and retrieval process of SE, ensuring data security and dynamic access.

2.2 Design of ABSE scheme Using B-Tree index structure

A dynamic user permission management method for SE has been designed to address the security management issues of data storage on cloud servers. However, in response to the improvement of SE technology, existing research has proposed ABSE technology that combines SE technology with attribute-based encryption. ABSE technology allows data owners to encrypt data based on its attributes, supports fine-grained access control and private keyword search, and ensures data security [16-17]. However, this technology still has certain shortcomings, such as rigid access policies and low search efficiency. In order to address these issues and further improve ABSE technology, methods that can supporting fine-grained access control have been researched and designed. The system model of this method is in Figure 6.

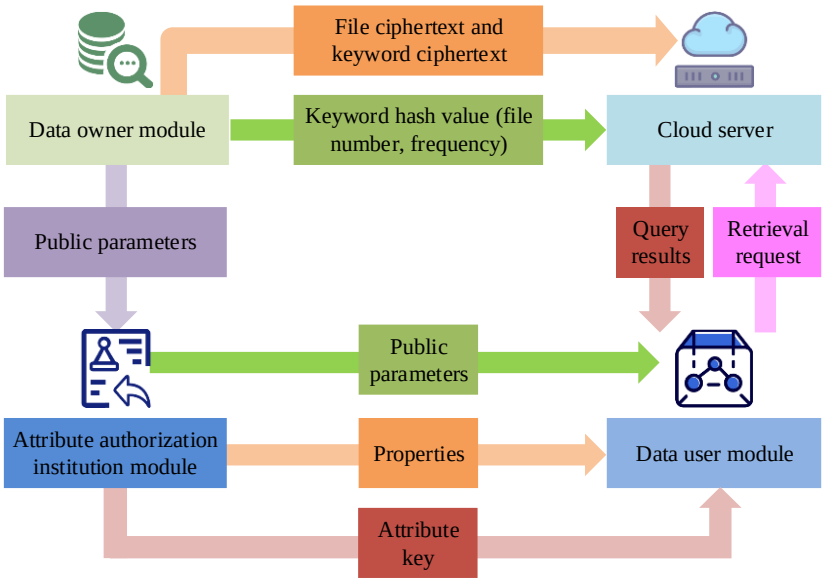


Figure 6: System model supporting fine-grained access control methods

From Figure 6, the system model supporting fine-grained access control methods mainly includes the data owner module, data user module, cloud server, attribute authorization authority module, public parameters, attribute sets, attribute keys, file ciphertexts and keyword ciphertexts, keyword hash values (file numbers, frequency), query results, and retrieval requests. Among them, the attribute authorization authority module needs to generate a system master key and system initialization, and the cloud server needs to run a search algorithm. Meanwhile, the data owner module needs to run encryption algorithms, while the data user module needs to obtain ciphertext files corresponding to keywords, and this process requires running search token generation algorithms.

In the construction process of supporting fine-grained access control methods, the B-Tree index structure is adopted to optimize search efficiency and search result accuracy. The B-Tree index structure is an efficient search tree, mainly used to accelerate data retrieval. It has the advantages of high data retrieval performance, support for ordered range queries and dynamic adjustments, and is widely used in database systems, in memory databases, and caching systems [18-19]. The construction process of B-Tree index structure is in Figure 7.

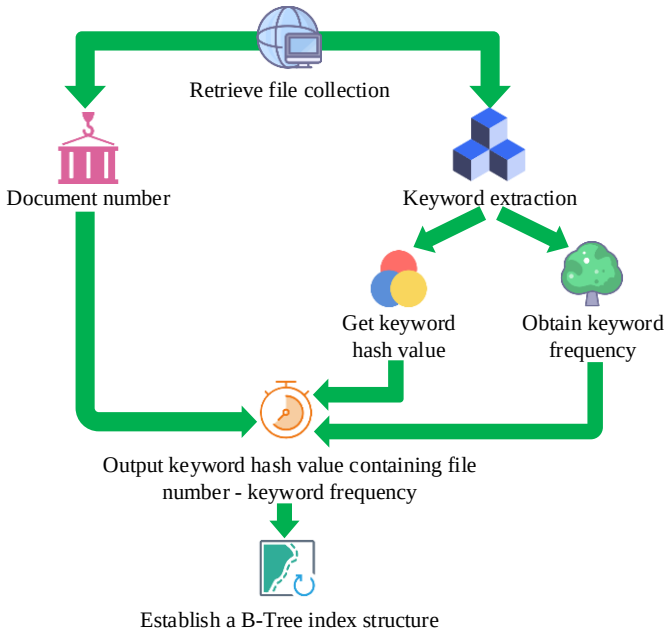


Figure 7: The construction process of B-Tree index structure

From Figure 7, the first step in constructing the B-Tree index structure is to obtain a set of files, and the second step involves extracting file numbers and keywords separately. Meanwhile, in keyword extraction, it is necessary to obtain keyword frequency and keyword hash value. The third step is to output a keyword hash

value containing the file number and keyword frequency, and the fourth step is to establish a B-Tree index structure. The pseudocode of B-Tree structure is shown in Figure 8.

```
class BTreeNode:
    def __init__(self):
        self.keys = []
        self.children = []
        self.is_leaf = True

function insertIntoBTree(node, key):
    if node.is_leaf:
        node.keys.append(key)
        node.keys.sort()
    else:
        child = findAppropriateChild(node, key)
        insertIntoBTree(child, key)
    if len(node.keys) > max_keys:
        splitNode(node)

function searchInBTree(node, key):
    if node.is_leaf:
        return key in node.keys
    else:
        child = findAppropriateChild(node, key)
        return searchInBTree(child, key)
```

Figure 8: Pseudo code of B-Tree structure

From Figure 8, the pseudocode defines the B-Tree node structure and implements insertion and search operations. In addition, when generating the system master key, it is necessary to first initialize and generate public parameters. The T expression of the system master key is in Equation (11) [20].

$$T = \left(\tau, \{u_v\}_{v=1}^{V}\right) \tag{11}$$

In Equation (11), τ represents a random element and u_v represents the random value corresponding to the attribute. v is the serial number of the attribute, and V represents the total number of attributes. τ and u_v have $\tau \in Z_p^*$ and $u_v \in Z_p^*$ respectively, and Z_p^* is a special symbol [21]. The expression of system public parameters is in Equation (12).

$$U = \left(w, w_1, e(w,w)^{\tau}, \{x_v\}_v^V, \ Y\right) \tag{12}$$

In Equation (12), both w and w_1 are generators of the multiplicative cyclic group, $e(w,w)^{\tau}$ represents bilinear pairing operations. Y is the hash algorithm, and x_v represents the group elements generated from the attribute set. In addition, the generation method of user attribute key X is in Equation (13).

$$X = \left(\sigma, \{\sigma_\vartheta\}_{\vartheta=1}^{\theta}\right) \tag{13}$$

In Equation (13), θ represents the number of user attributes and ϑ is the user attribute number. σ represents the authorization corresponding to user attributes. Among them, "authorization" σ refers to the authorization information of the attribute authorization agency for user attributes. It is employed together with attribute values and master keys to export user attribute keys. The attribute authorization agency uses the random elements and corresponding random values in the system master key, combined with user attribute information, to calculate the user attribute related key part through one-way hash function and key generation algorithm. This ensures that only authorized users can obtain the correct attribute key to decrypt or generate valid search tokens, ensuring both user attribute confidentiality and system security and fine-grained access control.

The study explains the implementation details of the B-Tree index structure, including order, balancing strategy, and concurrency control. In terms of sequence, a B-Tree index is constructed by extracting file numbers and keywords from the file set, storing keywords and their file information in order, and recursively locating when inserting new keywords to maintain order. In regard to balancing strategy, node splitting and merging operations are used to maintain balance, ensuring that the number of node keywords does not exceed the preset upper limit. In terms of concurrency control, existing technologies such as locking or timestamp sorting mechanisms are combined to manage concurrent access and updates, ensuring data consistency and integrity during multi-user operations.

The linear secret sharing matrix is one of the specific implementation details of LSSS, which describes in detail how secrets are divided and reconstructed from authorized shares. This matrix driven approach ensures the security and flexibility of the solution, enabling it to adapt to various complex access control requirements. The design and parameter selection of matrices directly affect the efficiency and security of the scheme, and are the foundation for implementing fine-grained access control and secure multi-party computation. Therefore, in the encryption stage, a linear secret sharing matrix was used to achieve access control for users. Linear secret sharing matrix is a technology based on matrix operations to achieve secret information segmentation and recovery. It has the advantages of high security, strong flexibility, good scalability, and supports dynamic adjustment. It finds extensive utility in confidential file transmission, cloud computing and privacy protection, IoT security, and blockchain security [22-24]. Therefore, the ϖ expression of the complete ciphertext is in Equation (14).

$$\varpi = (\mu, \lambda) \tag{14}$$

In Equation (14), μ represents symmetric encrypted ciphertext and λ represents keyword ciphertext. The expression of the search token η is in Equation (15).

$$\eta = \prod_{\varepsilon}^{\delta} w_1^{Y(\varphi_\varepsilon)} \tag{15}$$

In Equation (15), φ_ε represents the keyword and ε is the sequence number of the keyword. δ represents the total number of keywords, and $Y(\varphi_\varepsilon)$ represents the hash value of φ_ε. To address issues such as keyword conflicts, copying, and revocation, researchers have developed a series of processing methods. When keyword conflicts arise, the system employs salt hashing to boost the element of unpredictability, thereby mitigating conflicts. Additionally, it records relevant information to

differentiate between keywords. In scenarios where keyword copying occurs, the system assigns unique identifiers to the keywords, updates the information within the B-Tree index, and verifies uniqueness during token generation. When it comes to keyword revocation, the system utilizes version control tags to revoke the specific version and simultaneously updates the ciphertext file information.

In the ABSE framework, B-Tree stores file keywords and attribute information to support attribute access control. When encrypting files, the data owner stores keyword and attribute information in the B-Tree index. When users search, the attribute authorization agency generates a search token containing user attributes and access policies. The cloud server uses B-Tree to locate files, verifies whether user attributes meet policies through LSSS, and decides whether to allow or deny access. To determine user access permissions, cloud servers need to combine η and X. To form a complete encryption scheme, the two designed schemes will be integrated into the encryption system together. Figure 9 presents the comprehensive structure of the encryption system.

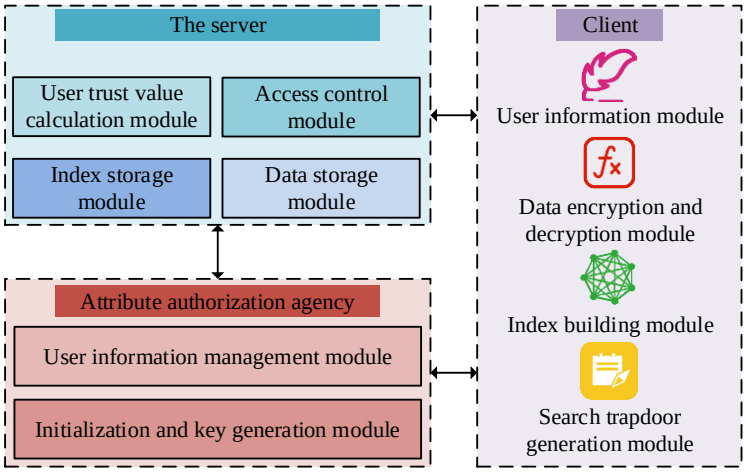


Figure 9: Comprehensive structure of encryption system

From Figure 9, the comprehensive structure of the encryption system mainly consists of three main parts, namely the client, server, and attribute authorization authority. Meanwhile, all three parts are interconnected with each other. In addition, the client includes a user information module, a data encryption and decryption module, an index construction module, and a search trapdoor generation module. The server consists of four main modules, and the most critical one is TV calculation. Finally, the user information management module and the initialization and key generation module constitute the attribute authorization authority. The sequence diagram of the encryption system is shown in Figure 10.

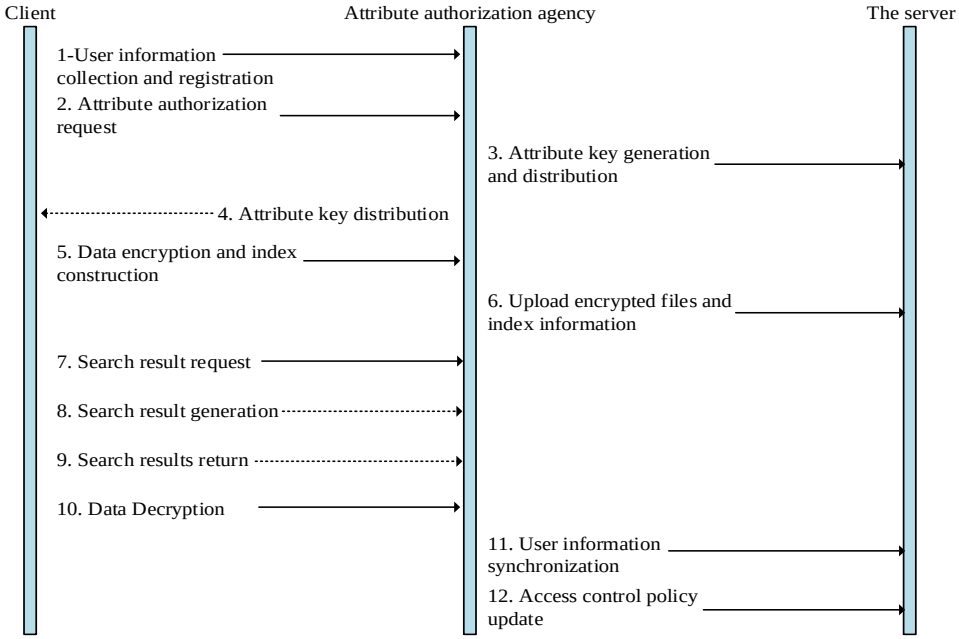


Figure 10: Sequence diagram of encryption system

From Figure 10, after collecting user information and registering, the client requests attribute authorization from the attribute authorization agency. The attribute authorization agency generates attribute keys and distributes them to clients, who use them to encrypt data and build indexes, and then upload the encrypted files and indexes to the cloud server. After the client initiates a search request, the cloud server returns the search results, and the client decrypts the results. Meanwhile, the cloud server synchronizes user information with the attribute authorization agency, which updates access control policies and notifies the cloud server to ensure system data security and the effectiveness of access control.

In the final plan, the TV-based dynamic access control mechanism enhances rather than replaces the attribute based mechanism. When making access decisions, first verify the user's identity and attributes, and then evaluate the TV. If the TV exceeds the threshold and the attributes comply with the policy, access is allowed; otherwise, access is denied. This combination enhances system security and flexibility, effectively protecting cloud server data.

3 Results

To confirm the capability of the research design scheme, the experimental environment was first described, and then the comparative scheme was selected. In regard to comparative indicators, the study selected factors such as time consumption, memory usage, Central Processing Unit (CPU) utilization, and traffic analysis.

3.1 Performance verification of dynamic management methods for user permissions

To confirm the performance of the dynamic management method for user permissions in the research design, experiments were conducted using Python programming language combined with PyCrypto and OpenSSL cryptographic libraries. The paper utilized the PyCrypto library to implement encryption methods to ensure data confidentiality and integrity, and used the SSL/TLS protocol supported by the OpenSSL library to ensure communication security and prevent data from being stolen or tampered with during transmission. The operating system was Windows 10, the CPU was Intel Core i5-12600KF, the main frequency and dynamic acceleration frequency were 3.7GHz and 4.9GHz, respectively, and the number of cores was ten. The maximum memory bandwidth was 76.8GB/s. The study established a threat model that included data leakage, unauthorized access, data tampering, insider threats, and man in the middle attacks. Attackers may have the ability to intercept data transmission, crack encryption algorithms, forge user identities, tamper with stored data, and exploit system vulnerabilities. The types of attacks that the system can defend against include data leakage, unauthorized access, data tampering, insider threats, and man in the middle attacks. The specific defenses are as follows: the system uses strong encryption algorithms to prevent data leakage, resists unauthorized access through fine-grained access control and dynamic trust evaluation, uses digital signatures and integrity verification to ensure data integrity, follows minimum privilege and audit mechanisms to prevent internal threats, and uses secure communication protocols to prevent man in the middle attacks. In addition, there are two trusted vacation options for servers or clients, namely: cloud servers are considered semi trusted and must perform operations according to the protocol, and ensure the security of the operating environment through regular audits and vulnerability scans. The client is generally trustworthy, but identity authentication and trust assessment are still required, while ensuring local security to prevent malicious software from stealing user credentials or keys. In regard to comparative methods, the study selected four schemes and represented them using Scheme A, Scheme B, Scheme C, and Scheme D. Among them, Scheme A is an access control scheme based on roles and user TVs, dynamically adjusting user permissions by defining roles and assigning TVs. Scheme B is a trust management system for wireless sensor networks based on Bayesian trust model. This scheme utilizes the principle of Bayesian networks to construct a trust model based on users' historical interaction behavior, recommendation information, etc., and calculates the trust relationship between users through probabilistic inference to determine access permissions. Scheme C is a dynamic access control and authorization system based on zero trust, which requires strict authentication of user identity and dynamic granting of minimum permissions for each access. Scheme D is a hybrid access control scheme based on attributes and TVs. This scheme integrates user attributes and TV evaluation, and users need to meet attribute requirements and TV standards to access resources. Meanwhile, TV comprehensively considers factors such as user attribute relevance and behavior history. The reason for choosing these four methods was that they cover different trust evaluation and access control mechanisms, which can be effectively compared with the paper design scheme, highlighting the advantages of the paper scheme in regard to TV evaluation accuracy, access control flexibility, and system performance. In regard to parameter values, there were four possible values for ω_1 in the research design scheme, namely 0.8, 0.6, 0.4, and 0.2. Meanwhile, the initial TV was set to 0.5. In addition, the initial TVs for schemes A, B, C, and D are also 0.5. The comparison of TV calculation time and access control determination time for different schemes is in Figure 11.

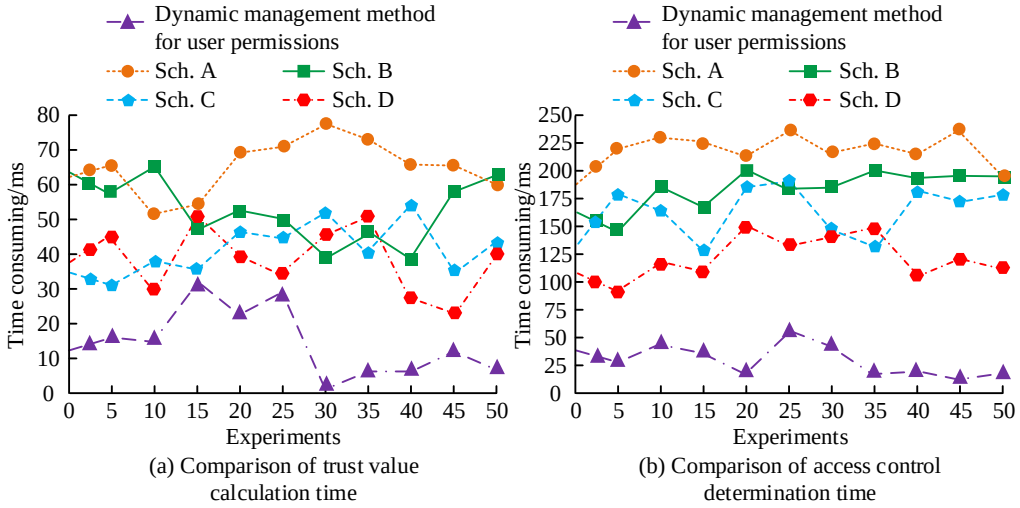


Figure 11: Comparison of TV calculation time and access control determination time for different schemes

From Figure 11 (a), in regard to the comparison of TV calculation time, the research-designed user permission dynamic management method performed better, with corresponding maximum and minimum values of 32 ms and 0.6 ms, respectively. Meanwhile, the maximum TV calculation time of Scheme A, Scheme B, Scheme C, and Scheme D was 78 ms, 65, 54 ms, and 51 ms, respectively, which were 46 ms, 33 ms, 22 ms, and 19 ms higher than 32 ms. From Figure 11 (b), when comparing the access control judgment time of different schemes, the maximum value of the research designed user permission dynamic management method was 56 ms, while the maximum values of the four comparison schemes were 237 ms and 201 ms, respectively. The differences between 189 ms, 150 ms, and 56 ms were 181 ms, 145 ms, 133 ms, and 94 ms, respectively. In summary, the TV calculation and access control determination time of the research and design user permission dynamic management method were both lower, and the performance was better. The rationality comparison of TVs for different schemes is in Figure 12.

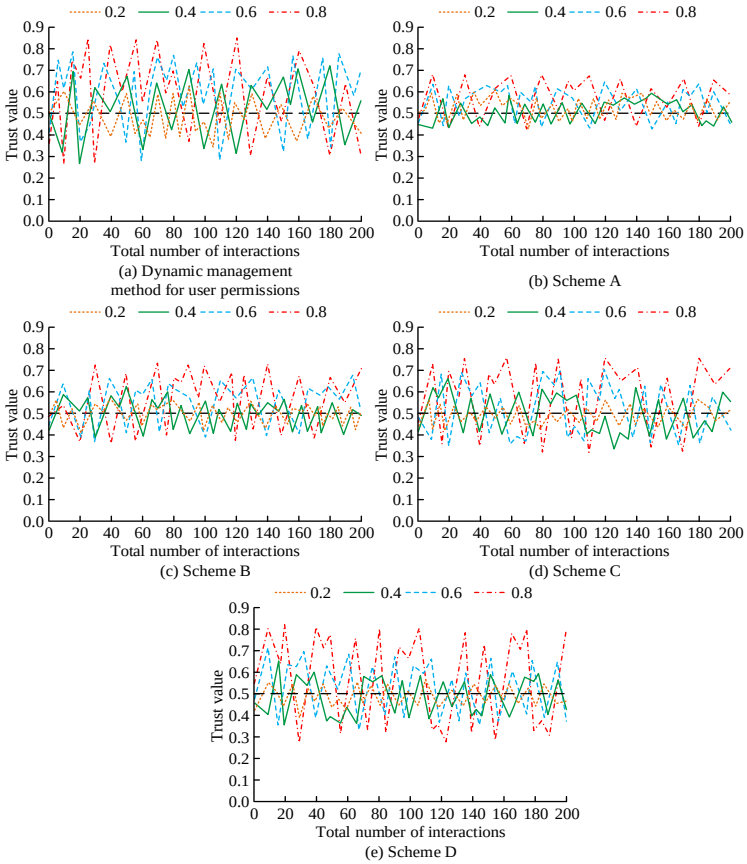
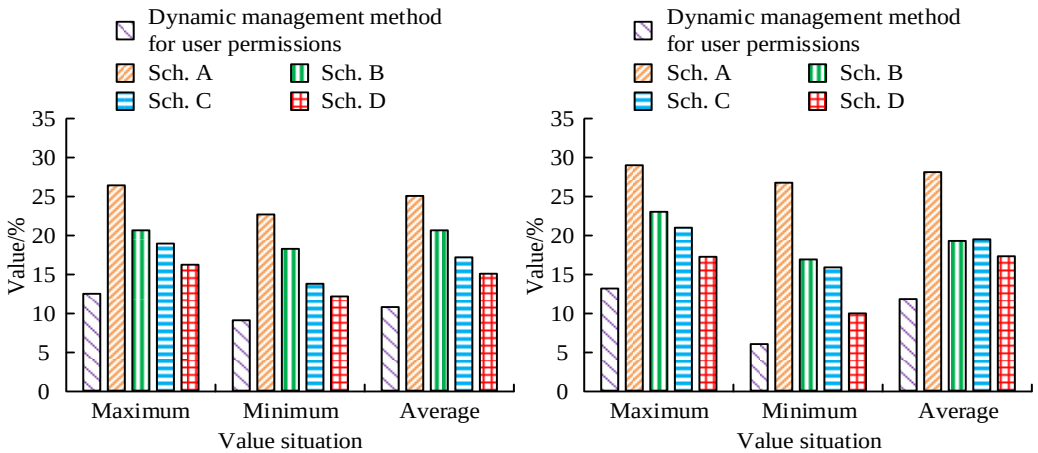


Figure 12: Comparison of the rationality of TVs for different schemes

According to Figure 12 (a), in the user permission dynamic management method designed for research, the fluctuation range of TVs for random users under different weight values was [0.227, 0.845], and the difference between their maximum and minimum values was 0.618. From Figure 12 (b), Figure 12 (c), Figure 12 (d), and Figure 12 (e), in Scheme A, Scheme B, Scheme C, and Scheme D, the fluctuation range of TVs of random users was [0.423, 0.698], [0.385, 0.725], [0.336, 0.761], and [0.285, 0.802], respectively. The difference between the maximum and minimum values was 0.275, 0.340, 0.425, and 0.517, all of which were less than 0.618. The comparison of CPU utilization and memory usage between different schemes is in Figure 13.



(a) Comparison of CPU Utilization (b) Memory usage rate

Figure 13: Comparison of CPU utilization and memory usage among different schemes

From Figure 13 (a), in regard to CPU utilization comparison, the maximum value for studying and designing user permission dynamic management methods was 12.17%, and the minimum value was 8.9%. Meanwhile, the maximum CPU utilization rates of the four comparison schemes were 26.98%, 20.17%, 18.55%, and 16.29%, respectively, which were 14.81%, 8.00%, 6.38%, and 4.12% higher than 12.17%. From Figure 13 (b), when comparing the memory occupancy rates of different schemes, the user permission dynamic management method designed by the research ranked first, followed by Scheme D, Scheme C, Scheme B, and Scheme A in the second to fifth positions. Meanwhile, according to the ranking order, the maximum memory occupancy rates of the five schemes were 13.95%, 17.52%, 20.31%, 23.17%, and 28.55%, respectively. In summary, the research and design of dynamic management methods for user permissions had lower CPU utilization and memory usage, and better performance. To further validate the performance of the research designed user permission dynamic management method, encryption evaluations were conducted on both the method and the comparison method, involving security proof indicators and resistance indicators to known attacks. The encryption evaluation results of different methods are shown in Table 2.

Table 2: Encryption evaluation results of different methods

Scheme	Safety certificate					Resistance to known attacks				
	Number of experiments					Number of experiments				
	1	2	3	4	5	1	2	3	4	5
Scheme A	64	68	68	67	61	68	65	70	61	62
Scheme B	66	68	66	66	66	66	73	67	71	67
Scheme C	79	82	79	81	77	77	77	80	82	81
Scheme D	86	81	80	89	82	87	80	83	82	85
Manuscript	93	91	96	92	93	96	97	98	95	94

From Table 2, in the five experiments of security proof, the scores of the research design user permission dynamic management method were all above 90 points, with the highest reaching 96 points, significantly higher than other comparative schemes. In regard to resistance to known attacks, the research and design of user permission dynamic management methods also scored above 90 points, with the highest reaching 98 points, demonstrating extremely strong resistance to attacks. In contrast, Scheme A and Scheme B scored lower on both indicators, while Scheme C and Scheme D performed well but still did not reach the level of research design methods.

3.2 Performance validation supporting fine-grained access control methods

To validate the performance of fine-grained access control methods, the same experimental setup was used in the study. In addition, in regard to comparative methods, the study selected four schemes and named them Scheme 1, Scheme 2, Scheme 3, and Scheme 4 respectively. Among them, Scheme 1 is a traditional symmetric SE scheme, which uses symmetric encryption algorithms to encrypt data and constructs an index structure for searching. Option 2 is the ABSE scheme, which implements fine-grained access control through attribute

encryption and SE techniques. Scheme 3 is a dynamic access control SE scheme based on zero trust. This scheme combines a zero trust security model and SE technology. Every time a user initiates a search request, their identity must be re verified and dynamically evaluated, and the evaluation results are used to determine whether to allow the user to search and access data. Scheme 4 is a blockchain based SE scheme, which utilizes blockchain to record data hash and permission information, and combines SE to achieve technical security retrieval. The reason for choosing these four methods was that they represent encryption and access control methods of different technological routes, which can comprehensively compare the advantages of this research scheme in regard to encryption efficiency, access control flexibility, security, and communication cost. Meanwhile, these solutions all included five stages: system initialization, key generation, encryption, search token generation, and search. In addition, there were a total of 6 values for the number of user attributes, namely 1, 6, 11, 16, 21, and 26, and the access policy involved the same number of attributes. The comparison of time consumption for different parts is in Figure 14.

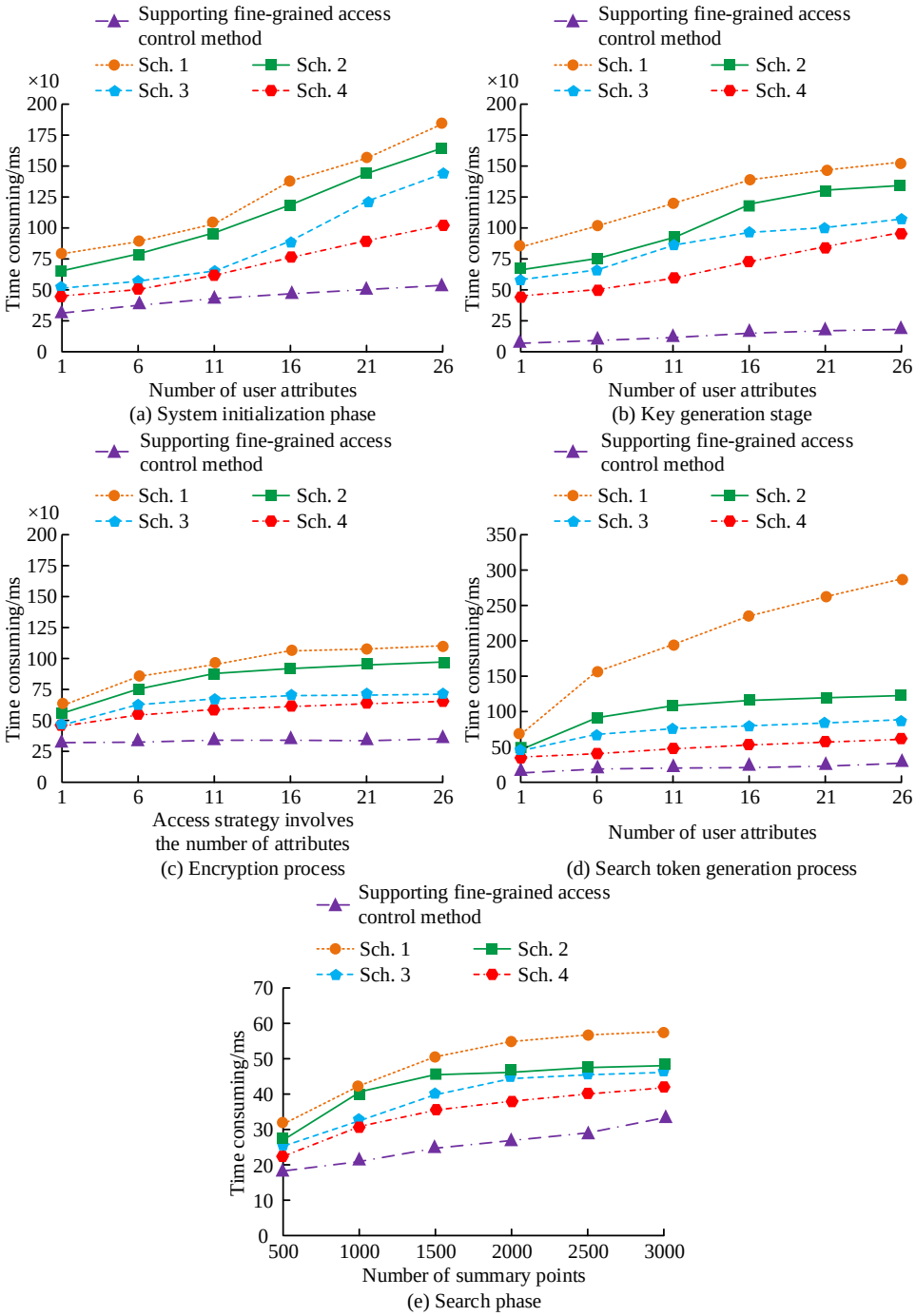


Figure 14: Comparison of time consumption for different parts

According to Figure 14 (a), during the system initialization phase, as the number of user attributes increased, the corresponding time consumption for different schemes also increased synchronously. The increase speed of the fine-grained access control method designed in this study was slower, while the increase speed of Scheme 1 was faster. In addition, in regard to specific values, the max time consumption of schemes 1, 2, 3, and 4 was 1852 ms, 1652 ms, 1400 ms, and 1050 ms, respectively, while the max time consumption of the research-designed fine-grained access control method was 580ms. As shown in Figure 14 (b), in the key generation stage, the max time consumption of the five schemes was 180 ms, 1558 ms, 1327 ms, 1106 ms, and 985 ms, respectively. From Figure 14 (c), in the encryption stage, the research-designed supporting fine-grained access control method performed better, followed by schemes 4, 3, and 2, and finally scheme 1. Meanwhile, the max encryption time for the five schemes was 325 ms, 657 ms, 739 ms, 962 ms, and 1107 ms, respectively. The trend of encryption time for the five schemes increased synchronously with the increase in the number of attributes involved in the access policy. From Figure 14 (d), in the search token generation stage, the max time consumption of the research-designed supporting fine-grained access control method and the four comparison schemes were 25 ms, 287 ms, 120 ms, 85 ms, and 60 ms, respectively. From Figure 14 (e), in the search stage, the max time consumption of the five schemes were 33 ms, 57 ms, 48 ms, 46 ms, and 42 ms, respectively. Overall, the research-designed supporting fine-grained access control method had shorter time consumption and better performance. The comparison of communication volume between different schemes is in Table 3.

Table 3: Comparison of communication volume between different schemes

Scheme	Number of keywords					
	1	2	3	4	5	6
Scheme 1	3.336	4.985	6.226	7.679	9.017	10.369
Scheme 2	3.017	4.119	5.654	7.301	8.435	10.195
Scheme 3	2.778	3.657	4.753	6.432	8.003	9.665
Scheme 4	2.374	2.892	4.147	6.158	7.553	9.448
Supporting fine-grained access control methods	1.137	2.015	2.225	2.785	3.012	3.257

In Table 3, the communication volume is represented by the amount of data, and the unit is Byte. From Table 3, as the number of keywords increased, the amount of data corresponding to different schemes also increased synchronously. Within the same range of keyword changes, the data volume of scheme 2 changed more significantly, about 7.178 Byte. Meanwhile, the data volume changes for Scheme 1, Scheme 3, Scheme 4, and the research-designed supporting fine-grained access control method were 7.033 Byte, 6.887 Byte, 7.074 Byte, and 2.120 Byte, respectively. The research-designed supporting fine-grained access control method had smaller changes in data volume, which were 4.913 bytes, 5.058 bytes, 4.767 bytes, and 4.954 bytes less than the comparison scheme, respectively. This method had lower communication volume and better performance. The storage overhead comparison of different schemes is in Table 4.

Table 4: Comparison of storage costs for different solutions

Scheme	User key storage overhead/KB					Cryptocurrency storage overhead/KB				
	User attribute set size					Leaf node set size				
	5	10	15	20	25	10	15	20	25	30
Scheme 1	3.45	4.73	5.69	8.71	8.98	5.13	6.08	7.32	8.65	9.17
Scheme 2	3.13	4.03	5.36	8.42	8.66	4.65	5.73	6.54	8.42	8.88
Scheme 3	2.65	4.12	4.95	7.11	7.79	4.11	4.72	6.36	7.89	8.34
Scheme 4	2.23	3.87	4.67	6.52	7.42	3.89	4.57	5.98	7.42	7.62
Supporting Fine-grained access control method	1.74	2.54	3.89	5.14	6.37	3.34	4.17	5.24	6.85	7.25

From Table 4, in regard to the comparison of user key storage costs, the research-designed supporting fine-grained access control method performed better, with an average value of 3.94 KB. In addition, the average user key storage costs of the four comparison schemes were 6.31 KB, 5.92 KB, 5.32 KB, and 4.94 KB, respectively, which were 2.37 KB, 1.98 KB, 1.38 KB, and 1.00 KB higher than 3.94 KB. In addition, as the size of the random user attribute set increased, the user key storage costs corresponding to different schemes also increased synchronously, and schemes 1 and 2 increase more. Meanwhile, in regard to ciphertext storage overhead, the research-designed supporting fine-grained access control method and the average values of the four comparison schemes were 5.37 KB, 7.27 KB, 6.84 KB, 6.28 KB, and 5.90 KB, respectively. The significance of reducing storage overhead by 2-3KB was twofold: firstly, it lowered the storage cost of cloud servers. Secondly, it reduced the storage pressure on the client and avoided problems caused by insufficient storage. Furthermore, it could improve system efficiency, accelerate key read/write and distribution speed, shorten loading and response time, and enhance user experience. Finally, reducing storage overhead made the system easier to cope with user and data growth, reduce performance bottleneck risks, and benefit system scalability. Overall, the research -designed supporting fine-grained access control method had lower storage overhead and better performance. To further verify the robustness of fine-grained access control methods, long-term performance analysis and stress testing were conducted, with specific indicators being system response time and system throughput. The

robustness comparison of different methods is shown in Table 5.

Table 5: Comparison of robustness of different methods

Scheme	System response time/ms					System throughput/(req/s)				
	Run time/hour					Number of concurrent users				
	12	24	48	72	96	50	100	200	500	1000
Scheme 1	120	135	150	165	180	25	45	65	80	90
Scheme 2	110	120	130	140	150	30	50	70	85	95
Scheme 3	100	110	120	130	140	35	60	80	90	95
Scheme 4	90	100	110	120	130	40	70	90	100	105
Support Fine-grained access control methods	80	90	100	110	120	55	85	120	140	165

From Table 5, methods that support fine-grained access control performed excellently in terms of system response time and throughput. After 96 hours of system operation, its response time was only 120 milliseconds, which was 10-60 milliseconds lower than other solutions. With 1000 concurrent users, the throughput reached 165 times/second, which was 60-75 times/second higher than other solutions, demonstrating better performance stability and system resilience.

In practical applications, a centralized key management scheme was studied to address the issue of handling key management across multiple users: attribute authorization agencies generated and distributed attribute keys uniformly, ensuring that key generation followed consistent policies and was securely distributed to users. Meanwhile, using the user information management module of the attribute authorization agency, user identity and attribute information were centrally managed and authenticated. The system dynamically adjusted the key according to changes in user attributes to ensure that only users with legitimate attributes can obtain the corresponding key. In addition, the system regularly updated keys and set expiration dates to further improve key security and reduce risks caused by key leaks.

4 Discussion

To conduct critical analysis, the study compared the designed method with the techniques in Table 1, and the comparison results are shown in Table 6.

Table 6: Comparison between the designed method and the techniques in Table 1

Comparative method	Quantitative results
Liu et al. [4]	Reduce computing costs by 20% and storage expenses by 10%
Ng et al. [5]	Efficiency increased by 30%, computational cost increased by 8%
Liu et al. [6]	40% increase in computational efficiency and 25% increase in storage overhead
Lu et al. [7]	Efficiency increased by 35%, storage overhead increased by 20%
Dynamic management method for user permissions	Reduce computing costs by 30% and storage expenses by 20%
Support Fine-grained access control methods	Reduce computing costs by 25% and storage expenses by 15%

From Table 6, the proposed method for dynamically managing user permissions reduced the overhead of permission determination and resource allocation through comprehensive trust evaluation and dual dynamic access control. Meanwhile, it optimized data retrieval efficiency using B-tree indexing, thereby reducing computation and storage costs. In addition, methods that support fine-grained access control utilize linear secret sharing schemes to achieve precise access control and reduce resource consumption during encryption and retrieval processes. However, in some cases, such as when the number of users increased sharply or the data scale expands significantly, the system may face performance pressure and need to balance security and efficiency.

5 Conclusion

The research designed a user permission dynamic management scheme and a fine-grained access control scheme for SE technology to address the security management issues of cloud server data storage, and developed a complete encryption scheme. The results showed that the max and min TV calculation time of the user permission dynamic management scheme were 32 ms and 0.6 ms, respectively, and the max access control determination time was 56 ms, which was 181 ms, 145 ms, 133 ms, and 94 ms lower than the max value of the comparison scheme. The user permission dynamic management scheme had a shorter time consumption, which might be due to its use of a comprehensive TV evaluation model and a dual dynamic access control mechanism, the introduction of an efficient mathematical model, and the significant improvement in computational efficiency. In regard to memory and CPU usage, dynamic management of user permissions also presented significant advantages. The max time consumption for supporting fine-grained access control method at different stages was 1852 ms, 180 ms, 325 ms, 25 ms, and 33 ms, respectively, which was significantly better than the comparative scheme. This might be due to its adoption of B-Tree indexing structure and linear secret sharing scheme, which optimized search efficiency and flexibility of access control. In regard to communication volume comparison, the supporting fine-grained access control method had a smaller change amplitude, with a value of 2.120 Byte, which as 4.913 Byte, 5.058 Byte, 4.767 Byte, and 4.954 Byte smaller than the comparison scheme, respectively. In summary, the overall encryption scheme

designed in the research had good performance. However, there are also certain shortcomings in the research. The linear secret sharing scheme brought high flexibility but also high complexity, mainly reflected in the complexity of the access structure and the computational cost of matrix operations. Complex access structures may lead to reduced efficiency in processing large-scale data, while frequent matrix operations may increase computational overhead. Future research can adopt simpler access structure designs, such as "lightweight LSSS", and introduce distributed computing technology to share computing tasks. In addition, when the number of users or data scale increases sharply, the system may face performance bottlenecks, and TV evaluation models and access control mechanisms may lead to reduced efficiency when deployed in resource constrained environments. Future research can be improved by introducing distributed computing technology, adopting lightweight encryption and access control schemes, machine learning and artificial intelligence technologies, and enhancing the caching mechanism of the system.

References

[1] Zhang Y, Zhu T, Guo R, Xu S, Cui H, Cao J. Multi-keyword searchable and verifiable attribute-based encryption over cloud data. IEEE Transactions on Cloud Computing, 2023, 11(1):971-983. https://doi.org/10.1109/TCC.2021.3119407

[2] Zhang X, Mu D, Zhao J. Attribute-based keyword search encryption for power data protection. High-Confidence Computing, 2023, 3(2):32-39. https://doi.org/10.1016/j.hcc.2023.100115

[3] Prasad S, Rao Y S. Designing secure data storage and retrieval scheme in cloud-assisted Internet-of-Drones environment. IEEE Internet of Things Journal, 2023, 11(8):13734-13751. https://doi.org/10.1109/JIOT.2023.3337265

[4] Liu W, Bai X, She W, Song X, Tian Z. Searchable encrypted data sharing scheme based on inverted index. Computer Engineering and Applications, 2023, 59(10):270-279. https://doi.org/10.3778/j.issn.1002-8331.2201-0159

[5] Ng Z Y, Salam I. Implementation of a blockchain-based searchable encryption for securing contact tracing data. Journal of Internet Technology, 2024, 25(2):241-254. https://doi.org/10.53106/160792642024032502007

[6] Liu P, He Q, Zhao B, Guo B, Zhai Z. Efficient multi-authority attribute-based searchable encryption scheme with blockchain assistance for cloud-edge coordination. Computers, Materials & Continua, 2023, 76(9):3325-3343. https://doi.org/10.32604/cmc.2023.041167

[7] Lu H, Xue X, Zhu Y, Chen C, Han H, Meng S, Lin H. Privacy-preserving SQL database driven by searchable encryption. Chinese Journal of Engineering, 2024, 46(11):2085-2098. https://doi.org/10.13374/j.issn2095-9389.2024.02.07.004

[8] Wang Z, Zhang Q, Meng L, Liu Y L. Secure content-based image retrieval scheme based on deep hashing and searchable encryption. CMC-Computers Materials & Continua, 2023, 75(3):6161-6184. https://doi.org/10.32604/cmc.2023.037134

[9] Souror S, Badawy M, El-Fishawy N. Secure query processing for smart grid data using searchable symmetric encryption. The Journal of Supercomputing, 2024, 80(16):24173-24211. https://doi.org/10.1007/s11227-024-06326-z

[10] Xie P, Li X, Feng T, Zhang M, Zhang P, Li P. A Trust and Risk Adaptive Access Control Model for Internet of Vehicles. International Journal of Network Security, 2024, 26(3):510-520. https://doi.org/10.6633/IJNS.202405_26(3).18

[11] Liu Y, Yang W, Wang Y, Liu Y. An access control model for data security sharing cross-domain in consortium blockchain. IET Blockchain, 2023, 3(1):18-34. https://doi.org/10.1049/blc2.12022

[12] Tong Q, Miao Y, Weng J, Liu X, Choo K K R, Deng R H. Verifiable fuzzy multi-keyword search over encrypted data with adaptive security. IEEE Transactions on Knowledge and Data Engineering, 2022, 35(5):5386-5399. https://doi.org/10.1109/TKDE.2022.3152033

[13] Geem D, Hercules D, Pelia R S, Venkateswaran S, Griffiths A, Noe J D. Progression of Pediatric Crohn's disease is Associated with anti-tumor necrosis factor timing and body Mass Index Z-Score normalization. Clinical Gastroenterology and Hepatology, 2024, 22(2):368-376. https://doi.org/10.1016/j.cgh.2023.08.042

[14] Chinthamu N, Karukuri M. Data science and applications. Journal of Data Science and Intelligent Systems, 2023, 1(1):83-91. https://doi.org/10.47852/bonviewJDSIS3202837

[15] Tian J, Lu Y, Li J. Lightweight searchable and equality-testable certificateless authenticated encryption for encrypted cloud data. IEEE Transactions on Mobile Computing, 2024, 23(8):8431-8446. https://doi.org/10.1109/TMC.2023.3348849

[16] Lu W. Construction of a Secure Sharing Model for Digital Educational Resources Using Blockchain and Cipher Policy Attribute Based Encryption in Smart Education. Informatica, 2024, 48(22):63-74. https://doi.org/10.31449/inf.v48i22.6627

[17] Yang Y, Zhang G, Li S, Liu Z. Offline/online attribute-based searchable encryption scheme from ideal lattices for IoT. Frontiers of Computer Science, 2024, 18(3):239-241. https://doi.org/10.1007/s11704-023-3128-3

[18] Xiong Y, Luo M X. Searchable encryption scheme for large datan sets in cloud storage environment. Radioengineering, 2024, 33(2):223-235. https://doi.org/10.13164/re.2024.0223

[19] Song J, Shen Z, Yu H, Lai R, Li Y, Wang Q, Li J. Secure and efficient multi-keyword fuzzy search over encrypted data on alliance chain. Recent Advances in Electrical & Electronic Engineering, 2024, 17(7):652-665.

https://doi.org/10.2174/0123520965251866230926050714

[20] Alyousif A A, Yassin A A. Improving Performance of searchable symmetric encryption through new information retrieval scheme. Iraqi Journal for Electrical and Electronic Engineering, 2024, 20(1):68-77. https://doi.org/10.37917/ijeee.20.1.7

[21] Hu Z, Deng L, Wu Y, Shi H, Gao Y. Secure and efficient certificateless searchable authenticated encryption scheme without random oracle for industrial internet of things. IEEE Systems Journal, 2023, 17(1):1304-1315. https://doi.org/10.1109/JSYST.2022.3197174

[22] Senthilkumar M, Murugan BS. Enhancing The Security Of An Organization From Shadow Iot Devices Using Blow-Fish Encryption Standard. Acta Informatica Malaysia. 2022; 6(1): 22-24. http//:doi.org/ 10.26480/aim.01.2022.22.24

[23] Talib E, Osipyan V. User Multi Group Key Distribution Using Secret Sharing with Circulate Matrices Based on Structures Pythagors Equation and Ecdh Key Exchange Protocol. Informatica, 2023, 47(5):127-136. https://doi.org/10.31449/inf.v47i5.4658

[24] Liu X H, Huang X Y, Wu W, Ning J T. Key-Policy Attribute-Based Encryption Based on SM9 and Its Fast Decryption. Journal of Computer Science and Technology, 2024, 47(5):971-986. https://doi.org/10.11897/SP.J.1016.2024.00971