mathematics

MDPI

Article

Enhancing Security and Efficiency: A Fine-Grained Searchable Scheme for Encryption of Big Data in Cloud-Based Smart Grids

Jing Wen [1], Haifeng Li [2,3,*], Liangliang Liu [4,*] and Caihui Lan [1]

[1] School of Information Engineering, Lanzhou City University, Lanzhou 730070, China; wj_83@lzcu.edu.cn (J.W.); lan_ch@lzcu.edu.cn (C.L.)
[2] School of Computer and Information Science, Qinghai University of Science and Technology, Xining 810016, China
[3] Department of Computer Technology and Applications, Qinghai University, Xining 810016, China
[4] School of Statistics and Information, Shanghai University of International Business and Economics, Shanghai 201620, China
* Correspondence: lihaifeng@qhu.edu.cn (H.L.); liangliang@suibe.edu.cn (L.L.)

Abstract: The smart grid, as a crucial part of modern energy systems, handles extensive and diverse data, including inputs from various sensors, metering devices, and user interactions. Outsourcing data storage to remote cloud servers presents an economical solution for enhancing data management within the smart grid ecosystem. However, ensuring data privacy before transmitting it to the cloud is a critical consideration. Therefore, it is common practice to encrypt the data before uploading them to the cloud. While encryption provides data confidentiality, it may also introduce potential issues such as limiting data owners' ability to query their data. The searchable attribute-based encryption (SABE) not only enables fine-grained access control in a dynamic large-scale environment but also allows for data searches on the ciphertext domain, making it an effective tool for cloud data sharing. Although SABE has become a research hotspot, existing schemes often have limitations in terms of computing efficiency on the client side, weak security of the ciphertext and the trapdoor. To address these issues, we propose an efficient server-aided ciphertext-policy searchable attribute-based encryption scheme (SA-CP-SABE). In SA-CP-SABE, the user's data access authority is consistent with the search authority. During the search process, calculations are performed not only to determine whether the ciphertext matches the keyword in the trapdoor, but also to assist subsequent user ciphertext decryption by reducing computational complexity. Our scheme has been proven under the random oracle model to achieve the indistinguishability of the ciphertext and the trapdoor and to resist keyword-guessing attacks. Finally, the performance analysis and simulation of the proposed scheme are provided, and the results show that it performs with high efficiency.

Keywords: searchable encryption; attribute-based encryption; server-aided decryption; trapdoor indistinguishability; random oracle model

MSC: 68P25

Citation: Wen, J.; Li, H.; Liu, L.; Lan, C. Enhancing Security and Efficiency: A Fine-Grained Searchable Scheme for Encryption of Big Data in Cloud-Based Smart Grids. *Mathematics* **2024**, *12*, 1512. https://doi.org/10.3390/math12101512

Academic Editor: Cheng-Chi Lee

Received: 13 March 2024
Revised: 6 May 2024
Accepted: 8 May 2024
Published: 13 May 2024

1. Introduction

The smart grid stands as a pivotal advancement in the future of energy, marking a transformative leap forward and modernization of traditional power systems. Compared to conventional grids, the smart grid harnesses advanced technologies and digital solutions to make the processes of transmitting, distributing, and monitoring electricity more intelligent, efficient, and reliable. It represents a critical pathway for the future evolution of power systems. Through continuous real-time monitoring, control, and optimization of power system operations, the smart grid enhances energy utilization efficiency and system stability. Moreover, big data technology offers robust support to the smart grid by gathering, storing, and analyzing vast data sets, unveiling patterns, trends, and potential issues within power

system operations. Harnessing big data analytics enables the smart grid to accurately forecast load demands, optimize energy distribution, and achieve intelligent maintenance and management of power supply equipment. Consequently, the integration of smart grid and big data not only elevates energy systems' intelligence and efficiency, but also bolsters efforts toward sustainable energy development and heightened energy security. Nevertheless, it faces security challenges, especially concerning cryptographic and network security issues. In the smart grid, safeguarding the security and privacy of data is paramount given the implications for energy supply stability, user privacy, and critical infrastructure operation. Hence, addressing security concerns in the smart grid demands vigilant attention and the implementation of robust cryptographic measures to protect system security. Cloud computing [1] provides ubiquitous access, high flexibility, low cost and scalability, making the cloud an attractive option for storing and managing data due to almost unlimited storage space and powerful processing capabilities. Outsourcing data storage to remote cloud servers presents an economical solution for enhancing data management within the smart grid ecosystem. Data sharing [2] is critical in data management. Attribute-based encryption has proven effective in enabling fine-grained access control and is considered as a valuable tool for realizing cloud data sharing. The concept of attribute-based encryption was first proposed by Sahai et al. [3]. Subsequently, Goyal et al. [4] discussed the differences and connections between key-policy attribute-based encryption (KP-ABE) and ciphertext-policy attribute-based encryption (CP-ABE), and proposed a KP-ABE scheme based on a tree access structure. At the same time, Bethencourt et al. [5] introduced the first CP-ABE scheme. In practice, users typically only need access to specific data rather than all data they can be permitted to access. Therefore, it is often necessary to control data access privileges and ensure encrypted data searchability in cloud computing environments. To achieve both encrypted data searchability and access control simultaneously, scholars have combined searchable encryption with attribute encryption and proposed numerous searchable attribute-based encryption schemes [6–16].

1.1. Motivation

We consider an example of the application of SABE for smart grids.

Smart grids integrate advanced communication and information technologies into the power system, enabling real-time monitoring and control of power flow, distribution, and management. Smart meters are a crucial component of smart grids, collecting and transmitting consumer electricity usage data as well as monitoring the status and load of the grid.

In such a system, ensuring the security of smart meter data is of the utmost importance. The SABE can be employed to securely store and control access to the data while allowing for authorized users to effectively search encrypted data without revealing sensitive information. For instance, in smart grids, smart meter data typically include vital information such as user details, electricity consumption, and grid status. By utilizing SABE, these data can be securely encrypted and associated with multiple attributes, such as "User ID", "Timestamp", and "Electricity Consumption", while grid status data may be labeled with "Geographic Location" and "Date".

Authorized users can search the encrypted data based on their access privileges. For instance, when an authorized user, such as an operator from a utility company, needs to inquire about a user's electricity consumption within a specific time period, they can utilize SABE to perform an encrypted search within the encrypted smart meter data. The system matches the encrypted data based on the search keywords provided by the user, which are within their permissions, such as the specific user's ID and a timestamp for a designated time period. The system then returns the matched ciphertext data as search results to the search user without revealing any other sensitive data information.

This application of SABE in smart grids not only ensures the security and privacy of data, but also enables efficient searching and access control of encrypted data, thereby enhancing the security and efficiency of smart grid systems.

Despite the significant benefits and conveniences to smart grids, SABE still faces two main challenges. One is that many existing SABE schemes are unable to resist keyword guessing attacks. The other is that many existing SABE schemes require expensive pairing operations which pose a formidable challenge to search efficiency. Thus, it is essential to design a novel SABE scheme that can simultaneously resist keyword guessing attacks and mitigate the decryption burden.

1.2. Contributions

To fill the above-mentioned gaps, in this paper, we propose a new searchable attribute-based encryption scheme (SA-CP-SABE) with the following contributions:

(1) We propose a searchable attribute-based encryption scheme (SA-CP-SABE) utilizing a ciphertext policy based on the tree access structure. Our SA-CP-SABE scheme has two distinctive features: first, it employs the same tree structure for both data encryption and keyword encryption, thereby ensuring consistent search rights and access permissions; second, it includes assisted decryption functionality, that is, it can enable a third party to verify whether a ciphertext matches a user's query based on the trapdoor. Additionally, the third party can leverage the search outcome to aid the user in decrypting the ciphertext. To the best of our knowledge, none of the existing SABE schemes in the smart grid can simultaneously resist keyword guessing attacks and mitigate the decryption burden so far.

(2) In the random oracle model, we prove that SA-CP-SABE satisfies indistinguishability between keyword ciphertexts and trapdoors. Compared with schemes in the literature [9–16], SA-CP-SABE can prevent keyword substitution attacks in both ciphertexts and trapdoors while also providing resistance against keyword guessing attacks. Furthermore, we conduct theoretical and experimental analyses of the SA-CP-SABE scheme, demonstrating its superior performance compared to that of the existing schemes.

1.3. Related Work

To enable user searching on the ciphertext domain, Song et al. [17] initially introduced the concept of searchable encryption and developed a symmetric searchable encryption scheme. In 2004, Boneh et al. [18] devised the first public key searchable encryption (PEKS) scheme. The PEKS scheme can delegate the search trapdoor to a third party to perform the search, while ensuring that the search trapdoor does not reveal any plaintext information. PEKS has attracted significant research attention in the realm of information security. Literature [19,20] discusses the keyword guessing attack based on the fact that the keyword space is much smaller than the key space in practical applications. Additionally, literature [21] explores the utilization of server assistance to enhance the efficiency of the scheme and proposes a server-aided PEKS scheme.

In 2005, Sahai et al. [3] initially introduced the concept of attribute-based encryption (ABE) where ciphertexts and keys are associated with sets of descriptive attributes. This allows a ciphertext to be decrypted by multiple users, overcoming the traditional limitation of decrypting a ciphertext with only a single key. Since the detailed work on attribute encryption provided by Goyal et al. [4] and Bethencourt et al. [5], attribute-based encryption has attracted widespread attention. ABE schemes have been applied in various domains such as video-on-demand [22], electronic medical health record access [23], and social networking site access operations [24]. To meet the data searchability requirement in data sharing environments, Li et al. [25] proposed a ciphertext policy-based searchable attribute-based encryption scheme by integrating ABE and PEKS concepts. In recent years, with the maturation of cloud computing technology, numerous searchable attribute-based encryption schemes have been proposed, providing ideal application scenarios for searchable attribute-based encryption [6–16]. However, searchable attribute-based encryption means that a keyword index must be searchable by a group of users. This leads to its construction method being more complex than traditional public key searchable schemes. From a construction perspective, the searchable attribute-based encryption schemes mentioned above can generally be divided into two categories. One category [6–8] combines

attribute encryption with public-key searchable encryption. Specifically, it encrypts data using attribute encryption to achieve access control, and then utilizes traditional searchable encryption techniques to process keywords, thus enabling ciphertext searchability. However, this approach may lead to the separation of data access rights and search rights. In other words, it could expose keyword information (which may be sensitive) during searches conducted by users without data access permissions.The other category [9–16] utilizes attribute encryption techniques to regulate user search privileges with a construction process similar to attribute encryption. However, these schemes possess certain drawbacks. The ciphertext models featuring indistinguishable keywords in the literature [9–12] are relatively weak as these security models lack the capability to perform trapdoor queries. In other words, these security models assume that attackers cannot access the trapdoor. This assumption is only suitable for scenarios where the search is conducted by the data owner themself or a fully trusted third party. Clearly, such security models are inadequate in a cloud computing environment where ciphertext searches are primarily conducted by a semi-trusted cloud. Although the security models in the literature [13–16] incorporate trapdoor queries, their schemes fail to meet the defined security criteria because the keywords in the ciphertexts can be replaced with other keywords. More specifically, after obtaining keyword w and its corresponding trapdoor through a trapdoor query, an attacker can carry out a keyword-guessing attack by modifying the attacked ciphertext (associated with keyword w) into another ciphertext (associated with keyword $w_{\prime}$) and then using the trapdoor for verification. Consequently, during the security game's challenge phase, the attacker can win the game by altering the keywords of the challenging ciphertext to another valid ciphertext contained within the intercepted trapdoor. Additionally, in the schemes presented in the literature [9–16], the keywords in the trapdoor can also be substituted with other keywords, thus failing to prevent insider keyword guessing attacks [26].

ABE schemes often involve a significant number of pairing and exponential operations, which means that the devices at the user side need more computations to share or access the data. This is obviously unacceptable for some devices with limited computational resources or in scenarios with high real-time demands. To meet the real-time encryption requirements of certain applications, scholars have recognized this issue and employed offline/online encryption techniques [9,11]. Additionally, outsourced decryption techniques [12,14,27] have been utilized to mitigate the decryption burden on the user side. A summary of the related work is provided in Table 1, where "$\surd$" indicates that the specified scheme (row) satisfies the security property (column), "$\times$" indicates that the specified scheme (row) does not satisfy the security property (column), and "-" indicates that the attack is not considered in this work.

Table 1. Security Analysis for Existing Schemes.

Schemes	Consistent with Data Access and Search Permissions	Supporting Trapdoor Queries	Based on the Strategy	Server-Aided Decryption
Literature [6]	$\times$	$\surd$	CP-ABE	$\surd$
LFSE [7]	$\times$	$\surd$	CP-ABE	$\surd$
LSABE [8]	$\times$	-	CP-ABE	$\surd$
DSF [9]	$\surd$	$\surd$	CP-ABE	$\surd$
ABKS-HD [10]	$\surd$	$\surd$	CP-ABE	$\surd$
Literature [12]	$\surd$	$\times$	CP-ABE	$\surd$
Literature [11]	$\surd$	$\times$	CP-ABE	$\surd$
Literature [13]	$\surd$	$\surd$	CP-ABE	$\times$
LABSE [16]	$\surd$	$\surd$	KP-ABE	$\surd$
KSF-OABE [14]	$\times$	$\surd$	KP-ABE	$\surd$
FKS-HPABE [15]	$\surd$	$\surd$	CP-ABE	$\times$
Literature [25]	-	$\surd$	KP-ABE	$\times$

1.4. Organization

The remainder of the paper is organized as follows. We present some preliminaries in Section 2. In Section 3, we define the system model and the security model of our scheme. In Section 4, we propose the concrete construction of the proposed SA-CP-SABE scheme. In Section 5, we provide the security analysis as well as the performance evaluation of our scheme. Finally, we draw the conclusion of the whole paper in Section 6.

2. Preliminaries

2.1. Notations

Table 2 provides the summary of notations used in our proposed scheme.

Table 2. Notations and their meanings.

Notation	Meaning	Notation	Meaning
q	a large prime	$params$	system parameters
Z_q, Z_q^*	Z_q denotes the residue group modulo, $Z_q^* = Z_q/0$	T	the access structure
G_1, G_2	two multiplicative cyclic groups with the equal prime order p	x	a node of the access tree T
e	the bilinear pair map between the two groups	at_x	an attribute associated with the leaf node x in the access tree
g	the generator of the group G_1	CT/CT'	the original ciphertext/the transformed ciphertext
g_1, g_2	two elements of the group G_1	S	the attribute set
MK/SK	the system master key/the user's private key	$Trap_w$	the search trapdoor

2.2. Bilinear Pairs

We let G_1 and G_2 be the cyclic multiplicative groups of order q where q is a large prime, and we let $e : G_1 \times G_1 \rightarrow G_2$ be a bilinear map between the two groups satisfying the following conditions:

(1) Bilinearity: $\forall a, b \in Z_q^*$, $\forall g_1, g_2 \in G_1$, such that $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ holds.
(2) Non-degeneracy: $\exists g_1, g_2 \in G_1$, such that $e(g_1, g_2) \neq 1$, where 1 is the unit element of G_2.
(3) Computability: For $\forall g_1, g_2 \in G_1$, there exist efficient algorithms that can compute $e(g_1, g_2)$.

2.3. Difficult Assumptions

Assumption 1. *Let $B = (q, G_1, G_2, e)$ be a bilinear group system and g be a generator of group G_1. Given $g, g^{\gamma_1}, g^{\gamma_2}, g^{\gamma_3}, g^{\gamma_3^2}, R \in G_2$, we determine whether R is equal to $e(g,g)^{\gamma_1\gamma_2\gamma_3}$. Drawing on the analysis of the literature [26,28], the difficult assumption above translates into proving whether $F = \gamma_1\gamma_2\gamma_3$ is independent of $\{P = (1, \gamma_1, \gamma_2, \gamma_3, \gamma_3^2), Q = (1)\}$, i.e., proving that there are no coefficients $\{x_{ij}\}, \{y_{ij}\} \in Z_q$ satisfying equation $\sum x_{ij} p_i p_j = \sum y_k F_k$. Obviously, due to $F = \gamma_1\gamma_2\gamma_3$, there always exists an expression in $\{x_{ij}\}, \{y_k\}$ with some coefficient of $\gamma_1, \gamma_2, \gamma_3, \gamma_3^2$ no matter how $p_i, p_j \in P$ is combined to take values. Therefore, determining whether R is equal to $e(g,g)^{\gamma_1\gamma_2\gamma_3}$ is a difficult problem.*

Based on the above analysis, the following difficult problem can be obtained.

Assumption 2. *We let $B = (q, G_1, G_2, e)$ be a bilinear group system and $g \in G_1$ be a generator. Given $(g \in G_1, g^{\gamma_1}, g^{\gamma_2}, R \in G_1)$, we determine whether R is equal to $e(g,g)^{\gamma_1\gamma_2^2}$.*

2.4. Access Tree

Definition 1 (Access Tree). *An access tree is used to describe an access structure. Each intermediate node of tree x represents a relation function, which can be "or", "with", or other threshold.*

Assuming that num_x denotes the number of children of a node and k_x represents its threshold, $0 \leq k_x \leq num_x$ is satisfied. Each leaf node of tree x represents an attribute item and threshold $k_x = 1$. In implementation, it is generally necessary to adopt a top-down approach to select a polynomial of degree $d_x = k_x - 1$, q_x for each node x, satisfying $q_x(0) = q_{p(x)}(index(x))$. Here, $p(x)$ denotes the parent of node x, and $index(x)$ is the index of node x.

We let T_x denote the subtree of T with node x as the root. When x is a leaf node, $keyword(x) = at_x$ represents the attribute value of the output leaf node. We say that attribute set S satisfies T_x (denoted by $T_x = 1$) if and only if the following two conditions are met:

(1) When x is a leaf node, $T_x(S) = 1$ if and only if at_x is an attribute in attribute set S.
(2) When x is an internal node, we compute $T_z(S)$ for each child z of x. $T_x(S) = 1$ if and only if there are at least k_x children.

3. System Model and Security Model

3.1. System Model

The system consists of a Key Generation Center (KGC), a Data Owner (DO), a Cloud Server Provider (CSP), and a Data User (DU). The system model is illustrated in Figure 1.

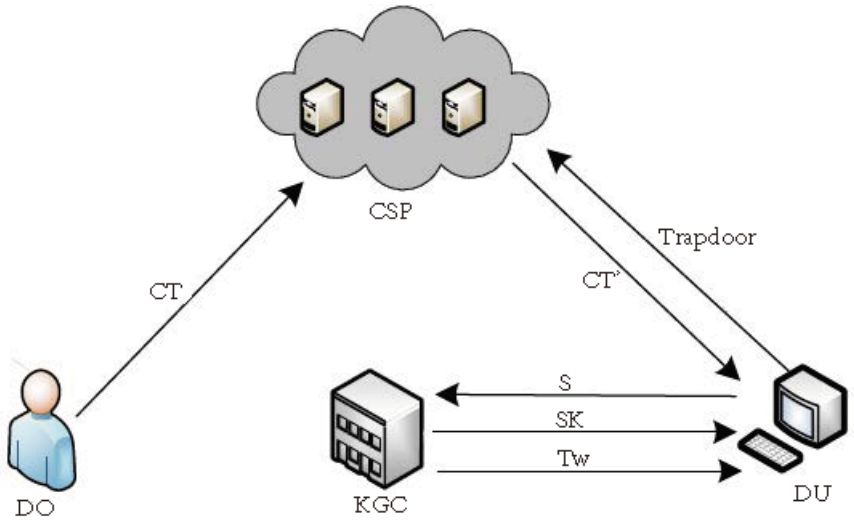

Figure 1. System model of the SA-CP-SABE scheme.

KGC: Responsible for generating private keys for data users and assisting them in generating trapdoors.

DO: The DO acts as the ciphertext generator, i.e., it performs encryption of keywords and uploads the encrypted ciphertext to the cloud server.

CSP: The CSP stores the ciphertext and conducts ciphertext search and transform tasks.

DU: The user of the data who generates the trapdoor and delegates it to the cloud for data search and performs decryption operations.

Definition 2. *The proposed SA-CP-SABE scheme comprises the following six probabilistic polynomial-time (PPT) algorithms:*

(1) System Initialization: This PPT algorithm is executed by the KGC to initialize the global system. Taking security parameter λ as input, it outputs the system master key, MK, and the system public parameters, params.
(2) Encryption: This PPT algorithm is executed by the DO to perform encryption. Taking the system public parameters, params, data m, keywords w_m and access structure T as input, it outputs the ciphertext, CT, which is then uploaded to the cloud.

(3) *User Private Key Generation: This PPT algorithm is executed by the KGC to generate a user private key. Taking the system public parameters, params, the system master key, MK, and attribute set S as input, it outputs a user private key, SK.*
(4) *Trapdoor Generation: This PPT algorithm is executed by the DU to generate a search trapdoor. Taking the system public parameters, params, the user private key, SK, and keyword w as input, it outputs the trapdoor,* $Trap_w$.
(5) *Search and Transformation: This PPT algorithm is executed by the CSP to perform search and transformation operations. Taking the system public parameters, params, the keyword ciphertext, CT, and trapdoor* $Trap_w$ *as input, it outputs the search result and server-aided decrypted ciphertext* CT' *and returns them to the search user.*
(6) *Decryption: This PPT algorithm is executed by the DU to perform decryption. Taking the system public parameters, params, the ciphertext,* CT'*, and private key SK as input, it outputs plaintext data m.*

3.2. Security Model

In this section, we define two security models for our SA-CP-SABE scheme to specify the capabilities and possible actions of the attacker by a game involving two participants: the challenger and the attacker. In the security model, the challenger assumes a dual role. First, the challenger interacts with the attacker, responding to queries that essentially serve to ascertain the attacker's capabilities and the type of information they can obtain. Second, the challenger acts as a problem solver for challenging tasks, leveraging the attacker's capabilities and the information provided during the challenge phase to tackle difficult problems. In searchable encryption schemes, two main security properties are typically considered: ciphertext privacy security (IND-CKA) and trapdoor privacy security (IND-KGA).

Ciphertext privacy security means that the ciphertext of a keyword does not reveal any information about the keyword to an unauthorized attacker. The specific security model is as follows:

Definition 3 (IND-CKA)**.** *Assuming* $\mathscr{A}_1$ *is the attacker and* $\mathscr{C}$ *is the challenger, the IND-CKA security model is defined by security game* $Game_{CKA}$ *between the challenger,* $\mathscr{C}$*, and the attacker,* $\mathscr{A}_1$*. The game,* $Game_{CKA}$*, is described as follows:*

Initialization: Challenger $\mathscr{C}$ *executes the system initialization algorithm, obtains system parameters params and master key MK, and offers params to attacker* $\mathscr{A}_1$.

Phase 1: Attacker $\mathscr{A}_1$ *can initiate the following queries:*

Hash queries: An attacker can, at any time, initiate hash queries of any message, and the challenger returns the corresponding hash value.

Key queries: Upon receiving a set of attributes S by the attacker, the challenger simulates private key sk and sends it to the attacker.

Trapdoor Queries: Upon receiving a set of attributes S and keyword w from the attacker, the challenger simulates trapdoor $Trap_w$ *and returns it to the attacker.*

Challenge phase: At the end of Phase 1 queries, the attacker outputs $\mathscr{A}_1$ $(m_0, w_0), (m_1, w_1)$ *with the same length and an access tree* T_* *(where the set of attributes required to satisfy its access rights has not been queried by the key queries). Challenger* $\mathscr{C}$ *randomly selects* $b \in \{0,1\}$*, performs the encryption algorithm on* (m_b, w_b)*, and returns ciphertext* CT_* *to the attacker.*

Phase 2: The attacker continues to initiate the same queries as in Phase 1 with the following restrictions:

(1) *If attribute set S satisfies access tree* T_**, key queries of S are prohibited.*
(2) *If attribute set S satisfies access tree* T_**, trapdoor queries with* (S, w_1^*) *and* (S, w_2^*) *are prohibited.*

Guess: At the end of the game, attacker $\mathscr{A}_1$ *outputs* $b' \in \{0,1\}$*; if* $b' = b$*, the attacker wins the game.*

Attacker $\mathscr{A}_1$ *has the advantage of winning the game defined as* $Adv_A^{Game_{CKA}}(\lambda) = \left|\Pr[b = b^{'}] - \frac{1}{2}\right|$.

If advantage $Adv_A^{Game_{CKA}}(\lambda)$ *of attacker* $\mathscr{A}_1$ *in winning the game is negligible, the scheme satisfies IND-CKA security.*

Trapdoor privacy means that trapdoors do not reveal any information about relevant keywords to unauthorized attackers. The specific security model is as follows.

Definition 4 (INA-KGA). *Assuming that* $\mathscr{A}_1$ *is the attacker and* $\mathscr{C}$ *is the challenger, the INA-KGA security model can be defined by security game* $Game_{KGA}$ *between the challenger,* $\mathscr{C}$*, and the attacker,* $\mathscr{A}_1$*, and game* $Game_{KGA}$ *is described as follows:*

Initialization: Challenger $\mathscr{C}$ *executes the system initialization algorithm, obtains the system parameters, params, and the master key, and offers params to attacker* $\mathscr{A}_1$.

Phase 1: The attacker at $\mathscr{A}_1$ *can initiate the following queries:*

Hash queries: An attacker can, at any time, initiate hash queries of any message, and the challenger returns the corresponding hash value.

Key queries: Upon receiving a set of attributes S from the attacker, the challenger simulates the private key, sk, and sends it to the attacker.

Trapdoor Queries: Upon receiving a set of attributes S and keyword w from the attacker, the challenger simulates trapdoor $Trap_w$ *and returns it to the attacker.*

Challenge phase: The attacker selects the given set of challenge attributes S_* *(no private key queries are queried) and keyword* $\{w_0, w_1\}$*. The challenger randomly selects* $b \in \{0, 1\}$ *and returns challenge trapdoor Tr. Here, data m,* $w \in \{w_0, w_1\}$*, and access structure T (*S_* *meets T) are never encrypted.*

Phase 2: The attacker continues to initiate queries as in Phase 1 with the following restrictions:

(1) Encryption of data m, $w \in \{w_0, w_1\}$*, and access structures T (*S_* *satisfies T) is not permitted.*

(2) Private key queries on attribute set S_* *are not permitted.*

Guess: At the end of the game, attacker $\mathscr{A}_1$ *outputs* $b^{'} \in \{0, 1\}$*. If* $b^{'} = b$*, the attacker wins the game.*

Attacker $\mathscr{A}_1$ *has the advantage of winning the game defined as* $Adv_A^{Game_{KGA}}(\lambda) = \left|\Pr[b = b^{'}] - \frac{1}{2}\right|$.

If advantage $Adv_A^{Game_{KGA}}(\lambda)$ *of attacker* $\mathscr{A}_1$ *in winning the game is negligible, the scheme satisfies INA-KGA security.*

4. Construction of the SA-CP-SABE Scheme

Next, we present the design of our SA-CP-SABE scheme. There are six polynomial-time algorithms described as follows:

1. System initialization: this algorithm selects two multiplicative groups (G_1, G_2) with the same prime order q. We define bilinear map $e : G_1 \times G_1 \to G_2$, and g is the generator of group G_1. We choose four random numbers $a, b, d, u \in Z_q^*$ and compute $f = g^d$, $K_1 = e(g, g)^a$ and $K_2 = e(g, g)^b$. We define hash function $H : \{0, 1\}^* \to G_1$. Finally, PKG publishes system parameters $params = \{G_1, G_2, K_1, K_2, H, g^u\}$ and the secret system master key $MK = \{a, b, d, u\}$. We use $L_{i,s} = \prod_{l \in I, l \neq i} (x - l)/(i - l)$ to denote the Lagrange coefficients of $i \in Z_q$ and $S = \{s_1, s_2, \cdots, s_m \in Z_q\}$.
2. Encryption: Given data $m \in \{0, 1\}^*$ and keyword $w_m \in G_2$, the DO selects a symmetric encryption algorithm (Enc, Dec) and encryption key ck and encrypts m with algorithm Enc and key ck represented as $C_m = Enc_{ck}(m)$. Then, we define access structure T and encrypt ck and keyword w_m according to T in the following steps:
   (1) We randomly select $r \in Z_q^*$ and calculate $C_{ck} = ck \cdot K_1^r$, $C_w = e(H(w)^{'}, g^u) \cdot K_2^r$, $C = f^r$.
   (2) Using a top-down approach, we start from the root node, and for each node x, we select polynomial q_x of degree $d_x = k_x - 1$. When x is the

root node, we make $q_x(0) = r$. Otherwise, we let $q_x(0) = q_{p(x)}(index(x))$ where $p(x)$ is the parent of node x and $index(x)$ is the index of node x. We let Y denote the set of all leaf nodes. Each leaf node y corresponds to a specific attribute value, which is denoted as at_y. We compute $CT_{at} = \left\{\forall at_y \in Y : C_y^1 = g^{q_y(0)}, C_y^2 = H(at_y)^{q_y(0)}\right\}$.

(3) Finally, we upload ciphertext $\{T, C_m, C_{ck}, C_w, C, C_{at}\}$ to the cloud.

3. User Private Key Generation: Once the KGC receives a request from a data user (with attribute set S) to generate a key, it first randomly selects $s \in Z_p^*$ and calculates $D_d = g^a g^s$ and $D_s = g^{\frac{b+s}{d}}$. Then, we randomize $r_i \in Z_q^*{}_{i=1,2,\cdots,|S|}$ and calculate $L_S = \left\{\forall at_i \in S : \ D_i = g^s H(at_i)^{r_i}, D_i^{'} = g^{r_i}\right\}$. Finally, we send $SK = \{D_d, D_s, L_S\}$ to the data user.
4. Trapdoor Generation: When the data user requests the search permission of keyword w from the KGC, the KGC randomly selects k and calculates $T_w = H(w)^{\frac{u}{d}} g^{\frac{k}{d}}$ and g^k, and returns it to the data user. After receiving it, the data user calculates the trapdoor,

$$Trap_w = \left\{ \begin{array}{l} Tr_1 = D_s \cdot T_w, \\ Tr_S^{'} = \left\{\forall at_i \in S : \ E_i = g^k D_i, E_i^{'} = D_i^{'}\right\} \end{array} \right\}, \tag{1}$$

and sends it to the cloud server.
5. Search and Transform: Upon receiving $Trap_w$, the cloud first verifies whether the user's attribute set S satisfies access control tree T in ciphertext CT. If not, it returns $\perp$. Otherwise, the search is conducted as follows:

(1) The cloud defines two recursive algorithms, $Test(CT, Trap_w, x)$ and $CS(CT, x)$, which take as input ciphertext CT, trapdoor $Trap_w$, attribute set S, and node x in access tree T and return the result as follows. The actual attribute $at_x = attr(x)$ is used to represent leaf node x.

(i) If x is a leaf node and $at_x = attr(x) \in S$, then we define

$$\begin{aligned} Test(CT, Trap_w, x) &= \frac{e(E_i, C_x^1)}{e(E_i^{'}, C_x^2)} \\ &= \frac{e(g^k g^r H(at_x)^{s_x}, g^{q_x(0)})}{e(g^{s_x}, H(at_x)^{q_x(0)})} \\ &= e(g,g)^{(k+s)q_x(0)} \end{aligned} \tag{2}$$

$$CS(CT, x) = C_x^1 \tag{3}$$

(ii) If x is a leaf node and $at_x = attr(x) \notin S$, then we define $Test(CT, Trap_w, x) = \perp$, $CS(CT, x) = \perp$.

(iii) If x is a non-terminal node, then we create the set $A_x = \{z | Test(CT, Trap_w, z) \neq \perp\}$ where z is the left child of node x. When $|A_x|$ is less than the threshold k_x, we make $Dec(CT, SK, x) = \perp$. Otherwise, we choose a subset of A_x that satisfies $\left|S_x{}^{'}\right| = k_x$, $S_x{}^{'} \subseteq A_x$ and denote the set $\{i = index(z) | z \in S_x^{'}\}$ by S_x. Finally, we define

$$\begin{aligned} &Test(CT, Trap_w, x) \\ &= \prod_{z \in S_x{}^{'}} Test(CT, Trap_w, z)^{L_{i,S_x}(0)} \\ &= e(g,g)^{(k+s)q_x(0)} \end{aligned} \tag{4}$$

$$\begin{aligned} & CS(CT, x) \\ &= \prod_{z \in S_x{}'} CS(CT, z)^{L_{i,S_x}(0)} \\ &= g^{q_x(0)} \end{aligned} \tag{5}$$

(2) The cloud calls $Test(CT, Trap_w, R)$ and $CS(CT, R)$ to obtain $e(g,g)^{(k+s)r} = Test(CT, Trap_w, R)$ and $g^r = CS(CT, R)$, respectively, where R is the root node.

(3) Ciphertext Verification.
The cloud server verifies whether $e(Tr_1, C) = C_w \cdot Test(CT, Trap_w, R)$ holds.

(i) If it holds, it implies that the keyword of the ciphertext matches the keyword in the trapdoor. Therefore, the cloud returns the ciphertext as follows:

$$CT' = \{C_m, C_{ck}, C_R = Test(CT, Trap_w, R), C_s = CS(CT, R)\} \tag{6}$$

(ii) If it does not hold, it indicates that the ciphertext is not the one searched by the data user.
In fact, here are

$$\begin{aligned} e(Tr_1, C) &= e(D_s \cdot T_w, f^r) \\ &= e(g^{\frac{b+s}{d}} H(w)^{\frac{u}{d}} g^{\frac{k}{d}}, g^{dr}) \\ &= e(g^{b+s+k} H(w)^u, g^r) \\ &= e(H(w)^u, g^r) e(g,g)^{br} e(g,g)^{(k+s)r} \\ &= C_w \cdot Test(CT, Trap_w, R) \end{aligned} \tag{7}$$

6. Decryption: The data user receives CT' and calculates $ck = \frac{C_{ck} \cdot C_R}{e(D_d g^k, C_s)}$. Finally, the data user can obtain the plaintext $m = Dec_{ck}(C_m)$.

5. Analysis of the SA-CP-SABE Scheme

5.1. Security Analysis

Theorem 1. *Under the random model, if there exists an attacker A who can win game $Game_{CKA}$ with probability ε in polynomial time, then there exists a challenger $\mathscr{C}$ who can solve the hard problem defined in Definition 1 with probability $\frac{\varepsilon}{2}$ in polynomial time.*

Proof. Given an instance $(g \in G_1, g^{\gamma_1}, g^{\gamma_2}, g^{\gamma_3}, g^{\gamma_3^2}, R \in G_2)$, challenger $\mathscr{C}$ performs the $Game_{CKA}$ game with attacker A and solves the hard problem in Definition 1 by using attacker A's ability to determine whether R is equal to $e(g,g)^{\gamma_1\gamma_2\gamma_3}$ as follows:

Initialization: Challenger $\mathscr{C}$ selects four random numbers $\lambda_1, \lambda_2, \lambda_3, \lambda_4 \in Z_q^*$ and calculates $f = (g^{\gamma_1})^{\lambda_3}$, $K_1 = e(g^{\gamma_1}, g)^{\lambda_1}$, and $K_2 = e(g^{\gamma_1}, g^{\gamma_2})^{\lambda_2}$. We define hash function $H : \{0,1\}^* \to G_1$. Finally, PKG issues system parameter $params = \{G_1, G_2, h, K_1, K_2, H, f, (g^{\gamma_1})^{\lambda_4}\}$. In fact, the system parameters set in this way can be regarded as the system master key owned by KGC, denoted as $MK = a = \lambda 1\gamma 1, b = \lambda 2\gamma 1\gamma 2, d = \lambda 3\gamma 1, u = \lambda 4\gamma_1$.

Phase 1: Attacker A adaptively initiates the following queries:

$H(w)$ queries: Upon receiving w from the attacker, challenger $\mathscr{C}$ first looks up list HL, and if $(w, H_w) \in HL$, returns H_w. Otherwise, it randomly selects $r \in Z_q^*$, computes and returns $H_w = g^r$, and writes (w, H_w, r) to list HL.

Key queries: Upon receiving a set of attributes S from the attacker, challenger $s \in Z_p^*$, randomly computes $D_d = g^{\lambda_1\gamma_1}(g^{\gamma_1})^s = g^a g^{\gamma_1 s}$ and $D_s = g^{s\lambda_3^{-1}}(g^{\gamma_2})^{\lambda_2\lambda_3^{-1}} = g^{\frac{b+\gamma_1 s}{d}}$. Then, it randomly selects $r_i \in Z_q^*{}_{i=1,2,\cdots,|S|}$ and computes

$$L_S = \begin{Bmatrix} \forall at_i \in S: \ D_i = (g^{\gamma_1})^s g^{r_i r_{at_i}} = g^{\gamma_1 s} H(at_i)^{r_i}, \\ D_i^{'} = g^{r_i} \end{Bmatrix}. \tag{8}$$

Finally, we send $SK = \{D_d, D_s, L_S\}$ to the user. Here, r_{at_i} satisfies $H(at_i) = g^{r_{at_i}}$ and can be obtained by querying $H(at_i)$.

Trapdoor queries: Given an attribute set S and keyword w by the attacker, the challenger first performs key queries with attribute set S to obtain $SK = \{D_d, D_s, L_S\}$. We randomly select k and compute $T_w = g^{r_w \lambda_4 \lambda_3^{-1}} g^k = H(w)^{\frac{u}{d}} g^{\frac{k \times d}{d}}$ and $(g^{\gamma_1})^{\lambda_3 k} = g^{d \times k}$. Finally, we compute the trapdoor as

$$Trap_w = \begin{Bmatrix} Tr_1 = D_s \cdot T_w, \\ Tr_S^{'} = \left\{ \forall at_i \in S: \ E_i = g^{d \times k} D_i, E_i^{'} = D_i^{'} \right\} \end{Bmatrix} \tag{9}$$

and return it to the attacker. Here, r_w satisfies $H(w) = g^{r_w}$, which can be obtained by querying $H(w)$.

Challenge phase: Once the decision is made to end the queries in Phase 1, the attacker is given $A\ (m_0, w_0), (m_1, w_1)$ with the same length and an access tree T_* (the set of attributes required to satisfy its access rights is not interrogated by the key). Challenger $\mathscr{C}$ randomly selects $b \in \{0, 1\}$ and $r_* \in Z_q^*$, obtains $q_y(0)$ by the method in the encryption algorithm based on the access tree, T_*, and returns the following ciphertext:

$$\begin{Bmatrix} T_*, C_m = E_{ck}(m_b), \\ C_{ck} = ck \cdot e(g^{\lambda_1 \gamma_1}, g^{\gamma_3})^{r_*} = ck \cdot K_1^{\gamma_3 r_*}, \\ C_w = (g^{r_w \gamma_3}, g^u) R^{r_*} = (H(w_b), g^u) R^{r_*}, \\ C = (g^{\gamma_3^2})^{\lambda_3 r_*} = f^{\gamma_3 r_*}, \\ CT_{at} = \begin{Bmatrix} \forall at_y \in Y: C_y^1 = (g^{\gamma_3})^{q_y(0)}, \\ C_y^2 = (g^{\gamma_3})^{k r_{at_y} q_y(0)} = H(at_y)^{\gamma_3 q_y(0)} \end{Bmatrix} \end{Bmatrix} \tag{10}$$

Phase 2: The attacker continues to initiate the same queries as in Phase 1, with the following restrictions:

(1) If attribute set S satisfies access tree T_*, key queries of S are prohibited.
(2) If attribute set S satisfies access tree T_*, trapdoor queries with (S, w_1^*) and (S, w_2^*) are prohibited.

Guess: At the end of the game, attacker A outputs $b^{'} \in \{0, 1\}$, and if $b^{'} = b$, the attacker wins the game.

Obviously, when $R = e(g, g)^{\gamma_1 \gamma_2 \gamma_3}$, the above ciphertext is a legitimate ciphertext. Assuming that the attacker has the advantageous attack scheme of ε, when the ciphertext is valid, the attacker can guess it correctly with the probability of $\frac{1}{2} + \varepsilon$. When $R \neq e(g, g)^{\gamma_1 \gamma_2 \gamma_3}$, which is some random number to the attacker, the attacker can guess accurately with probability $\frac{1}{2}$. Therefore, the challenger also has $\frac{\varepsilon}{2}$ probability of solving the hard problem in Definition 1. □

Theorem 2. *Under the stochastic prediction model, if there exists an attacker A who can win with* $Game_{KGA}$ *ε probability in polynomial time, then there exists a challenger* $\mathscr{C}$ *who can win the hard problem in Definition 2 with* $\frac{\varepsilon}{2}$ *probability in polynomial time.*

Proof. Given instance $(g \in G_1, g^{\gamma_1}, g^{\gamma_2}, R \in G_1)$, challenger $\mathscr{C}$ performs the $Game_{KGA}$ game with attacker A and uses the difficult problem in the definition of attacker A's ability to determine whether R is equal to $g^{\gamma_1 \gamma_2^2}$, as follows:

Initialization: Challenger $\mathscr{C}$ selects four random numbers $\lambda_1, \lambda_2, d, u \in Z_q^*$ and calculates $f = g^d$, $K_1 = e(g^{\gamma_1}, g)^{\lambda_1}$ and $K_2 = e(g, g^{\gamma_2})^{\lambda_2}$. We define hash function $H: \{0, 1\}^* \rightarrow$

G_1. Finally, PKG issues system parameter $params = \{G_1, G_2, h, K_1, K_2, H, f, g^u\}$. In fact, the system parameters set in this way can be regarded as PKC with the system master key as $MK = \{a = \lambda_1\gamma_1, b = \lambda_2\gamma_2, d, u\}$.

Phase 1: Attacker A adapts to initiate the following queries:

$H(w)$ The query: Given w by the attacker, challenger $\mathscr{C}$ first looks up list HL, and if $(w, H_w) \in HL$, it returns H_w. Otherwise, it randomly selects $r \in Z_q^*$, computes and returns $H_w = g^r$, and appends (w, H_w, r) to list HL.

Key queries: The attacker is given a set of attributes S, challenger $s \in Z_p^*$ randomly computes $D_d = (g^{\gamma_1})^{\lambda_1} g^s = g^a g^s$ and $D_s = ((g^{\gamma_2})^{\lambda_2} g^s)^{d^{-1}} = g^{\frac{b+s}{d}}$. Then, it randomly selects $r_i \in Z_{q\ i=1,2,\cdots,|S|}^*$, and computes $L_S = \left\{\forall at_i \in S:\ D_i = g^s g^{r_i r_{at_i}} = g^s H(at_i)^{r_i}, D_i^{'} = g^{r_i}\right\}$. Finally, it sends $SK = \{D_d, D_s, L_S\}$ to the user. Here, r_{at_i} satisfies $H(at_i) = g^{r_{at_i}}$ and can be obtained by asking $H(at_i)$.

Trapdoor queries: Given an attribute set S and keyword w by the attacker, the challenger first performs key queries with attribute set S to obtain $SK = \{D_d, D_s, L_S\}$. It randomly selects k and computes $T_w = (g^{\gamma_w u} g^k)^{d^{-1}} = H(w)^{\frac{u}{d}} g^{\frac{k}{d}}$ and g^k, and finally computes the trapdoor,

$$Trap_w = \begin{Bmatrix} Tr_1 = D_s \cdot T_w, \\ Tr_S^{'} = \left\{\forall at_i \in S:\ E_i = g^k D_i, E_i^{'} = D_i^{'}\right\}, \end{Bmatrix} \tag{11}$$

and returns it to the attacker. Here, r_w satisfies $H(w) = g^{r_w}$, which can be obtained by interrogating $H(w)$.

Challenge phase: The attacker selects the given set of challenge attributes S_* (no private key queries performed) and keyword $\{w_0, w_1\}$. The challenger randomly selects $b \in \{0, 1\}$ and returns the challenge trapdoor Tr. Here, data m, $w \in \{w_0, w_1\}$ and access structure T (S_* meets T) are never encrypted. The specific challenge trapdoor Tr is generated as follows:

(1) We randomly select $l, r_i \in Z_{q\ i=1,2,\cdots,|S_*|}^*$ and calculate

$$Tr_{S_*}^{'} = \begin{Bmatrix} \forall at_i \in S_*:\ D_i = R^l g^{r_i r_{at_i}} = R^l H(at_i)^{r_i}, \\ D_i^{'} = g^{r_i} \end{Bmatrix}. \tag{12}$$

(2) We calculate $Tr_1 = ((g^{\gamma_2\lambda_2}) H(w_b)^u)^{d^{-1}}$.

(3) We return $Trap_{w_b} = \left\{Tr_1, Tr_{S_*}^{'}\right\}$.

Phase 2: The attacker continues to initiate the queries as in Phase 1 with the following restrictions:

(1) It is not possible to encrypt data m, $w \in \{w_0, w_1\}$ and access structures T (S_* meets T).

(2) Private key queries cannot be performed on attribute set S_*.

Guess: At the end of the game, attacker A outputs $b' \in \{0, 1\}$, if $b^{'} = b$. The attacker wins the game.

Obviously, considering random number s in key generation and k in trapdoor generation as being opposite to each other (i.e., $s = -k = -l\gamma_1\gamma_2$), and when $R = g^{\gamma_1\gamma_2^2}$, the trapdoor described above is a legitimate one. Assuming the attacker has an advantage of ε in breaking the scheme, under the condition of valid ciphertexts, the attacker can correctly guess with a probability of $\frac{1}{2} + \varepsilon$. When $R \neq g^{\gamma_1\gamma_2^2}$, it appears as a collection of some random numbers to the attacker, and the probability of the attacker's accurate guess is $\frac{1}{2}$. Therefore, the challenger also has a probability of $\frac{\varepsilon}{2}$ to solve the difficult BDDH problem. □

5.2. Performance Analysis

5.2.1. Functionality Comparison

In this section, we compare the security and functionality features of our searchable attribute-based encryption scheme with current searchable attribute-based encryption schemes [9,10,13,16]. The comparison specifically includes ciphertext indistinguishability, trapdoor indistinguishability, resistance to keyword guessing attacks, whether it employs ciphertext policy or key policy, and whether it supports aided decryption. The comparison results are presented in Table 3. Table 3 demonstrates that our scheme has significant advantages in terms of security features.

Table 3. Comparison of security properties.

Schemes	Ciphertext Indistinguishability	Trapdoor Indistinguishability	Keyword Guessing Attack	Based on the Strategy	Server-Aided Decryption
DSF [9]	No	No	Yes	CP-ABE	Yes
Literature [13]	No	No	Yes	CP-ABE	No
ABKS-HD [10]	No	No	Yes	CP-ABE	Yes
LABSE [16]	No	No	Yes	KP-ABE	Yes
Ours	Yes	Yes	No	CP-ABE	Yes

Yes: denotes that the specified scheme is secure; No: denotes that the specified scheme is insecure.

5.2.2. Storage Cost

In this subsection, we compare our schemes with the ABKS-HD [10] scheme and the DSF [9] scheme in terms of user key length, ciphertext length, and trapdoor length, as shown in Table 4. There are two reasons why we chose to conduct performance analysis on the ABKS-HD [10] scheme and the DSF [9] scheme. First, they are both based on ciphertext-policy attribute-based encryption (CP-ABE), similar to our proposed SA-CP-SABE scheme. Second, they both have server-assisted decryption capabilities. These factors make them appropriate candidates for comparative performance evaluations. It should be noted that the calculation of ciphertext length for the CP-ABESA scheme does not include the part $C_m = Enc_{ck}(m)$, as this component is consistent across all three schemes. Therefore, when comparing storage and computational costs, this part is not taken into account.

Table 4. Comparison of storage cost.

Schemes	Size of User Key	Size of Ciphertext	Size of Trapdoor
DSF [9]	$(k+4)\|G_1\| + \|Z_q\|$	$(3l+4)\|G_1\| + l\|z_q\| + 2\|G_2\|$	$2\|G_1\|$
ABKS-HD [10]	$2(k+1)\|G_1\|$	$(2l+4)\|G_1\| + l\|G_2\|$	$(2k+3)\|G_1\|$
Ours	$2(k+1)\|G_1\|$	$(2l+1)\|G_1\| + 2\|G_2\|$	$(2k+1)\|G_1\|$

Note: k indicates the number of user's attributes, $|X|$ indicates the length of object X, l Indicates the number of leaf nodes in the access tree or the number of rows of matrix M in the linear access structure (LSSS).

5.2.3. Computation Cost

For the encryption algorithm, the trapdoor generation algorithm, the search algorithm, and the decryption algorithm, we first perform a theoretical estimation of the computation time for the ABKS-HD [10] scheme and the SA-CP-SABE scheme, as shown in Table 5. Note that T_d, T_m, and T_e, respectively, denote the inverse, multiplication, and exponential operations in the group. T_p denotes the pair operation. n_1 represents the size of the smallest subset of attributes in the user attribute set that satisfies the access tree. n_2 denotes the number of internal nodes of the subtree from which the subset forms the access tree. d represents the average threshold value of the internal nodes.

Evidently, as depicted in Table 5, in our scheme, the exponential operations in the encryption algorithm are linearly related to the number of leaf nodes in the access control tree, the multiplication operations in the trapdoor generation algorithm are linearly correlated with the size of user attributes, and the pairing operations in the search and transformation

algorithm are linearly related to the size of the smallest subset of attributes in the user attribute set that satisfies the access tree.

Table 5. Comparison of computation cost.

Stage	ABKS-HD [10]	Ours
Encryption	$3T_m + (2l+7)T_e$	$2T_m + (2l+4)T_e + T_p$
Trapdoor Generation	$T_m + (2k+4)T_e$	$(k+1)T_m + 3T_e$
Search and Transform	$n_1T_d + n_2dT_m + n_2T_e + (2n_1+3)T_p$	$n_1T_d + (2n_2d+1)T_m + 2n_2T_e + (2n_1+1)T_p$
User Decryption	$2T_d + T_e + T_p$	$T_d + 2T_m + T_p$

Subsequently, we compare their computational costs through simulation experiments. The experimental simulation platform is as follows: Intel(R) Core(TM) i3-4130 CPU @3.40GHz processor, 4GB memory, Ubuntu 14.04.3 operating system, and the programming language is Python 3.7. To handle group operations, we utilize the PBC library. Additionally, we conduct testing on the "SS512" super-singular symmetric group. The time cost of each phase (encryption, trapdoor generation, search and transform and user decryption) are illustrated in Figures 2–5, respectively.

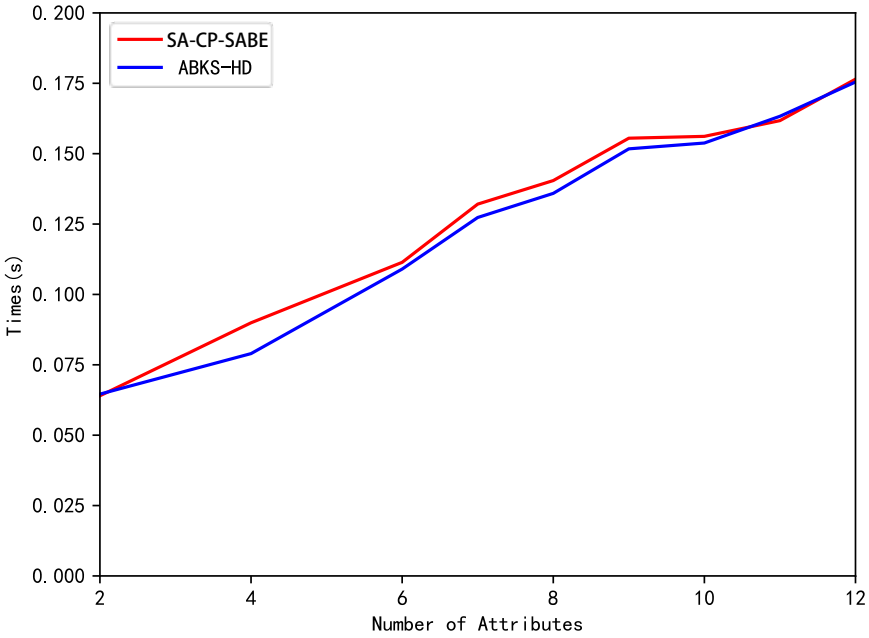

Figure 2. Comparison of encryption time.

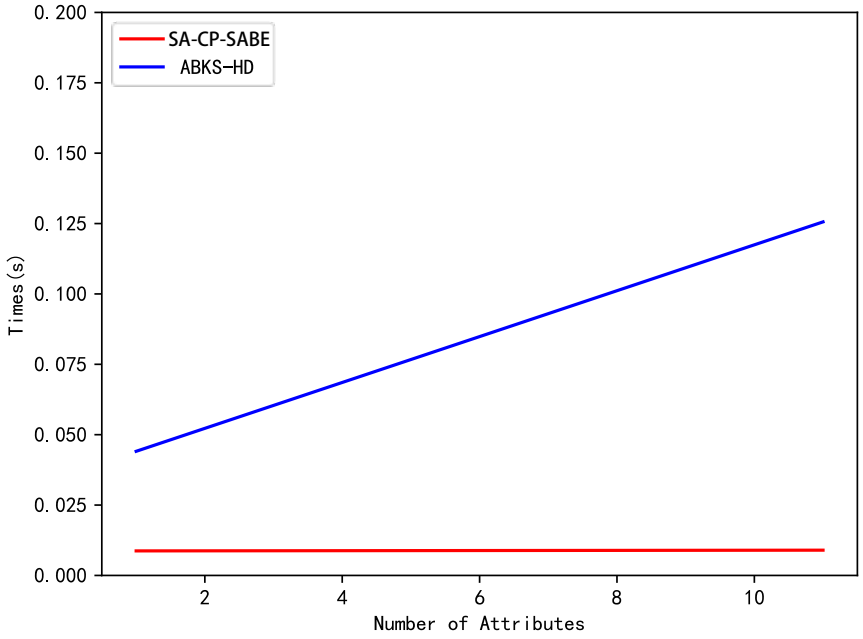

Figure 3. Comparison of trapdoor generation time.

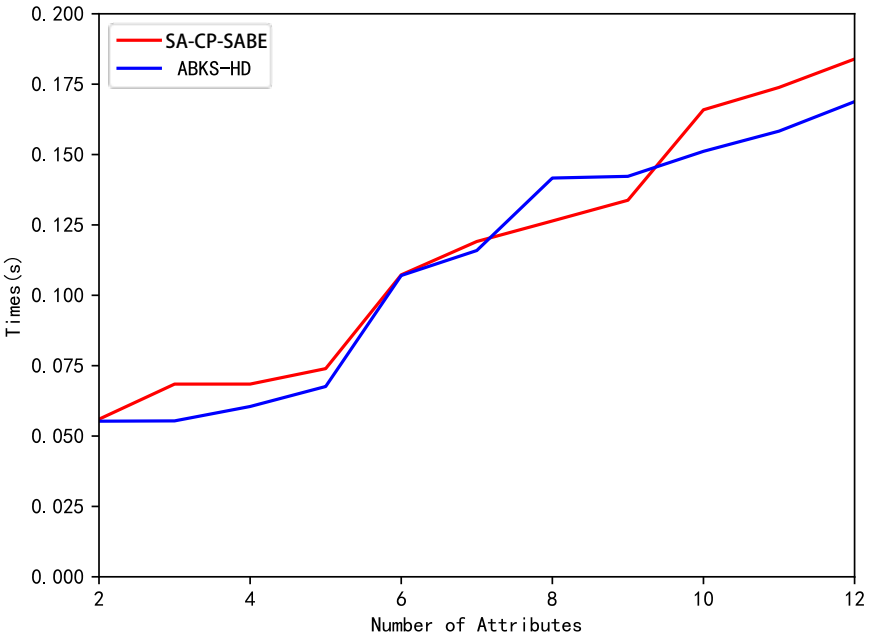

Figure 4. Comparison of search transform times.

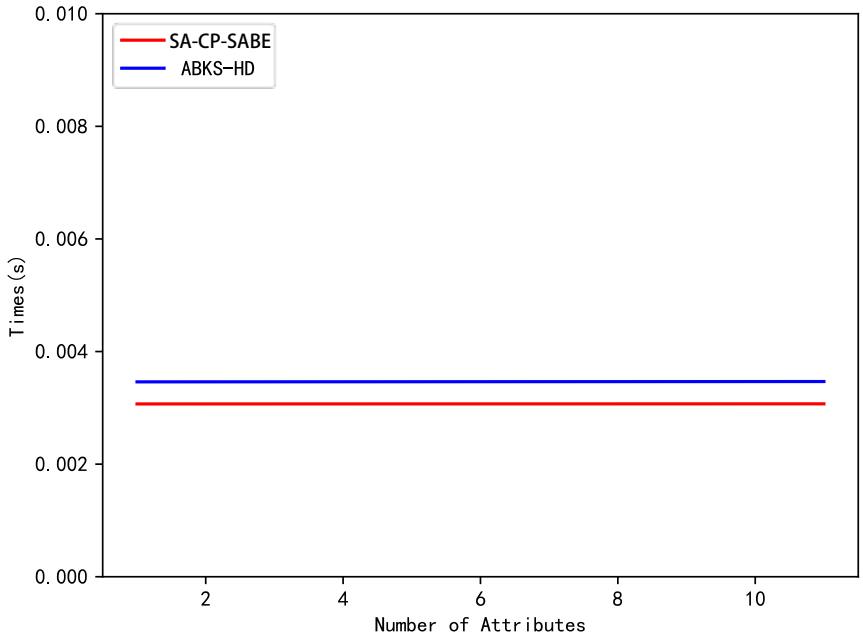

Figure 5. Comparison of decryption time.

5.2.4. Discussion

Figure 2 illustrates the encryption time (in seconds) as it varies with the number of leaf nodes in the access tree. It should be noted that the encryption time is not solely determined by the number of leaf nodes but also relates to the structural form of the access tree. However, the computationally intensive operations (such as multiplication, exponentiation, etc.) mainly occur at the leaf nodes. Therefore, only the number of leaf nodes is calculated.

Figure 3 depicts the execution time (in seconds) for trapdoor generation corresponding to keywords as it varies with the number of attributes. Under the condition of disregarding other non-algorithmic factors, the trapdoor generation time exhibits a linear function of the number of attributes. Therefore, based on the raw data obtained from the simulation results, we utilized the least squares method to model the trend of the generation time with the number of attributes.

Figure 4 illustrates the variation of the search and transform time (in seconds) with the number of attributes. From the graph, it is evident that the search and transform time are linearly related to the number of attributes, which confirms the theoretical analysis discussed in the previous section.

Figure 5 illustrates the variation of decryption time with the number of attributes. Since the ciphertext decryption performed by the user is independent of the number of attributes after the computation by the cloud server in the search and transform phase, Figure 5 also employs the least squares method to model the trend of the decryption time with the number of attributes.

From the analysis above, it can be concluded that the overall performance of the SA-CP-SABE scheme is comparable to that of the ABKS-HD scheme, but it offers higher security.

6. Conclusions

In this paper, we propose a new ciphertext policy-based searchable attribute-based encryption scheme (SA-CP-SABE) to enhance the security for cloud-based smart grids and efficiency which achieves the control of user data access rights and data search rights. SA-CP-SABE has both the unforgeability of the ciphertext and the indistinguishability of the trapdoor, overcoming the security problems of many similar existing schemes. In addition, the performance analysis shows that the proposed SA-CP-SABE scheme also offers superior performance benefits. However, the limitation of the current scheme is that,

to prevent offline keyword guessing attacks, data users need to request authorization from the KGC for each search trapdoor generation, which increases the operational load on the KGC. In our future work, we will focus on developing a one-time authorization system to eliminate the need for repeated permissions with each trapdoor generation and design more functional, more efficient, and more secure searchable encryption schemes.

Author Contributions: Conceptualization, J.W., H.L. and C.L.; methodology, J.W., H.L. and C.L.; software, C.L.; validation, H.L., L.L. and C.L.; security analysis, H.L. and C.L.; resources, H.L.; writing—original draft preparation, J.W. and L.L.; writing—review and editing, C.L. and L.L.; visualization, H.L. and C.L.; supervision, H.L., L.L. and C.L.; funding acquisition, H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported, in part, by "Kunlun Elite" Talent Recruitment Research Project under Grant No. 2023-QLGKLYCZX-028, and New Faculty (Ph.D.) Extended Research and Cultivation Program under Grant No. 202302lwys018.

Data Availability Statement: The authors confirm that the data supporting the findings of this study are available within the article.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

ABE	Attribute-Based Encryption
KP-ABE	Key-Policy Attribute-Based Encryption
CP-ABE	Ciphertext-Policy Attribute-Based Encryption
PEKS	Public Key Searchable Encryption
SABE	Searchable Attribute-Based Encryption
KGC	Key Generation Center
DO	Data Owner
DU	Data User
CSP	Cloud Server Provider

References

1. Mell, P.; Grance, T. The NIST Definition of Cloud Computing. Available online: http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf (accessed on 7 May 2024).
2. Tabrizchi, H.; Rafsanjani, M.K. A survey on security challenges in cloud computing: Issues, threats, and solutions. *J. Supercomput.* **2020**, *76*, 9493–9532. [CrossRef]
3. Sahai, A.; Waters, B. Fuzzy Identity-Based Encryption. In *Advances in Cryptology—EUROCRYPT 2005*; Cramer, R., Ed.; Springer : Berlin/Heidelberg, Germany, 2005; pp. 457–473.
4. Goyal, V.; Pandey, O.; Sahai, A.; Waters, B. Attribute-based encryption for fine-grained access control of encrypted data. In Proceedings of the 13th ACM Conference on Computer and Communications Security, OCT 2006, Alexandria, VA, USA, 30 October–3 November 2006. [CrossRef]
5. Bethencourt, J.; Sahai, A.; Waters, B. Ciphertext-Policy Attribute-Based Encryption. In Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP '07), Oakland, CA, USA, 20–23 May 2007; pp. 321–334. [CrossRef]
6. Wang, S.; Ye, J.; Zhang, Y. A keyword searchable attribute-based encryption scheme with attribute update for cloud storage. *PLoS ONE* **2018**, *13*, e0197318. [CrossRef] [PubMed]
7. Li, H.; Jing, T. A lightweight fine-grained searchable encryption scheme in fog-based healthcare IoT networks. *Wirel. Commun. Mob. Comput.* **2019**, *2019*, 1019767. [CrossRef]
8. Zhang, K.; Long, J.; Wang, X.; Dai, H.N.; Liang, K.; Imran, M. Lightweight Searchable Encryption Protocol for Industrial Internet of Things. *IEEE Trans. Ind. Inform.* **2021**, *17*, 4248–4259. [CrossRef]
9. Miao, Y.; Tong, Q.; Choo, K.K.R.; Liu, X.; Deng, R.H.; Li, H. Secure Online/Offline Data Sharing Framework for Cloud-Assisted Industrial Internet of Things. *IEEE Internet Things J.* **2019**, *6*, 8681–8691. [CrossRef]
10. Miao, Y.; Ma, J.; Liu, X.; Li, X.; Jiang, Q.; Zhang, J. Attribute-Based Keyword Search over Hierarchical Data in Cloud Computing. *IEEE Trans. Serv. Comput.* **2020**, *13*, 985–998. [CrossRef]
11. Chen, D.; Cao, Z.; Dong, X. Online/offline ciphertext-policy attribute-based searchable encryption. *J. Comput. Res. Dev.* **2016**, *53*, 2365–2375. [CrossRef]

12. Niu, S.; Xie, Y.; Yang, P.; Du, X. Cloud-Assisted Attribute-Based Searchable Encryption Scheme on Blockchain. *J. Comput. Res. Dev.* **2021**, *50*, 811–821. [CrossRef]
13. Yin, H.; Zhang, J.; Xiong, Y.; Ou, L.; Li, F.; Liao, S.; Li, K. CP-ABSE: A Ciphertext-Policy Attribute-Based Searchable Encryption Scheme. *IEEE Access* **2019**, *7*, 5682–5694. [CrossRef]
14. Li, J.; Lin, X.; Zhang, Y.; Han, J. KSF-OABE: Outsourced Attribute-Based Encryption with Keyword Search Function for Cloud Storage. *IEEE Trans. Serv. Comput.* **2017**, *10*, 715–725. [CrossRef]
15. Wang, H.; Ning, J.; Huang, X.; Wei, G.; Poh, G.S.; Liu, X. Secure Fine-Grained Encrypted Keyword Search for E-Healthcare Cloud. *IEEE Trans. Dependable Secur. Comput.* **2021**, *18*, 1307–1319. [CrossRef]
16. Bao, Y.; Qiu, W.; Cheng, X. Secure and lightweight fine-grained searchable data sharing for IoT-oriented and cloud-assisted smart healthcare system. *IEEE Internet Things J.* **2022**, *9*, 2513–2526. [CrossRef]
17. Song, D.X.; Wagner, D.; Perrig, A. Practical techniques for searches on encrypted data. In Proceedings of the Proceeding 2000 IEEE Symposium on Security and Privacy, S&P 2000, Berkeley, CA, USA, 14–17 May 2000; pp. 44–55. [CrossRef]
18. Boneh, D.; Di Crescenzo, G.; Ostrovsky, R.; Persiano, G. Public Key Encryption with Keyword Search. In *Advances in Cryptology—EUROCRYPT 2004*; Cachin, C., Camenisch, J.L., Eds.; Springer: Berlin/Heidelberg, Germany, 2004; pp. 506–522.
19. Rhee, H.S.; Park, J.H.; Susilo, W.; Lee, D.H. Trapdoor security in a searchable public-key encryption scheme with a designated tester. *J. Syst. Softw.* **2010**, *83*, 763–771. [CrossRef]
20. Yang, N.; Zhou, Q.; Xu, S. Public-Key Authenticated Encryption with Keyword Search without Pairings. *J. Comput. Res. Dev.* **2020**, *57*, 2125–2135. [CrossRef]
21. Chen, R.; Mu, Y.; Yang, G.; Guo, F.; Huang, X.; Wang, X.; Wang, Y. Server-Aided Public Key Encryption With Keyword Search. *IEEE Trans. Inf. Forensics Secur.* **2016**, *11*, 2833–2842. [CrossRef]
22. Yu, S.; Ren, K.; Lou, W.; Li, J. Defending against Key Abuse Attacks in KP-ABE Enabled Broadcast Systems. In *Security and Privacy in Communication Networks*; Chen, Y., Dimitriou, T.D., Zhou, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 311–329.
23. Wei, J.; Chen, X.; Huang, X.; Hu, X.; Susilo, W. RS-HABE: Revocable-Storage and Hierarchical Attribute-Based Access Scheme for Secure Sharing of e-Health Records in Public Cloud. *IEEE Trans. Dependable Secur. Comput.* **2021**, *18*, 2301–2315. [CrossRef]
24. Liang, K.; Liu, J.K.; Lu, R.; Wong, D.S. Privacy Concerns for Photo Sharing in Online Social Networks. *IEEE Internet Comput.* **2015**, *19*, 58–63. [CrossRef]
25. Li, S.; Xu, M. Attribute-based public encryption with keyword search. *Chin. J. Comput.* **2014**, *37*, 1017–1024.
26. Zhou, R.; Zhang, X.; Du, X.; Wang, X.; Yang, G.; Guizani, M. File-centric multi-key aggregate keyword searchable encryption for industrial internet of things. *IEEE Trans. Ind. Inform.* **2018**, *14*, 3648–3658. [CrossRef]
27. Lai, J.; Deng, R.H.; Guan, C.; Weng, J. Attribute-Based Encryption With Verifiable Outsourced Decryption. *IEEE Trans. Inf. Forensics Secur.* **2013**, *8*, 1343–1354. [CrossRef]
28. Delerablée, C.; Pointcheval, D. Dynamic Threshold Public-Key Encryption. In *Advances in Cryptology—CRYPTO 2008*; Wagner, D., Ed.; Springer:Berlin/Heidelberg, Germany, 2008; pp. 317–334.