ABDKS: Attribute-Based Encryption with Dynamic Keyword Search in Fog Computing

Fei Meng, Mingqiang Wang*, and Leixiao Cheng

School of Mathematics, Shandong University, Jinan Shandong 250100, China
201720214@mail.sdu.edu.cn
wangmingqiang@sdu.edu.cn

Abstract. Attribute-based encryption with keyword search (ABKS) achieves both fine-grained access control and keyword search. However, in the previous ABKS schemes, the search algorithm requires that each keyword between the target keyword set and the ciphertext keyword set be the same, otherwise the algorithm doesn't output any search result, which is not conducive to use. Moreover, the previous ABKS schemes are vulnerable to what we call a *peer-decryption attack*, that is, the ciphertext may be eavesdropped and decrypted by an adversary who has sufficient authorities but no information about the ciphertext keywords.
In this paper, we provide a new system in fog computing, the ciphertext-policy attribute-based encryption with dynamic keyword search (ABDKS). In ABDKS, the search algorithm requires only *one* keyword to be identical between the two keyword sets and outputs the corresponding correlation which reflects the number of the same keywords in those two sets. In addition, our ABDKS is resistant to peer-decryption attack, since the decryption requires not only sufficient authority but also at least one keyword of the ciphertext. Beyond that, the ABDKS shifts most computational overheads from resource constrained users to fog nodes. The security analysis shows that the ABDKS can resist Chosen-Plaintext Attack (CPA) and Chosen-Keyword Attack (CKA).

Keywords: Fog computing · Outsourcing · Access control · Attribute-based encryption · Keyword search.

1 Introduction

Cloud computing is an Internet-based computing method, through which data can be stored, shared and processed. With the development of 5G, IoT, and the emergence of countless intelligent devices, tremendous data needs storage and procession in the cloud, which could gives rise to huge network congestion and latency. Cloud computing is unable to meet the requirements of the contemporary era, so fog computing [3, 21, 22, 27] is proposed. As shown in Fig. 1, fog nodes are closer to the end user and process data at the edge of the network, which infiltrates into factories, automobiles, electrical appliances, street lamps

* Corresponding author

and various articles in people's daily life. Compared with cloud computing, fog computing can reduce request response time, save energy, and reduce network bandwidth, although it's overall computing ability is not as powerful as cloud's. While enjoying the convenience brought by fog computing services, data security is still a critical issue to be considered.

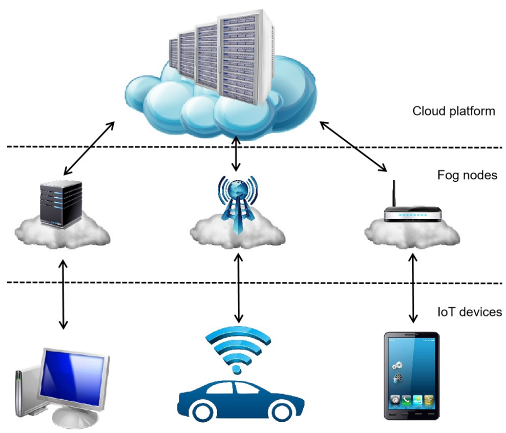

Fig. 1. The instruction of fog computing.

Sensitive information is usually encrypted before being uploaded to the cloud and the encrypted data should be amenable to access control. Attribute-based encryption (ABE) [19] achieves fine-grained access control on encrypted data. For finding a ciphertext containing a specific keyword among all encrypted data, ciphertext-policy attribute-based encryption with keyword search (CP-ABKS) [23, 30] supports both fine-grained access control and keyword search simultaneously, which has a wide range of applications in industrial, academic and medical fields. In CP-ABKS, the user's computational overheads increase with the complexity of the access structure. Outsourcing technology [8] is considered as a promising solution. CP-ABKS schemes in fog computing environment [18, 26] reduce user's computational overheads by outsourcing computing tasks to fog nodes. Therefore, user only needs to perform few operations on resource-limited devices, such as smartphone or ipad.

1.1 Motivation and Contributions

The search algorithms in previous CP-ABKS schemes [13,18,23,26,30] require that each keyword between the target keyword set and the ciphertext keyword set be the same when searching for a ciphertext. As long as the two keyword sets are not completely identical, their algorithms can't output any search result.

Moreover, we find that schemes in [18, 23, 30] are vulnerable to what we call *peer-decryption attack*. In such attack, the ciphertext may be eavesdropped and decrypted by an adversary who has sufficient authorities but noting about the keywords. For example, while downloading a ciphertext, an employee maybe eavesdropped and peeped by colleagues or bosses with the same or higher access authorities. In this case, the ciphertext can be decrypted by his peers or superiors, and his privacy will be revealed.

Motivated by the observations in previous CP-ABKS schemes as above, we first present a new system in fog computing, the ABDKS, which greatly saves the computational costs of resource limited users and achieves dynamic keyword search and peer-decryption resistance. Specifically, the main contributions of our paper are as follows:

- **Dynamic keyword search:** The search algorithm of ABDKS returns the search result as long as *one keyword* is identical between the target keyword set and the ciphertext keyword set. In addition, it outputs the correlation of the two keyword set for result ranking.
- **Peer-decryption resistance:** In ABDKS, anyone who wants to decrypt a ciphertext if and only if he has sufficient authority and knows *at least one element* in the ciphertext keyword set. Thus, the ABDKS can resist peer-decryption attack.
- **Lightweight computational overheads:** With the help of fog nodes, most computational costs of data owner or end users, during data encryption, or trapdoor generation and ciphertext decryption are outsourced to fog nodes without any loss of data confidentiality, and leaving a constant number of operations for the data owner or end users to execute locally.

1.2 Organization

This paper is organized as follows. Section 2 discusses several previous works. Section 3 describes the necessary preliminaries. Section 4 presents the system and security model. We give a concrete construction and explicit analysis of ABDKS in section 5 and section 6 respectively. In the end, section 7 summarizes the paper and prospects for the future research.

2 Related Works

Sahai et al. [19] initially introduced the concept of attribute-based encryption (ABE). Generally, there are two types of ABE schemes, i.e., key-policy ABE (KP-ABE) [7] and ciphertext-policy ABE (CP-ABE) [1]. Bethencourt et al. [1] proposed the first CP-ABE scheme which realizes fine-grained access control based on the tree-based structure. From then on, large numbers of CP-ABE schemes have been proposed to achieve various functions [5,10,25]. However, with the increase of the number of attributes and the complexity of access structure, general CP-ABE schemes are computationally expensive.

Green et al. [8] provided a method to outsource the decryption of ABE ciphertexts. Li et al. [14] outsources both key-issuing and decryption. Zhang et al. [29] fully outsources key generation, encryption and decryption. In the wake of 5G and IoT techniques, fog computing is considered as a new data resource, which can provide many high-quality outsourcing services. Zuo et al. [31] proposed a practical CP-ABE scheme in fog computing environment and Zhang et al. [28] initially supports fog computing as well as attribute update.

Searching over encrypted data, the keyword can not be revealed because it may reflect sensitive information of ciphertext. In 2000, Song et al. [20] initially introduced a searchable encryption (SE) technique. Boneh et al. [2] proposed the first public key encryption with keyword search. After that, various SE schemes, such as single keyword search [24], multi-keyword search [4] and fuzzy keyword search [15] have emerged. To support both fine-grained access control and keyword search simultaneously, plenty of ciphertext-policy attribute-based encryption (CP-ABKS) schemes [6,12,13,17,18,23,26,30] have been proposed. Among them, [18,26] are constructed in fog computing environment.

3 Preliminaries

In this section, we introduce some background knowledge, which includes access structure, access tree, bilinear maps, Diffie-Hellman assumption and its variants.

3.1 Access Structures

Definition 1 (Access structure [1]). *Let* $\{P_1, P_2, \ldots, P_n\}$ *be a set of parties. A collection* $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}}$ *is monotone if* $\forall B, C$*: if* $B \in \mathbb{A}$ *and* $B \subseteq C$ *then* $C \in \mathbb{A}$*. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection)* $\mathbb{A}$ *of non-empty subsets of* $\{P_1, P_2, \ldots, P_n\}$*, i.e.,* $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}} \setminus \{\emptyset\}$*. The sets in* $\mathbb{A}$ *are called the authorized sets, and the sets not in* $\mathbb{A}$ *are called the unauthorized sets.*

In this paper, attributes take the role of the parties and we only focus on the monotone access structure $\mathbb{A}$, which consists of the authorized sets of attributes. Obviously, attributes can directly reflect a user's authority.

Definition 2 (Access tree [1]). *Let* $\mathcal{T}$ *be a tree representing an access structure. Each non-leaf node of the tree represents a threshold gate, described by its children and a threshold value. If* num_x *is the number of children of a node* x *and* k_x *is its threshold value, then* $0 \le k_x \le num_x$*. When* $k_x = 1$*, the threshold gate is an OR gate and when* $k_x = num_x$*, it is an AND gate. Each leaf node* x *of the tree is describe by an attribute and a threshold value* $k_x = 1$*.*

We introduce a few functions defined in [1] as follows. $parent(x)$ denotes the parent of the node x in the tree. The function $att(x)$ is defined only if x is a leaf node and denotes the attribute associated with the leaf node x in the tree. The access tree $\mathcal{T}$ also defines an ordering between the children of every node, that is, the children of a node are numbered from 1 to num. The function $index(x)$ returns such a number associated with the node x, where the index values are uniquely assigned to nodes in the access structure for a given key in an arbitrary manner.

Definition 3 (Satisfying an access tree [1]). *Let $\mathcal{T}$ be an access tree with root r. Denote by $\mathcal{T}_x$ the subtree of $\mathcal{T}$ rooted at the node x. Hence $\mathcal{T}$ is the same as $\mathcal{T}_r$. If a set of attributes γ satisfies the access tree $\mathcal{T}_x$, we denote it as $\mathcal{T}_x(\gamma) = 1$. We compute $\mathcal{T}_x(\gamma)$ recursively as follows. If x is a non-leaf node, evaluate $\mathcal{T}_{x'}(\gamma) = 1$ for all children x' of node x. $\mathcal{T}_x(\gamma)$ returns 1 if and only if at least k_x children return 1. If x is a leaf node, then $\mathcal{T}_x(\gamma)$ returns 1 if and only if $att(x) \in \gamma$.*

3.2 Bilinear Map and DBDH Assumption

We briefly recall the definitions of the bilinear map and the decisional bilinear Diffie-Hellman (DBDH) assumption. Let $\mathbb{G}_0$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}_0$ and e be a efficient computable bilinear map, $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_T$. The bilinear map e has a few properties: (1) Bilinearity: for all $u, v \in \mathbb{G}_0$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$. (2) Non-degeneracy: $e(g, g) \neq 1$.

We say that $\mathbb{G}_0$ is a bilinear group if the group operation in $\mathbb{G}_0$ and the bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_T$ are both efficiently computable. Notice that the map e is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

Given the bilinear map parameter $(\mathbb{G}_0, \mathbb{G}_T, p, e, g)$ and three random elements $(x, y, z) \in \mathbb{Z}_p^3$, if there is no probabilistic polynomial time (PPT) adversary $\mathcal{B}$ can distinguish between the tuple $(g, g^x, g^y, g^z, e(g, g)^{xyz})$ and the tuple $(g, g^x, g^y, g^z, \vartheta)$, we say that the DBDH assumption holds, where ϑ is randomly selected from $\mathbb{G}_T$. More specifically, the advantage ϵ of $\mathcal{B}$ in solving the DBDH problem is defined as

$$\Big| \Pr[\mathcal{A}(g, g^x, g^y, g^z, e(g, g)^{xyz}) = 1] - \Pr[\mathcal{A}(g, g^x, g^y, g^z, \vartheta) = 1] \Big|. \tag{1}$$

Definition 4 (DBDH). *We say that the DBDH assumption holds if no PPT algorithm has a non-negligible advantage ϵ in solving DBDH problem.*

4 System and Security Model

In this section, we introduce the system description, system model, threat model and security model of ABDKS.

4.1 System Description

As shown in Fig. 2, we consider a ciphertext retrieval scenario in fog computing. It consists of five parties: Key Authority Center (KAC), Data Owner (DO), Cloud Server (CS), End User (EU), and Fog Nodes. The specific role of each party is given as follows:

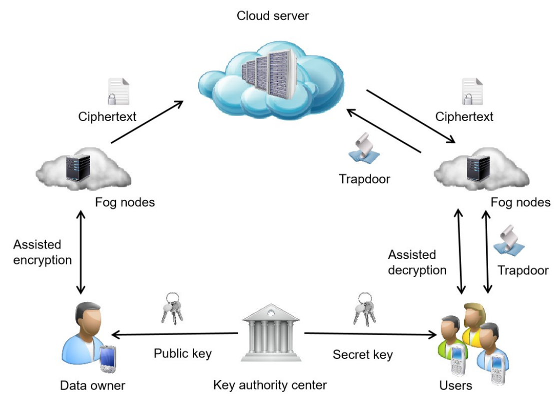

Fig. 2. System description of fog computing.

- Key Authority Center (KAC): The KAC is a fully trusted third party which is in charge of generating public parameters, secret keys, and handling attribute update.
- Data Owner (DO): The DO defines an access structure, chooses a set of keywords to generate a ciphertext CT with the help of fog nodes, then uploads CT to the CS.
- Cloud Server (CS): The CS has unlimited computing power and storage capacity, it can provide computing and storage services to users.
- End User (EU): Resource-constrained user can generate trapdoor with the help of fog nodes and issue search queries based on their authority. Moreover, it can take the advantage of fog nodes to decrypt ciphertext.
- Fog Nodes: The fog node can help reduce computational overheads during the encryption process of DO or the trapdoor generation and decryption processes of EU.

4.2 System Model

The ABDKS includes the following six algorithms:

- **Setup**$(1^\lambda, \mathcal{L}) \rightarrow (PK, MSK)$: Given security parameter λ and a set of all possible attributes $\mathcal{L}$, the KAC generates public key PK and master secret key MSK.
- **KeyGen**$(PK, MSK, S) \rightarrow SK$: On input the public key PK, the master secret key MSK and an attribute set S, the KAC generates a secret key SK for the EU.
- **Enc**$(PK, \mathbb{A}, M, KW) \rightarrow CT$: On input an access structure $\mathbb{A}$, a keyword set KW and the message M, the DO generates the ciphertext CT with the help of fog nodes, and uploads the ciphertext to the CS.
- **Trap**$(SK, KW') \rightarrow (Ta, Tk)$: To issue a search query, the EU generates a trapdoor (Ta, Tk) by his own secret key SK and a set of target keyword KW' with the help of fog nodes.

- **Search**$(\{CT\}, (Ta, Tk)) \rightarrow \{(CT_{(i)}, \overrightarrow{mr}_{(i)}, l_{(i)})\}$ or $\perp$: On input a trapdoor (Ta, Tk), the CS conducts searching operations among all ciphertexts $\{CT\}$. For each ciphertext $CT \in \{CT\}$, if the following two conditions satisfied, we call CT an accessible ciphertext:
  - there is at least one keyword kw such that $kw \in KW' \bigcap KW$,
  - the user is authorized to obtain CT,

  where KW' is the target keyword set of (Ta, Tk), KW is the ciphertext keyword set of CT. The algorithm finally returns the accessible ciphertext set $\{(CT_{(i)}, \overrightarrow{mr}_{(i)}, l_{(i)})\}$, where $CT_{(i)} \in \{CT\}$ is the i-th accessible ciphertext; matching result vector $\overrightarrow{mr}_{(i)}$ implies the relationship between KW' of (Ta, Tk) and $KW_{(i)}$ of the accessible ciphertext $CT_{(i)}$; correlation $l_{(i)}$ is the number of non-zero integers in $\overrightarrow{mr}_{(i)}$, indicating the correlation of KW' and $KW_{(i)}$. If no accessible ciphertext exists, it outputs $\perp$.
- **Dec**$(CT_{(i)}, \overrightarrow{mr}_{(i)}, SK, KW') \rightarrow M$: On input $CT_{(i)}, \overrightarrow{mr}_{(i)}$, SK, KW', the EU can decrypt the accessible ciphertext $CT_{(i)}$ with the help of fog nodes and outputs M.

4.3 Threat Model

In this paper, we assume that the KAC is a fully trusted third party, while the CS and fog nodes are honest-but-curious entities, which exactly follow the protocol specifications but also are curious about the sensitive information of ciphertexts and trapdoors. Users are not allowed to collude with CS or fog nodes. Nevertheless, malicious users may collude with each other to access some unauthorized ciphertexts. During transmitted, the ciphertext may be eavesdropped and decrypted by peer-decryption adversary who has sufficient authorities but noting about the keywords.

4.4 Security Model

The ABDKS achieves chosen plaintext security, and the security game between a PPT adversary $\mathcal{A}$ and the challenger $\mathcal{C}$ is as follows.

- *Initialization*: $\mathcal{A}$ chooses and submits a challenge access structure $\mathbb{A}^*$ to its challenger $\mathcal{C}$.
- *Setup*: $\mathcal{C}$ runs **Setup** algorithm and returns the public key PK to $\mathcal{A}$.
- *Phase* 1: $\mathcal{A}$ adaptively submits any attribute set S to $\mathcal{C}$ with the restriction that S doesn't satisfy $\mathbb{A}^*$. In response, $\mathcal{C}$ runs **KeyGen** algorithm and answers $\mathcal{A}$ with the corresponding SK.
- *Challenge*: $\mathcal{A}$ chooses two equal-length challenge messages (m_0, m_1), a set of keywords KW^* and submits them to $\mathcal{C}$. Then $\mathcal{C}$ picks a random bit $\vartheta \in \{0, 1\}$, runs **Enc** algorithm to encrypt (m_ϑ, KW^*), and returns the challenge ciphertext CT^* to $\mathcal{A}$.
- *Phase* 2: This phase is the same as Phase 1.
- *Guess*: $\mathcal{A}$ outputs a guess bit ϑ' of ϑ. We say that $\mathcal{A}$ wins the game if and only if $\vartheta' = \vartheta$. The advantage of $\mathcal{A}$ to win this security game is defined as $Adv(\mathcal{A}) = \left| \Pr[\vartheta' = \vartheta] - \frac{1}{2} \right|$.

Definition 5. *The ABDKS achieves IND-CPA security if there exist no PPT adversary winning the above security game with a non-negligible advantage ϵ under the DBDH assumption.*

In addition, the ABDKS also achieves chosen keyword security, and the security game between $\mathcal{A}$ and $\mathcal{C}$ is as follows.

- *Initialization*: $\mathcal{A}$ chooses and submits a challenge access structure $\mathbb{A}^*$ to its challenger $\mathcal{C}$.
- *Setup*: $\mathcal{C}$ runs **Setup** algorithm and gives PK to $\mathcal{A}$.
- *Phase* 1: $\mathcal{A}$ adaptively submits any attribute set S and keyword set KW to $\mathcal{C}$ with the restriction that S doesn't satisfy $\mathbb{A}^*$. In response, $\mathcal{C}$ runs **Trap** algorithm and responds $\mathcal{A}$ with the corresponding trapdoor (Ta, Tk).
- *Challenge*: $\mathcal{A}$ submits two challenge keyword sets KW^{0*} and KW^{1*} with equal number of keywords. Then, $\mathcal{C}$ picks a random bit $\vartheta \in \{0,1\}$, and returns the challenge ciphertext CT^*.
- *Phase* 2: This phase is the same as Phase 1.
- *Guess*: $\mathcal{A}$ outputs a guess bit ϑ' of ϑ. We say that $\mathcal{A}$ wins the game if and only if $\vartheta' = \vartheta$. The advantage of $\mathcal{A}$ to win this security game is defined as $Adv(\mathcal{A}) = \left| \Pr[\vartheta' = \vartheta] - \frac{1}{2} \right|$.

Definition 6. *The ABDKS achieves IND-CKA security if there exist no PPT adversary winning the above security game with a non-negligible advantage ϵ under the DBDH assumption.*

5 Construction of ABDKS

Here, we present the concrete construction of ABDKS scheme with dynamic keyword search and peer-decryption resistance. Without loss of generality, we suppose that there are n possible attributes in total and $\mathcal{L} = \{a_1, a_2, ..., a_n\}$ is the set of all possible attributes. Assume $\mathbb{G}_0, \mathbb{G}_T$ are multiplicative cyclic groups with prime order p and the generator of $\mathbb{G}_0$ is g. Let λ be the security parameter which determines the size of groups. Let $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_T$ be a bilinear map and $H : \{0,1\}^* \longrightarrow \mathbb{Z}_p$ be a hash function which maps any string to a random element of $\mathbb{Z}_p$. We also define the Lagrange coefficient $\Delta_{i,L}(x) = \prod_{j \in L, j \neq i} \frac{x-j}{i-j}$, where $i \in \mathbb{Z}_p$ and a set L, of elements in $\mathbb{Z}_p$. The details of our scheme are as follows.

- **Setup**$(1^\lambda, \mathcal{L}) \to (PK, MSK)$: Given a security parameter λ and all possible attributes $\mathcal{L}$, the KAC chooses a bilinear group $\mathbb{G}_0$ with prime order p and generator g. Next, it randomly picks $\alpha, \beta \in \mathbb{Z}_p$ and $h \in \mathbb{G}_0$. For each attribute $a_j \in \mathcal{L}$, it randomly selects a $v_j \in \mathbb{Z}_p$ and sets $PK_j = g^{v_j}$. Finally, it generates the the public key PK and master secret key MSK as

$$PK = \begin{Bmatrix} \mathbb{G}_0, g, h, g^\alpha, e(g,g)^\beta, e(g,h)^\beta, \\ \{PK_j = g^{v_j} \mid \forall a_j \in \mathcal{L}\} \end{Bmatrix}; \tag{2}$$

$$MSK = \{\alpha, \beta, \{v_j \mid \forall a_j \in \mathcal{L}\}\}. \tag{3}$$

- **KeyGen**$(MSK, S) \to SK$: While receiving an attribute set S from the EU, the KAC randomly selects $r, r' \in \mathbb{Z}_p$ and returns the secret key SK as

$$SK = \left\{ g^{\beta+\alpha r}, g^{\alpha r} h^{r'}, h^{\alpha r} h^{r'}, g^{r'}, \{ g^{\frac{\alpha r}{v_j}}, h^{\frac{\alpha r}{v_j}} \mid \forall a_j \in S \} \right\}. \quad (4)$$

- **Enc**$(PK, \mathbb{A}, M, KW) \to CT$: The DO randomly chooses $ck \in \mathbb{Z}_p$ as a symmetric encryption key and encrypts message M with ck, $E_{ck}(M)$, by using symmetric encryption (AES). Then, it encrypts ck as follows:
  1. The DO sends an access structure $\mathbb{A}$ to fog nodes, which in turn represent $\mathbb{A}$ with an access tree $\mathcal{T}$. Then the fog nodes randomly choose a polynomial q_x for each node x of $\mathcal{T}$ from the root node R in a top-down manner: for each node x of $\mathcal{T}$, the degree of q_x is $d_x = k_x - 1$, where k_x is the threshold value of x; beginning with root node R, fog nodes pick a random $s_1 \in \mathbb{Z}_p$, set $q_R(0) = s_1$ and randomly choose $d_R = k_R - 1$ other points of q_R to define the polynomial completely; for any other node x, fog nodes set $q_x(0) = q_{parent(x)}(index(x))$ and choose d_x other points to define q_x completely. Fog nodes send the **attribute ciphertext** CT'_1,

$$CT'_1 = \left\{ \begin{array}{l} \mathcal{T}, g^{s_1}, h^{s_1}, \{C_{j,1} = g^{v_j q_x(0)}, \\ C_{j,2} = h^{v_j q_x(0)} \mid \forall a_j = att(x) \in \mathcal{X}\} \end{array} \right\}, \quad (5)$$

  to DO, where $\mathcal{X}$ is a set of attributes corresponding with all leaf nodes in $\mathcal{T}$. Given CT'_1, the DO randomly picks $s_2 \in \mathbb{Z}_p$ and generates CT_1 as

$$CT_1 = \{g^{s_2}, g^{s_1} g^{s_2}, h^{s_1}, h^{s_2}, CT'_1\}. \quad (6)$$

  2. The DO picks the ciphertext keyword set $KW = \{kw_1, kw_2, \cdots, kw_t\}$, where kw_i means the i-th keyword. Then, it randomly selects $r_1, r_2 \in \mathbb{Z}_p$, computes $g^{\frac{1}{r_1}}$, and sends $h^{s_2 r_1}, KW_1, KW_2$ to fog nodes, where $KW_1 = \{H(kw_1)r_1, \ldots, H(kw_t)r_1\}$ and $KW_2 = \{H(kw_1)r_2, \ldots, H(kw_t)r_2\}$.
  The fog nodes return DO the **keyword ciphertext**

$$CT'_2 = \{C_1^i = h^{\frac{s_2 r_1}{H(kw_i) r_1}}, C_2^i = g^{\frac{1}{H(kw_i) r_2}} \mid \forall i \in [1, t]\}. \quad (7)$$

  3. The DO generates $CT_2 = \{e(g,h)^{\beta s_2}, g^{\frac{1}{r_1}}, CT'_2\}$ and outputs the final ciphertext CT as

$$CT = \left\{ \begin{array}{l} \mathcal{T}, E_{ck}(M), C = ck \cdot e(g,g)^{\beta s_2} \cdot e(g,g)^{\frac{1}{r_2}}, \\ CT_1, CT_2 \end{array} \right\}. \quad (8)$$

- **Trap**$(SK, KW') \to (Ta, Tk)$: To generate a search query on the target keyword set KW', supposed that $KW' = \{kw'_1, \ldots, kw'_t\}$, the **Trap** algorithm proceeds as follows.

1. The EU randomly chooses $x', y', z', r_3 \in \mathbb{Z}_p$, and uses SK to generate the **attribute trapdoor** Ta as

$$\begin{aligned} Ta &= \big(Ta_0, Ta_1, Ta_2, Ta_3, Ta_4\big) \\ &= \Big(x'+y', h^{(\alpha r+r')(x'+y')+z'}, g^{r'(x'+y')+z'}, \\ &\{Ta_3^j = g^{\frac{\alpha r x'}{v_j}} \mid \forall a_j \in S\}, \{Ta_4^j = h^{\frac{\alpha r y'}{v_j}} \mid \forall a_j \in S\}\Big). \end{aligned} \tag{9}$$

2. The EU computes g^{r_3} and sends $\{H(kw'_1)r_3, \ldots, H(kw'_t)r_3\}$ to fog nodes, which return $\{g^{H(kw'_j)r_3} \mid \forall j \in [1,t]\}$ to the former. Finally, the EU generates the **keyword trapdoor** Tk as

$$\begin{aligned} Tk &= (Tk_0, Tk_1) \\ &= (g^{(\beta+\alpha r)+r_3}, \{Tk_1^j = g^{H(kw'_j)r_3} \mid \forall j \in [1,t]\}). \end{aligned} \tag{10}$$

- **Search**$(\{CT\}, (Ta, Tk)) \rightarrow \{(CT_{(i)}, \overrightarrow{mr}_{(i)}, l_{(i)})\}$ or $\perp$: For each $CT \in \{CT\}$, the algorithm conducts the following two steps: access precomputation and keyword matching.
  1. **Access Precomputation**: Due to the access precomputation process is a recursive procedure, we first define a recursive algorithm $F'_x = F'_x(C_{j,1}, C_{j,2}, Ta_3^j, Ta_4^j, x)$ intaking $C_{j,1}$,$C_{j,2}$,x in CT'_1 of CT and Ta_3^j, Ta_4^j in (Ta, Tk).
  For each node x of $\mathcal{T}$ in CT, the CS runs a recursive algorithm as follows:
  (a) If x is a leaf node of $\mathcal{T}$. Let $a_j = att(x)$. If $a_j \in S$, the CS computes

$$\begin{aligned} F'_x &= e(C_{j,2}, Ta_3^j) \cdot e(C_{j,1}, Ta_4^j) \\ &= e(h^{v_j q_x(0)}, g^{\frac{\alpha r x'}{v_j}}) \cdot e(g^{v_j q_x(0)}, h^{\frac{\alpha r y'}{v_j}}) \\ &= e(g,h)^{\alpha r q_x(0)(x'+y')}. \end{aligned} \tag{11}$$

  If $a_j \notin S$, set $F'_x = null$.
  (b) If x is a non-leaf node, the recursive algorithm is defined as: for all child nodes z of x, where $a_i = att(z)$, the CS computes $F'_z = F'_z(C_{i,1}, C_{i,2}, Ta_3^i, Ta_4^i, z)$ recursively. Let S_x be an arbitrary k_x-sized set of child nodes z satisfying $F'_z \neq null$. If S_x doesn't exist, $F'_x = null$. Otherwise, the CS calculates

$$\begin{aligned} F'_x &= \prod_{z \in S_x} F_z'^{\Delta_{i,S'_x}(0)} \\ &= \prod_{z \in S_x} (e(g,h)^{\alpha r(x'+y')q_{parent(z)}(index(z))})^{\Delta_{i,S'_x}(0)} \\ &= e(g,h)^{\alpha r q_x(0)(x'+y')}, \end{aligned} \tag{12}$$

  where $i = index(z)$ and $S'_x = \{index(z) : z \in S_x\}$.

By calling the above algorithm on the root node R of $\mathcal{T}$, the CS gets $F'_R = e(g,h)^{\alpha r s_1 (x'+y')}$. Then, the CS computes D' as

$$\begin{aligned} D' &= e(g,h)^{\beta s_2} \cdot \Big[\frac{e(Ta_1, g^{s_1+s_2})}{F'_R \cdot e(Ta_2, h^{s_1+s_2})}\Big]^{\frac{1}{Ta_0}} = e(g,h)^{\beta s_2} \cdot \\ &\Big[\frac{e(h^{(\alpha r + r')(x'+y')+z'}, g^{s_1+s_2})}{e(g,h)^{\alpha r s_1 (x'+y')} \cdot e(g^{r'(x'+y')+z'}, h^{s_1+s_2})}\Big]^{\frac{1}{x'+y'}} \\ &= e(g,h)^{(\beta+\alpha r)s_2}. \end{aligned} \tag{13}$$

2. **Keyword Matching**: Given TK and CT, the CS interacts $Tk_1 = \{Tk_1^j = g^{H(kw'_j)r_3} \mid \forall j \in [1,t]\}$ of TK with $\{C_1^i = h^{\frac{s_2 r_1}{H(kw_i) r_1}} \mid \forall i \in [1,t]\}$ of CT: for each $j \in [1,t]$, the CS checks whether there exists an $i \in [1,t]$ such that

$$D' \cdot e(Tk_1^j, C_1^i) = e(Tk_0, h^{s_2}). \tag{14}$$

   (a) If no i makes the above formula hold, the CS outputs a matching result $mr_j = 0$, which means that kw'_j in KW' of Tk is not in KW of CT.
   (b) If there is an i such that the above formula holds, which means that there exists a kw_i in KW such that $kw'_j = kw_i$, the CS sets $mr_j = i$.

   Then CS sets the **matching result vector** $\overrightarrow{mr} = (mr_1, mr_2, \ldots, mr_t)$, the **correlation** $l \in [0,t]$ be the number of non-zero elements in $\overrightarrow{mr}$, which represents the number of identical keywords between KW' and KW. If $l > 0$, the CS outputs the accessible ciphertext tuple $(CT, \overrightarrow{mr}, l)$; otherwise, it turns to the next ciphertext.

If there is no ciphertext satisfying the above conditions, the algorithm outputs $\perp$. Else, the CS ranks the accessible ciphertext tuples as $\{CT_{(i)}, \overrightarrow{mr}_{(i)}, l_{(i)}\}$ based on the correlation, as shown in Table 1, which is allowed to be obtained by the EU.

Table 1. Accessible ciphertext for the EU generated by the CS.

Ciphertext	Matching result vector	Correlation
$CT_{(1)}$	$\overrightarrow{mr}_{(1)}$	$l_{(1)}$
$CT_{(2)}$	$\overrightarrow{mr}_{(2)}$	$l_{(2)}$
$\vdots$	$\vdots$	$\vdots$

- **Dec**$((CT, \overrightarrow{mr}), SK, KW') \rightarrow M$: The EU accesses $(CT, \overrightarrow{mr})$ in Table 1 according to l, and decrypt CT with the help of fog nodes in the following steps:

1. The EU selects a random $d \in \mathbb{Z}_p$, keeps it secret, and sends a randomized secret key SK' to fog nodes, where

$$\begin{aligned} SK' &= \Big\{SK'_1, SK'_2, SK'_3, SK'_4 = \{SK'^j_4\}\Big\} \\ &= \Big\{g^{\frac{\beta+\alpha r}{d}}, g^{\frac{\alpha r}{d}} h^{\frac{r'}{d}}, g^{\frac{r'}{d}}, \{g^{\frac{\alpha r}{v_j d}} \mid \forall a_j \in S\}\Big\}. \end{aligned} \tag{15}$$

2. The fog nodes interact SK' and CT to perform some precomputation, which greatly reduces the computational costs of user decryption. The interaction procedure is similar to the access precomputation of **Search** algorithm described above. Due to the precomputation process is a recursive procedure, we define the recursive algorithm $F_x = F_x(C_{j,1}, SK_4'^j, x)$ intaking $C_{j,1}$,x in CT_1' of CT and $SK_4'^j$ in SK'. For each node x of $\mathcal{T}$ in CT, the fog nodes perform the following recursive algorithm:
   (a) If x is a leaf node of $\mathcal{T}$. Let $a_j = att(x)$. If $a_j \notin S$, we set $F_x = null$. If $a_j \in S$, then fog nodes compute
   $$F_x = e(g^{v_j q_x(0)}, g^{\frac{\alpha r}{v_j d}}) = e(g,g)^{\frac{\alpha r q_x(0)}{d}}. \tag{16}$$
   (b) If x is a non-leaf node, the recursive algorithm is defined as: for all child nodes z of x, where $a_i = att(z)$, the fog nodes calculate $F_z = F_z(C_{i,1}, SK_4'^i, z)$ recursively. Let S_x be an arbitrary k_x-sized set of z satisfying $F_z \neq null$. If S_x doesn't exist, $F_x = null$. Otherwise, the fog nodes compute
   $$F_x = \prod_{z \in S_x} F_z^{\Delta_{i,S_x'}(0)} = e(g,g)^{\frac{\alpha r q_x(0)}{d}}. \tag{17}$$

   By running the above algorithm recursively, the fog nodes obtain $F_R = e(g,g)^{\frac{\alpha r s_1}{d}}$ for the root node R of $\mathcal{T}$ and return D to the EU, where
   $$\begin{aligned} D &= \frac{e(SK_1', g^{s_2}) \cdot F_R \cdot e(SK_3', h^{s_1} h^{s_2})}{e(SK_2', g^{s_1} g^{s_2})} \\ &= \frac{e(g^{\frac{\beta+\alpha r}{d}}, g^{s_2}) \cdot e(g,g)^{\frac{\alpha r s_1}{d}} \cdot e(g^{\frac{r'}{d}}, h^{s_1} h^{s_2})}{e(g^{\frac{\alpha r}{d}} h^{\frac{r'}{d}}, g^{s_1} g^{s_2})} \\ &= e(g,g)^{\frac{\beta s_2}{d}}. \end{aligned} \tag{18}$$
3. For the $\overrightarrow{mr}$, there is at least one element mr_j in $\overrightarrow{mr}$ such that $mr_j = i \neq 0$, i.e., the j^{th} keyword kw_j' in KW' is identical to the i^{th} keyword kw_i in KW. Then, the EU derives ck as
   $$\begin{aligned} &\frac{C}{D^d \cdot e(g^{H(kw_j')}, C_2^i)} \\ =& \frac{ck \cdot e(g,g)^{\beta s_2} \cdot e(g,g)^{\frac{1}{r_2}}}{(e(g,g)^{\frac{\beta s_2}{d}})^d \cdot e(g^{H(kw_j')}, g^{\frac{1}{H(kw_i) r_2}})} = ck. \end{aligned} \tag{19}$$
4. Finally, the EU decrypts $E_{ck}(M)$ with ck by symmetric decryption.

Remark 1. Attribute update is very important to keep the system dynamic and protect data from eavesdropping and sniffing by revoked users. we briefly describe how to apply the basic idea of attribute update in [28] to the ABDKS as follows:

1. To update $a_j \to a_w$, the KAC randomly selects $v'_j \neq v_j \in \mathbb{Z}_p$ and computes $uk_{j\to w} = \frac{v_j}{v_w}$, $uk_{j\to j} = \frac{v_j}{v'_j}$, $cu_{j\to j} = \frac{v'_j}{v_j}$. The KAC updates the public attribute key of a_j as $PK'_j = PK_j^{\frac{1}{uk_{j\to j}}} = g^{v'_j}$ and sends $uk_{j\to w}, uk_{j\to j}, cu_{j\to j}$ to the updated user, non-updated users, the CS respectively.
2. The updated user updates its secret key as

$$SK_u = \left\{ \begin{matrix} g^{\beta+\alpha r}, g^{\alpha r}h^{r'}, h^{\alpha r}h^{r'}, g^{r'}, \\ \{g^{\frac{\alpha r}{v_i}}, h^{\frac{\alpha r}{v_i}} \mid \forall a_i \in S \backslash \{a_j\}\}, g^{\frac{\alpha r}{v_j}\cdot uk_{j\to w}}, h^{\frac{\alpha r}{v_w}\cdot uk_{j\to w}} \end{matrix} \right\}. \quad (20)$$

Each non-updated user updates its secret keys as

$$SK_{nu} = \left\{ \begin{matrix} g^{\beta+\alpha r}, g^{\alpha r}h^{r'}, h^{\alpha r}h^{r'}, g^{r'}, \\ \{g^{\frac{\alpha r}{v_i}}, h^{\frac{\alpha r}{v_i}} \mid \forall a_i \in S \backslash \{a_j\}\}, g^{\frac{\alpha r}{v_j}\cdot uk_{j\to j}}, h^{\frac{\alpha r}{v_j}\cdot uk_{j\to j}} \end{matrix} \right\}. \quad (21)$$

The CS updates the attribute ciphertext CT'_1 of CT as

$$CT'_1 = \left\{ \begin{matrix} \mathcal{T}, g^{s_1}, h^{s_1}, C_{j,1} = g^{v_j q_x(0)\cdot cu_{j\to j}}, \\ C_{j,2} = h^{v_j q_x(0)\cdot cu_{j\to j}}, \Big\{C_{i,1} = g^{v_i q_x(0)}, \\ C_{i,2} = h^{v_i q_x(0)}, \mid \forall a_i = att(x) \in \mathcal{X} \setminus \{a_j\}\Big\} \end{matrix} \right\}. \quad (22)$$

6 Analysis of ABDKS

In this section, we provide a security analysis of ABDKS and demonstrate its performance from in a theoretical point of view.

6.1 Security Analysis

Theorem 1. *Supposed that a PPT adversary $\mathcal{A}$ can break the IND-CPA security of ABDKS with a non-negligible advantage $\epsilon > 0$, then there exists a PPT simulator $\mathcal{B}$ that can distinguish a DBDH tuple from a random tuple with an advantage $\frac{\epsilon}{2}$.*

Proof. Given the bilinear map parameter $(\mathbb{G}_0, \mathbb{G}_T, p, e, g)$. The DBDH challenger $\mathcal{C}$ selects $a', b', c' \in \mathbb{Z}_p$, $\theta \in \{0,1\}$, $\mathcal{R} \in \mathbb{G}_T$ at random. Let $\mathcal{Z} = e(g,g)^{a'b'c'}$, if $\theta = 0$, $\mathcal{R}$ else. Next, $\mathcal{C}$ sends $\mathcal{B}$ the tuple $\langle g, g^{a'}, g^{b'}, g^{c'}, \mathcal{Z}\rangle$. Then, $\mathcal{B}$ plays the role of challenger in the following security game.

- *Initialization*:$\mathcal{A}$ submits a challenge access structure $\mathbb{A}^*$ to $\mathcal{B}$.
- *Setup*: $\mathcal{B}$ chooses $\beta', x \in \mathbb{Z}_p$ at random and sets $h = g^x$, $g^\alpha = g^{a'}$, $e(g,g)^\beta = e(g,g)^{\beta'+a'b'} = e(g,g)^{\beta'}e(g^{a'}, g^{b'})$, $e(g,h)^\beta = (e(g,g)^\beta)^x$. For each attribute $a_j \in \mathcal{L}$, $\mathcal{B}$ picks a random $s_j \in Z_p$. If $a_j \in \mathbb{A}^*$, set $PK_j = g^{v_j} = g^{\frac{a'}{s_j}}$; otherwise, $PK_j = g^{v_j} = g^{s_j}$. Then, $\mathcal{B}$ sends PK to $\mathcal{A}$, where $PK = \{\mathbb{G}_0, g, h, g^\alpha, e(g,g)^\beta, e(g,h)^\beta, \{PK_j \mid \forall a_j \in \mathcal{L}\}\}$.

- *Phase* 1: $\mathcal{A}$ adaptively submits any attribute set $S \in \mathcal{L}$ to $\mathcal{B}$ with the restriction that $S \nvDash \mathbb{A}^*$. In response, $\mathcal{B}$ picks $\hat{r}, \tilde{r} \in \mathbb{Z}_p$ at random, computes $g^r = \frac{g^{\hat{r}}}{g^{b'}} = g^{\hat{r}-b'}$, $g^{\beta+\alpha r} = g^{\beta'+a'b'+a'(\hat{r}-b')} = g^{\beta'+a'\hat{r}}$, $g^{\alpha\hat{r}}h^{\tilde{r}}$, $h^{\alpha\hat{r}}h^{\tilde{r}}$, $g^{\tilde{r}}$. For each $a_j \in S$, if $a_j \in \mathbb{A}^*$, $\mathcal{B}$ computes $g^{\frac{\alpha\hat{r}}{v_j}} = g^{s_j\hat{r}}$ and $h^{\frac{\alpha\hat{r}}{v_j}} = h^{s_j\hat{r}}$; otherwise, $g^{\frac{\alpha\hat{r}}{v_j}} = g^{\frac{a'\hat{r}}{s_j}}$ and $h^{\frac{\alpha\hat{r}}{v_j}} = h^{\frac{a'\hat{r}}{s_j}}$. Afterwards, $\mathcal{B}$ answers $\mathcal{A}$ with the corresponding secret key $SK = \{g^{\beta'+\alpha\hat{r}}, g^{\alpha\hat{r}}h^{\tilde{r}}, h^{\alpha\hat{r}}h^{\tilde{r}}, g^{\tilde{r}}, \{g^{\frac{\alpha\hat{r}}{v_j}}, h^{\frac{\alpha\hat{r}}{v_j}} \mid \forall a_j \in S\}\}$.
- *Challenge*: $\mathcal{A}$ chooses two equal-length challenge messages (m_0, m_1), a set of target keywords $KW^* = \{kw_1^*, \dots, kw_t^*\}$ and submits them to $\mathcal{B}$. Then, $\mathcal{B}$ randomly chooses $r_1, r_2, s_1 \in \mathbb{Z}_p$, sets $g^{s_2} = g^{c'}$, $h^{s_2} = g^{c'x}$, $e(g,h)^{\beta s_2} = \mathcal{Z}^x \cdot e(g,g)^{\beta' c' x}$ and generates

$$CT_1'^* = \left\{ \begin{matrix} \mathcal{T}^*, g^{s_1}, h^{s_1}, \{C_{j,1} = g^{v_j q_{x^*}(0)}, \\ C_{j,2} = h^{v_j q_{x^*}(0)} \mid \forall a_j = att(x^*) \in \mathcal{X}^*\} \end{matrix} \right\},$$

where $\mathcal{X}^*$ is a set of attributes corresponding with all leaf nodes in $\mathcal{T}^*$, $CT_1^* = \Big\{g^{s_2}, g^{s_1}g^{s_2}, h^{s_1}, h^{s_2}, CT_1'^*\Big\}$, $CT_2'^* = \Big\{\{C_1^{i*} = h^{\frac{s_2}{H(kw_i^*)r_1}}, C_2^{i*} = g^{\frac{1}{H(kw_i^*)r_2}} \mid \forall i \in [1,t]\}\Big\}$, and $CT_2^* = \Big\{e(g,h)^{\beta s_2}, g^{\frac{1}{r_1}}, CT_2'^*\Big\}$.
After that, $\mathcal{B}$ randomly picks $\theta' \in \{0,1\}$, sets $C^* = m_{\theta'} \cdot e(g,g)^{\beta s_2} \cdot e(g,g)^{\frac{1}{r_2}}$ where $e(g,g)^{\beta s_2} = \mathcal{Z} \cdot e(g,g)^{\beta' c'}$, and returns $\mathcal{A}$ the final challenge ciphertext $CT^* = \{\mathcal{T}^*, C^*, CT_1^*, CT_2^*\}$.
- *Phase* 2: This phase is the same as Phase 1.
- *Guess*: $\mathcal{A}$ outputs a guess bit θ'' of θ'. If $\theta'' = \theta'$, $\mathcal{B}$ guesses $\theta = 0$ which indicates that $\mathcal{Z} = e(g,g)^{a'b'c'}$ in the above game. Otherwise, $\mathcal{B}$ guesses $\theta = 1$ i.e., $\mathcal{Z} = \mathcal{R}$.

If $\mathcal{Z} = \mathcal{R}$, then CT^* is random from the view of $\mathcal{A}$. Hence, $\mathcal{B}'s$ probability to guess θ correctly is

$$\Pr\left[\mathcal{B}\left(g, g^{a'}, g^{b'}, g^{c'}, \mathcal{Z} = \mathcal{R}\right) = 1\right] = \frac{1}{2}. \tag{23}$$

Else $\mathcal{Z} = e(g,g)^{a'b'c'}$, then CT^* is available and $\mathcal{A}'s$ advantage of guessing θ' is ϵ. Therefore, $\mathcal{B}'s$ probability to guess θ correctly is

$$\Pr\left[\mathcal{B}\left(g, g^{a'}, g^{b'}, g^{c'}, \mathcal{Z} = e(g,g)^{a'b'c'}\right) = 0\right] = \frac{1}{2} + \epsilon. \tag{24}$$

In conclusion, $\mathcal{B}'s$ advantage to win the above security game is

$$\begin{aligned} Adv(\mathcal{B}) = \frac{1}{2}\Big(& \Pr\left[\mathcal{B}\left(g, g^{a'}, g^{b'}, g^{c'}, \mathcal{Z} = e(g,g)^{a'b'c'}\right) = 0\right] \\ & + \Pr\left[\mathcal{B}\left(g, g^{a'}, g^{b'}, g^{c'}, \mathcal{Z} = \mathcal{R}\right) = 1\right] \Big) - \frac{1}{2} = \frac{1}{2}\epsilon. \end{aligned} \tag{25}$$

Theorem 2. *Supposed that a PPT adversary $\mathcal{A}$ can break the IND-CKA security of ABDKS with a non-negligible advantage $\epsilon > 0$, then there exists a PPT simulator $\mathcal{B}$ that can distinguish a DBDH tuple from a random tuple with an advantage $\frac{\epsilon}{2}$.*

Proof. The proof process of this theorem is similar to that of Theorem 1. The DBDH challenger $\mathcal{C}$ sends $\mathcal{B}$ the tuple $\langle g, g^{a'}, g^{b'}, g^{c'}, \mathcal{Z} \rangle$, in which $\mathcal{Z} = e(g,g)^{a'b'c'}$ or $\mathcal{R}$. $\mathcal{A}$ chooses a challenge access structure $\mathbb{A}^*$ initially. $\mathcal{B}$ returns public key in the same way as in Theorem 1. Then $\mathcal{A}$ adaptively submits any attribute set S and keyword set KW to $\mathcal{B}$, where $S \nvDash \mathbb{A}^*$. Since $\mathcal{B}$ can generate secret keys as in Theorem 1, it can naturally answer $\mathcal{A}$ with the corresponding trapdoor (Ta, Tk). In the challenge phase, $\mathcal{A}$ submits two challenge keyword sets KW^{0*} and KW^{1*} with equal number of keywords. $\mathcal{B}$ randomly picks $\theta' \in \{0,1\}$, generates CT_2^* with $KW^{\theta' *}$ and returns the challenge ciphertext (CT_1^*, CT_2^*). If $\mathcal{A}'s$ advantage of guessing θ' is ϵ, then $\mathcal{B}'s$ advantage to distinguish a DBDH tuple from a random tuple is $\frac{\epsilon}{2}$.

Remark 2. Previous ABKS schemes [18,23,30] are vulnerable to peer-decryption attack, in which ciphertext may be eavesdropped and decrypted by an adversary who has sufficient authority but noting about the keywords. In those schemes, the symmetric secret key (or message) is encrypted as $ck \cdot e(g,g)^{\beta s_2}$ by an access structure $\mathbb{A}$. Any adversary $\mathcal{A}$ with attribute set $S_{\mathcal{A}} \models \mathbb{A}$ can calculate $e(g,g)^{\beta s_2}$ so to get ck. While in ABDKS, the symmetric secret key (or message) is encrypted as $ck \cdot e(g,g)^{\beta s_2} \cdot e(g,g)^{\frac{1}{r_2}}$. As shown in Eq.(19), $\mathcal{A}$ without any information of keywords $KW = \{kw_i\}$ cannot compute $e(g,g)^{\frac{1}{r_2}}$ even if he has sufficient authority $S_{\mathcal{A}} \models \mathbb{A}$ to compute $e(g,g)^{\beta s_2}$. Thus, the ABDKS resists peer-decryption attack, since adversary $\mathcal{A}$ can't get ck.

6.2 Comparison with Other Schemes

From a theoretical point of view, we compared our ABDKS with a few up-to-the-minute CP-ABE schemes [18,26,28] in fog computing environment as shown in Table 2. Besides fine-grained access control, keyword search and attribute update, the ABDKS has richer functions such as dynamic keyword search and peer-decryption resistance.

Table 2. Functional comparison among previous ABKS schemes and ABDKS.

Schemes	Fine-grained access control	Keyword search	Attribute update	Dynamic keyword search	Peer-decryption resistance
[28]	√		√		
[26]	√	√			
[18]	√	√	√		
ABDKS	√	√	√	√	√

The ABDKS achieves dynamic keyword search rather than single keyword search in [18], even though, we compared the computational overheads of ABDKS with [18] from the perspective of user, fog nodes, cloud as shown in Table 3. For user, compared with [18], the user workload of ABDKS is significantly lower in the Enc phase, and slightly higher in other phases. For fog nodes, the ABDKS has more computational costs in the Enc and Trap phases than those of [18].

Table 3. Comparison of computational overhead between ABDKS and [18].

Algorithm	ABDKS		[18]	
	Fog nodes	User	Fog nodes	User
Enc	$2(1+n+t)g$	$5g+2e$	$(n+2)g$	$(n+4)g+e$
Trap	tg	$(2S+3)g$	$2(S+1)g$	$(2S+1)g$
Search	$(2Se+e+g)+(t^2e+e)$		$(S+1)e+2g$	
Dec	$(n+3)e$	$e+g$	$(n+2)e$	e

[1] e: Bilinear pairing. g: Exponentiation in group. S: Number of user attributes. n: Number of attributes in $\mathcal{T}$. t: Number of keywords.

Specifically, in the Enc phase of ABDKS, $\{C_{j,2} \mid j \leq n\}$ in CT' is generated by fog nodes, which can be regarded as the index like $\{I_{l,1} \mid l \leq n\}$ in [18]. While in [18], $\{I_{l,1} \mid l \leq n\}$ is generated by the DO. This means that, compared with [18], the ABDKS outsources part of the index generation from user to fog nodes. In the Trap phase, if only single keyword considered as in [18], the computational costs of fog nodes in ABDKS will be reduced to g rather than tg which is greatly lower than that of [18].

For cloud, in order to achieve dynamic keyword search, the ABDKS also has more computational costs in the Search phase than that of [18]. In fact, [18] can also achieve dynamic search, but if so, their computational costs of EU in the Enc phase will grow to $(nt+4)g+e$, and the costs of CS in the Search phase will grow to $(t^2S+1)e+2g$, which are both greatly larger than those of ABDKS. Therefore, the ABDKS is more efficient than [18] with the same functionality.

7 Conclusion

In this paper, we propose an attribute-based encryption with dynamic keyword search (ABDKS) scheme in fog computing environment at first. The ABDKS initially achieves dynamic keyword search and peer-decryption resistance, which makes it more functional and secure. The strict security proof has shown that it is selective CPA and CKA security. Theoretical computational complexity confirms that our proposed ABDKS is more efficient and practical than [18].

Acknowledgments

The authors are supported by National Cryptography Development Fund (Grant No. MMJJ20180210) and National Natural Science Foundation of China (Grant No. 61832012 and No. 61672019).

References

1. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: 2007 IEEE Symposium on Security and Privacy (S&P 2007), 20-23 May 2007, Oakland, California, USA. pp. 321–334 (2007)
2. Boneh, D., Crescenzo, G.D., Ostrovsky, R., Persiano, G.: Public key encryption with keyword search. In: Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004, Proceedings. pp. 506–522 (2004)
3. Bonomi, F., Milito, R.A., Zhu, J., Addepalli, S.: Fog computing and its role in the internet of things. In: Proceedings of the first edition of the MCC workshop on Mobile cloud computing, MCC@SIGCOMM 2012, Helsinki, Finland, August 17, 2012. pp. 13–16 (2012)
4. Cao, N., Wang, C., Li, M., Ren, K., Lou, W.: Privacy-preserving multi-keyword ranked search over encrypted cloud data. In: INFOCOM 2011. 30th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 10-15 April 2011, Shanghai, China. pp. 829–837 (2011)
5. Cheung, L., Newport, C.C.: Provably secure ciphertext policy ABE. In: Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28-31, 2007. pp. 456–465 (2007)
6. Cui, H., Deng, R.H., Liu, J.K., Li, Y.: Attribute-based encryption with expressive and authorized keyword search. In: Information Security and Privacy - 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3-5, 2017, Proceedings, Part I. pp. 106–126 (2017)
7. 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, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006. pp. 89–98 (2006)
8. Green, M., Hohenberger, S., Waters, B.: Outsourcing the decryption of ABE ciphertexts. In: 20th USENIX Security Symposium, San Francisco, CA, USA, August 8-12, 2011, Proceedings (2011)
9. Halpern, J.Y., Teague, V.: Rational secret sharing and multiparty computation: extended abstract. In: Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004. pp. 623–632 (2004)
10. Horváth, M.: Attribute-based encryption optimized for cloud computing. In: SOFSEM 2015: Theory and Practice of Computer Science - 41st International Conference on Current Trends in Theory and Practice of Computer Science, Pec pod Sněžkou, Czech Republic, January 24-29, 2015. Proceedings. pp. 566–577 (2015)
11. Kol, G., Naor, M.: Cryptography and game theory: Designing protocols for exchanging information. In: Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008. pp. 320–339 (2008)
12. Li, H., Yang, Y., Luan, T.H., Liang, X., Zhou, L., Shen, X.S.: Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data. IEEE Trans. Dependable Sec. Comput. **13**(3), 312–325 (2016)
13. Li, J., Shi, Y., Zhang, Y.: Searchable ciphertext-policy attribute-based encryption with revocation in cloud storage. International Journal of Communication Systems **30**(1), n/a–n/a (2015)
14. Li, J., Huang, X., Li, J., Chen, X., Xiang, Y.: Securely outsourcing attribute-based encryption with checkability. IEEE Trans. Parallel Distrib. Syst. **25**(8), 2201–2210 (2014)

15. Li, J., Wang, Q., Wang, C., Cao, N., Ren, K., Lou, W.: Fuzzy keyword search over encrypted data in cloud computing. In: INFOCOM 2010. 29th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 15-19 March 2010, San Diego, CA, USA. pp. 441–445 (2010)
16. Liu, H., Li, X., Xu, M., Mo, R., Ma, J.: A fair data access control towards rational users in cloud storage. Inf. Sci. **418**, 258–271 (2017)
17. Miao, Y., Ma, J., Liu, X., Wei, F., Liu, Z., Wang, X.A.: m2-abks: Attribute-based multi-keyword search over encrypted personal health records in multi-owner setting. J. Medical Systems **40**(11), 246:1–246:12 (2016)
18. Miao, Y., Ma, J., Liu, X., Weng, J., Li, H., Li, H.: Lightweight fine-grained search over encrypted data in fog computing. IEEE Transactions on Services Computing pp. 1–1 (2018)
19. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005, Proceedings. pp. 457–473 (2005)
20. Song, D.X., Wagner, D.A., Perrig, A.: Practical techniques for searches on encrypted data. In: 2000 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 14-17, 2000. pp. 44–55 (2000)
21. Stojmenovic, I., Sheng, W.: The fog computing paradigm: Scenarios and security issues. In: Federated Conference on Computer Science and Information Systems, IEEE Computer Society. pp. 1–8 (2014)
22. Stojmenovic, I., Wen, S., Huang, X., Luan, H.: An overview of fog computing and its security issues. Concurrency and Computation: Practice and Experience **28**(10), 2991–3005 (2016)
23. Sun, W., Yu, S., Lou, W., Hou, Y.T., Li, H.: Protecting your right: Verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. IEEE Trans. Parallel Distrib. Syst. **27**(4), 1187–1198 (2016)
24. Wang, H., Dong, X., Cao, Z., Li, D.: Secure and efficient attribute-based encryption with keyword search. Comput. J. **61**(8), 1133–1142 (2018)
25. Wang, S., Zhou, J., Liu, J.K., Yu, J., Chen, J., Xie, W.: An efficient file hierarchy attribute-based encryption scheme in cloud computing. IEEE Trans. Information Forensics and Security **11**(6), 1265–1277 (2016)
26. Xiao, M., Zhou, J., Liu, X., Jiang, M.: A hybrid scheme for fine-grained search and access authorization in fog computing environment. Sensors **17**(6), 1423 (2017)
27. Yi, S., Li, C., Li, Q.: A survey of fog computing: Concepts, applications and issues. In: Proceedings of the 2015 Workshop on Mobile Big Data, Mobidata@MobiHoc 2015, Hangzhou, China, June 21, 2015. pp. 37–42 (2015)
28. Zhang, P., Chen, Z., Liu, J.K., Liang, K., Liu, H.: An efficient access control scheme with outsourcing capability and attribute update for fog computing. Future Generation Comp. Syst. **78**, 753–762 (2018)
29. Zhang, R., Ma, H., Lu, Y.: Fine-grained access control system based on fully outsourced attribute-based encryption. Journal of Systems and Software **125**, 344–353 (2017)
30. Zheng, Q., Xu, S., Ateniese, G.: VABKS: verifiable attribute-based keyword search over outsourced encrypted data. In: 2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014. pp. 522–530 (2014)
31. Zuo, C., Shao, J., Wei, G., Xie, M., Ji, M.: Cca-secure ABE with outsourced decryption for fog computing. Future Generation Comp. Syst. **78**, 730–738 (2018)