IEEE Access
Multidisciplinary ⁝ Rapid Review ⁝ Open Access Journal

Received January 31, 2019, accepted March 8, 2019, date of publication March 15, 2019, date of current version April 2, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2905346

A Practical Attribute-Based Document Collection Hierarchical Encryption Scheme in Cloud Computing

JUNSONG FU[1] AND NA WANG[2]
[1]School of Cyberspace Security, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]School of Computer Science, Beijing University of Posts and Telecommunications, Beijing 100876, China

Corresponding author: Na Wang (nawang@bupt.edu.cn)

This work was supported in part by the Fundamental Research Funds for the Central Universities, and in part by the China Postdoctoral Science Foundation.

ABSTRACT Ciphertext-policy attribute-based encryption can provide fine-grained access control and secure data sharing to the data users in cloud computing. However, the encryption/decryption efficiency of existing schemes can be further improved when encrypting a large document collection. In this paper, we propose a practical Ciphertext-Policy Attribute-Based Hierarchical document collection Encryption scheme named CP-ABHE. By practical, we mean that CP-ABHE is more efficient in both computation and storage space without sacrificing data security. In CP-ABHE, we first construct a set of integrated access trees based on the documents' attribute sets. We employ the greedy strategy to build the trees incrementally and grow the trees dynamically by combining the small ones. Then, all the documents on an integrated access tree are encrypted together. Different to existing schemes, the leaves in different access trees with the same attribute share the same secret number, which is employed to encrypt the documents. This greatly improves the performance of CP-ABHE. The security of our scheme is theoretically proved based on the decisional bilinear Diffie–Hellman assumption. The simulation results illustrate that CP-ABHE performs very well in terms of security, efficiency, and the storage size of the ciphertext.

INDEX TERMS Cloud computing, attribute-based document collection encryption, encryption/decryption efficiency, information security.

I. INTRODUCTION

Cloud computing collects and organizes a large amount of information technique resources to provide secure, efficient, flexible and on demand services [29]. Attracted by these advantages, more and more enterprise and individual users trend to outsource the local documents to the cloud. In general, the documents need to be encrypted before being outsourced to protect them against leaking. If the data owner wants to share these documents with an authorized data user, they can employ any searchable encryption techniques [2], [6], [9], [14], [30], [31] or privacy-preserving multi-keyword document search schemes [3], [5], [8], [37] to achieve this goal. However, all these schemes cannot provide fine-grained access control mechanisms to the encrypted documents.

Attribute-based encryption (ABE) schemes can provide complicated systems to diversify the data users' access paths. In ABE schemes, each document is encrypted individually and a data user can decrypt a document if her attribute set matches the access structure of the document. Existing ABE schemes can be divided into Key-Policy ABE (KP-ABE) schemes [11], [12], [15], [16], [20], [24], [25], [28] and Ciphertext-Policy ABE (CP-ABE) schemes [1], [7], [10], [19], [21]–[23], [27], [34]. Compared with KP-ABE schemes, CP-ABE schemes are more flexible and suitable for general applications. In the following, we first analyze the existing ABE schemes in detail and further present the novelty and innovation of the CP-ABHE scheme proposed in this paper. For convenience, we choose the schemes in [1] and [11] as typical examples of KP-ABE scheme and CP-ABE scheme, respectively.

The associate editor coordinating the review of this manuscript and approving it for publication was Zheng Yan.

2169-3536 © 2019 IEEE. Translations and content mining are permitted for academic research only.
Personal use is also permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}_0$ and e be a bilinear map, $e : \mathbb{G}_0 \times \mathbb{G}_0 = \mathbb{G}_1$. Further, let $H : \{0, 1\}^* \rightarrow \mathbb{G}_0$ be a hash function which can map an attribute string to a random element in $\mathbb{G}_0$. Assume that we need to encrypt a set of documents $\mathcal{F} = \{F_1, F_2, \cdots, F_N\}$. Attribute set $\mathcal{A} = \{A_1, A_2, \cdots, A_M\}$ is the common attribute dictionary of both documents and data users. We further assume that document F_i is related with a set of attributes, denoted as $att(F_i)$. We encrypt $\mathcal{F}$ in two phases. First, each document F_i is encrypted by a proper symmetric encryption algorithm with a unique content key ck_i. Second, all the content keys of $\mathcal{F}$ are encrypted by ABE schemes. Note that, both the ciphtexts of F_i and ck_i are provided to data users. In decryption process, data users need to first decrypt ck_i based on their attribute-related secret keys and then decrypt document F_i based on ck_i. In this way, ciphertext of F_i can be decrypted only by the data users who have the matched attributes with $att(F_i)$. Considering that the first encryption phase does not fall in the scope of this paper, we focus on the second phase which is strongly related to the proposed scheme.

To encrypt all the content keys of $\mathcal{F}$, KP-ABE scheme in [9] is executed as follows.

For each content key ck_i with attribute set $att(F_i)$ and access tree $\mathcal{T}$, the public key is calculated as $PK = \{e(g, g)^{\alpha}, \forall j \in att(ck_i), T_j = g^{r_j}\}$ where α is a random number in $\mathbb{Z}_p$ and r_j is a number randomly chosen from $\mathbb{Z}_p$ for attribute j. Then the ciphertext of ck_i is calculated as $CT_{ck_i} = \{\mathcal{T}, ck_i \cdot e(g, g)^{\alpha s}, \forall j \in att(F_i), E_j = T_j^s\}$ where s is a random number in $\mathbb{Z}_p$. The above process must be executed for N times to encrypt all the content keys. The total number of elements in the ciphertext can be calculated as $N_{cip} = N + \sum_{i=1}^{N} |att(F_i)|$, where $|att(F_i)|$ denotes the number of attributes in $att(F_i)$. To decrypt the ciphertext of ck_i, a data user needs to store the secret key $SK = \{\forall j \in att(F_i), D_j = g^{\frac{q_j(0)}{r_j}}\}$, where $q_j(x)$ is the polynomial of the leaf node in $\mathcal{T}$ corresponding to attribute j. To decrypt all the content keys, N secret keys for the N access trees need to be stored by a data user and the number of total secret values in the keys can be calculated as $N_{sk} = \sum_{i=1}^{N} |att(F_i)|$. It can be observed that N_{sk} increases with the increasing of documents' number and we call this as **the secret key expanding problem**.

To encrypt all the content keys of $\mathcal{F}$, CP-ABE scheme in [12] is executed as follows.

For each content key ck_i with attribute set $att(F_i)$ and access tree $\mathcal{T}$, the public key is calculated as $PK = \{h = g^{\beta}, e(g, g)^{\alpha}\}$, where β and α are random numbers in Z_p. Then the scheme calculates the ciphertext of ck_i as $CT_{ck_i} = \{\mathcal{T}, ck_i \cdot e(g, g)^{\alpha s}, C = h^s, \forall j \in att(F_i), C_j = g^{q_j(0)}, C'_j = H(j)^{q_j(0)}\}$, where $q_j(x)$ is the polynomial of the leaf node in $\mathcal{T}$ corresponding to attributes j. Similar to KP-ABE, the above process is also executed for N times to encrypt all the content keys. The total number of elements in the ciphertext can be calculated as $N_{cip} = 2 * N + 2 * \sum_{i=1}^{N} |att(F_i)|$. Apparently, N_{cip} greatly expands with the increasing of documents' number. To decrypt the ciphertext of ck_i, the secret key of a data user is calculated as $SK = \{D = g^{\frac{(\alpha+r)}{\beta}}, \forall j \in att(F_i), D_j = g^r H(j)^{r_j}, D'_j = g^{r_j}\}$ where r is a random number in $\mathbb{Z}_p$ and r_j is a random number chosen from $\mathbb{Z}_p$ for attribute j.

Both the KP-ABE and CP-ABE schemes are impractical to encrypt a large document collection because of the following reasons. First, the encryption process in both the two schemes is executed N times, leading to high computation complexity. Second, there is a tradeoff between the size of the content keys' ciphertext and data users' secret keys. In KP-ABE, the number of secret values in a data user's secret key is extremely large for a document collection, imposing a heavy burden on the data user. In CP-ABE, the size of the ciphertext is extremely large. Consequently, CP-ABE scheme increases the data transmission amount between the cloud server and data users, which is a huge challenge for the network. This is reasonable considering that the access structure of each document must be embedded into the ciphertext or the secret keys. Third, decrypting the ciphertext is also time-consuming considering that each document is encrypted individually. Recently, Wang *et al.* [33] attempted to improve the encryption efficiency and propose a file hierarchy attribute-based encryption scheme named FH-CP-ABE. However, this scheme focused only on how to encrypt a set of documents that share an integrated access tree and hence it also cannot be directly employed to encrypt a document collection.

In this paper, we design an attribute-based document hierarchical encryption scheme named CP-ABHE which performs well in terms of computation and storage space efficiency. The scheme consists of two modules including integrated access tree construction and tree encryption. We first propose an algorithm to generate the integrated access trees for a document collection. The most important design goal of the algorithm is decreasing the number of integrated access trees which can greatly improve the encryption/decryption efficiency.

Then, the documents that share an access tree are encrypted together. Each node in a tree is assigned with a secret number which is used to encrypt the content keys of documents on the node. The secret numbers of the nodes are constructed in a bottom-up manner and it is totally different to the methods in KP-ABE, CP-ABE, and FH-CP-ABE schemes. In this way, the number of all the elements in content keys' ciphertext is smaller than $2 * N$ and it is much smaller than that in KP-ABE scheme and CP-ABE scheme. In addition, we decrease the number of secret values in the keys stored by the data users compared with KP-ABE. To decrypt all the documents in $\mathcal{F}$, only $2 * |\mathcal{A}| + 1$ secret values need to be stored by a data user, where $|\mathcal{A}|$ is the size of $\mathcal{A}$. In conclusion, both the encryption/decryption efficiency and storage efficiency of CP-ABHE are very high. The security of our scheme is

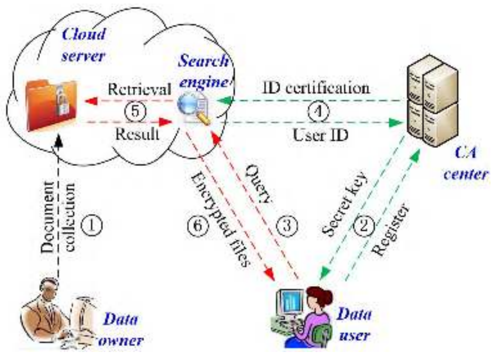


FIGURE 1. The architecture of document outsourcing and sharing.

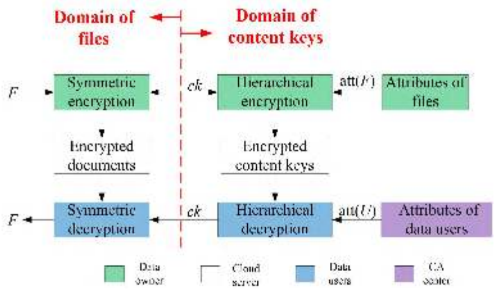


FIGURE 2. The flow chart of document encryption and decryption.

proved theoretically and we evaluate the scheme's efficiency by a series of simulation.

The contributions of this paper are mainly summarized as follows:

- An algorithm to construct the integrated access trees incrementally for the document collection is proposed and it can significantly decrease the number of the access trees.
- A document collection hierarchical encryption scheme is proposed. All the documents that share an integrated access tree are encrypted together which can significantly improve the encryption/decryption efficiency. Moreover, the **secret key expanding problem** is solved properly.
- The security of CP-ABHE is theoretically proved and the effectiveness of the integrated access tree construction algorithm is analyzed in detail. In addition, a thorough comparison between CP-ABHE, KP-ABE, and CP-ABE in terms of encyption/decryption efficiency and storage space is provided.

The rest of this paper is organized as follows: We present the system model and preliminaries in Section 2. The detailed process of access trees construction is given in Section 3 and Section 4 discusses the scheme to encrypt the document collection. We analyze the security and efficiency of our scheme theoretically in Section 5. Section 6 evaluates the performance of the integrated access trees and the efficiency of CP-ABHE is analyzed and simulated in Section 7. In Section 8, the related work is provided and this paper is concluded in Section 9.

II. SYSTEM MODEL AND PRELIMINARIES

A. SYSTEM MODEL

Fig. 1 describes the document outsourcing and sharing system which mainly comprises four entities: the data owner, data user, certificate authority (CA) center and cloud server. The entire process of querying a set of interested documents for a data user includes 6 phases:

① Data owner is responsible for collecting documents and assigning a proper attribute set to each document. The documents are encrypted in two phases. Each document is first encrypted by a symmetric encryption algorithm with a unique content key. Then, the content keys are encrypted by ABE-schemes. At last, both the encrypted documents and content keys are outsourced to the cloud server.

② To search the interested documents in the cloud server, a data user first needs to register herself to the CA center. Then, the CA center assigns an attribute set to the data user and sends an attribute-related secret key to the data user.

③ The authorized data user can send query requests to the cloud server. In this paper, we assume that the cloud server is trustable. Otherwise, we may need to further integrate the secure kNN algorithm [35] into our scheme to encrypt the document vectors and query vectors [3], [5], [8], [37].

④ Once a query request is received, the cloud server first communicates with the CA center to check the identity of the data user and an ID certification message is received if the data user is authorized.

⑤ For an authorized query, the cloud server employs a search engine to search the encrypted document collection and get the related ciphertexts to the query. Note that only the documents whose attributes match the data user are returned.

⑥ Having received the encrypted documents and content keys, the data user first decrypts the content keys by her attribute-related secret key and then decrypt the documents based on the content keys. At last, the document retrieval process is completed.

The whole document outsourcing and sharing system contains numerous research lines. In this paper, we restrict our attention to the document collection encryption/decryption process and ignore the other technical challenges such as symmetric encryption algorithms and encrypted document search algorithms. The flowchart of encrypting/decrypting a document collection is shown in Fig. 2. Given a set of documents, the data owner first randomly selects a set of content keys $ck = \{ck_1, ck_2, \cdots, ck_N\}$ which are used to encrypt the documents in $\mathcal{F}$ symmetrically, i.e., $\mathcal{C}_i = E_{ck_i}(F_i), i = 1, 2, \cdots N$ where $\mathcal{C}_i$ is the ciphertext of F_i. Then, the content keys are encrypted by the proposed scheme CP-ABHE. Encrypting the document collection in a two-tier manner is reasonable considering that directly encrypting the documents based on the bilinear map is of extremely large computation complexity and this is impractical. At last, all the encrypted documents, hierarchical access structures and encrypted content keys are outsourced to the cloud server. In the decryption process, the data users first decrypt the content keys with their secret keys and further decrypt the documents based on the decrypted content keys. Encrypting the documents symmetrically by the content keys is out of this paper's scope and we mainly discuss how to encrypt/decrypt the content keys in detail.

B. DEFINITIONS AND PRELIMINARY TECHNIQUES

1) MONOTONE ACCESS STRUCTURE

Let $\mathbb{A} = \{A_1, A_2, \cdots, A_M\}$ be a set of attributes. A collection $\mathbb{A} \subseteq 2^{\mathcal{A}}$ is monotone: Given $\forall B, C$, if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. A monotone access structure of a document is a monotone collection $\mathbb{A}$ of non-empty subsets of $\mathcal{A}$, i.e., $\mathbb{A} \subseteq 2^{\mathcal{A}} \backslash \{\emptyset\}$. 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, we assume that the access structure of each document is monotone.

2) BILINEAR MAPS

Let $\mathbb{G}_0$ and $\mathbb{G}_1$ be two multiplicative groups of prime order p. Naturally, they are cyclic groups and each non-identity element in $\mathbb{G}_i (i = 0, 1)$ is a generator of the group $\mathbb{G}_i$. Let g be a generator of $\mathbb{G}_0$ and e be a bilinear map, $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$, with the following properties:

1) Bilinearity: For all $u, v \in \mathbb{G}_0$ and $a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$.
2) Non-degeneracy: $e(g, g) \neq 1$.
3) Distributivity: For $u, v, w \in \mathbb{G}_0$ and $a, b, c \in \mathbb{Z}_p$, $e(u^a, v^b w^c) = e(u^a, v^b) e(u^a, w^c)$.

In addition, $\mathbb{G}_0$ is a bilinear group if the group operations in $\mathbb{G}_0$ and the bilinear map, $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$, are both efficiently computable.

3) LAGRANGE INTERPOLATION

Given a set of data points $\{(x_1, y_1), (x_2, y_2), \cdots, (x_n, y_n)\}$ and $x_i \neq x_j$ if $i, j \in \{1, 2, , n\}, i \neq j$, they uniquely decide a $n - 1$ degree polynomial which can be constructed by Lagrange interpolation algorithm. Specifically, the polynomial can be represented as follows:

$$f(x) = \sum_{i \in \{1,2,\cdots,n\}} (y_i \prod_{j \in \{1,2,\cdots,n\}, j \neq i} \frac{x - x_j}{x_i - x_j}),$$

where $\prod_{j \in \{1,2,\cdots,n\}, j \neq i} \frac{x-x_j}{x_i-x_j}$ is the Lagrange Coefficient. For convenience, we denote the coefficient as $\Delta_{i,S}$ for $i \in \mathbb{Z}_p$ and a set, S, of elements in $\mathbb{Z}_p$ and it is defined as $\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} (x - j)/(i - j)$.

4) DECISIONAL BILINEAR DIFFIE-HELLMAN (BDH) ASSUMPTION

Assume that a, b, c, t are randomly selected from $\mathbb{Z}_p$ and g is a generator of $\mathbb{G}_0$. The decisional BDH assumption is that no probabilistic polynomial-time algorithm $\mathcal{B}$ can distinguish the tuple $(A = g^a, B = g^b, C = g^c, e(g, g)^{abc})$ from the tuple $(A = g^a, B = g^b, C = g^c, e(g, g)^t)$ with more than a negligible advantage.

In addition, we say that an adversary $\mathcal{A}dv$ can solve the decisional BDH problem with an advantage ϵ if:

$$\begin{aligned} &|Pr[\mathcal{A}dv(g, g^a, g^b, g^c, e(g, g)^{abc}) = 0] \\ &\quad -Pr[\mathcal{A}dv(g, g^a, g^b, g^c, e(g, g)^z) = 0]| \geqslant \epsilon \end{aligned}$$

5) SELECTIVE-SET SECURITY GAME

In this paper, [1], [11], and [33] to prove our scheme's security. The game is composed of 6 phases and they are presented as follows.

Init: The adversary declares an access tree with a set of attributes S that he wants to be challenged upon.

Setup: The challenger runs the Setup algorithm of CP-ABHE to generate the public parameters which are provided to the adversary.

Query Phase 1: The adversary is allowed to issue queries to obtain the secret keys of any access structure $\mathbb{A}^*$ with attribute set S', where $S \nsubseteq S'$.

Challenge: The adversary provides two different messages M_0 and M_1 with equal length to the challenger. The challenger randomly flips a coin μ and encrypts M_μ with attribute set S. Then the encrypted message is sent to the adversary.

Query phase 2: Query Phase 1 is repeated.

Guess: Based on the obtained information, the adversary outputs a guess μ' of μ.

We say that our scheme is secure if all the polynomial time adversaries have at most a negligible advantage in the game, where the advantage of the adversary is defined as $\left|Pr(\mu' = \mu) - 1/2\right|$. Note that, if our scheme can resist the Selective-set security game, it naturally resists collusion attack which is an extremely important property for the ABE schemes. This can be explained by the fact that the adversary can take multiple secret key queries before and after the challenge phase.

III. INTEGRATED ACCESS STRUCTURE OF A DOCUMENT COLLECTION

A. ACCESS POLICY OF DOCUMENTS AND ACCESS TREES

In this paper, we assume that each document F_i is of several attributes in $att(F_i)$ and F_i can be accessed only by the data users who possess all the attributes in $att(F_i)$. As shown in Fig. 3(a), we assume that the attribute dictionary of a document collection includes three basic attributes including "communication", "computer" and "network". Each document has at least one attribute and some documents may have two or three attributes such as the documents in region A, B, C and D. In this case, the documents in region A can be accessed by the data users who own all the three roles of communication researcher, computer researcher, and network researcher. Apparently, the access structure of a document is monotone. As an example, a data user who owns the attributes of communication and computer researcher can access the documents in region B. Meanwhile, any other data users who have at least these two attributes can also access the documents in region B. Compared with the threshold-based access policy proposed in [1], [11], and [33], our access policy is

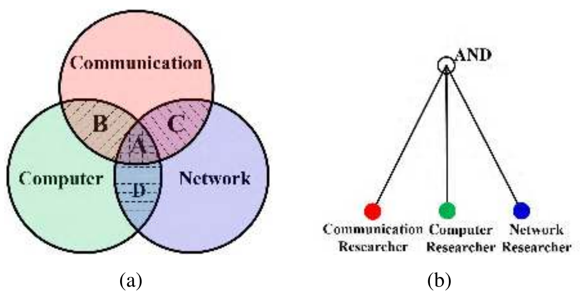


FIGURE 3. (a) Assumption of access control strategy. (b) The access tree of the documents in region *A*.

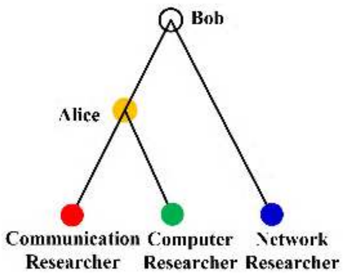


FIGURE 4. Integrated access tree of Alice and Bob.

stricter and more suitable for the documents with high privacy requirements such as personal health records [36].

We can represent the access structure of a document by an access tree $\mathcal{T}$. Under our access policy, the leaf nodes in the tree represent the attributes related to the document and the root node represents an "AND" gate. The access tree of a document in region A is shown in Fig. 3(b) and the tree contains three leaf nodes representing three attributes. The root node represents an "AND" gate. In this case, if the leaf node set of an access tree is a subset of another access tree's leaf node set, we can combine these two trees to a new tree which is called an integrated access tree. Apparently, each non-leaf node in the integrated access tree also represents an "AND" gate. The integrated access tree of Alice, who is a communication and computer researcher, and Bob, who is a communication, computer, and network researcher, is shown in Fig. 4. In a document collection, the attribute sets of the documents are various and each document has an access tree. How to combine these single access trees to a small number of integrated trees is a huge challenge. To our knowledge, given a set of single access trees, minimizing the number of integrated access trees is an NP-hard problem and hence we propose a greedy-strategy-based integrated access tree construction algorithm in Section III-B.

B. ACCESS STRUCTURE OF A DOCUMENT COLLECTION

In this section, we present the process of constructing the integrated access trees of a document collection $\mathcal{F} = \{F_1, F_2, \cdots, F_N\}$ with identifiers $\{f_1, f_2, \cdots, f_N\}$. Let $\mathcal{T}$ be an integrated access tree of a set of documents and all the integrated access trees compose the access structure of the whole document collection. We first define some notations and functions about the integrated access trees. The number

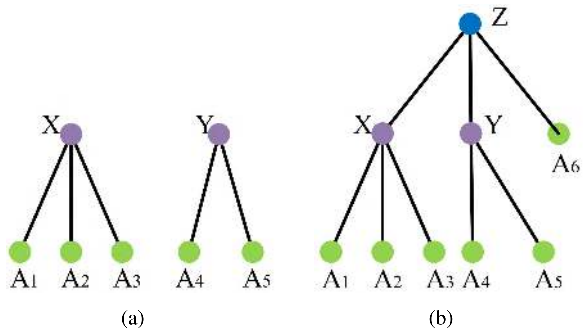


FIGURE 5. (a) Assumption of access control strategy. (b) The access tree of the documents in region *A*.

of the child nodes of an internal node x is denoted as num_x. Note that, the child nodes of x means the nodes which are derived from x and directly connected with x. Function $att(x)$ denotes the associated attributes with the node x, i.e., the attributes represented by all the leaves derived from node x. Each node x in a tree has a unique numerical identifier and it is returned by $index(x)$. Moreover, $att(\mathcal{T})$ returns all the attributes in the tree.

We say that an attribute set S matches an access tree $\mathcal{T}$ if and only if S exactly equals to $att(\mathcal{T})$. As an example in Fig. 5(a), tree X matches S if and only if $S = \{A_1, A_2, A_3\}$ and, in this time, we denotes it as $S(X) = 0$. Moreover, we say that an attribute set S covers tree X if and only if S is a proper superset of $att(X)$. Apparently, S covers X if $S = \{A_1, A_2, A_3, A_4\}$and we denote it as $S(X) = 1$. Roughly speaking, we construct $S_{\mathcal{T}}$ of a document collection in an incremental manner and $S_{\mathcal{T}}$ is updated for one time once a new document is entered. The integrated access trees in $S_{\mathcal{T}}$ grow by continuously combining the small access trees. The pseudo-code of constructing the access structure for a document collection $\mathcal{F}$ is presented in Algorithm 2.

In the initial, we sort the documents in ascent order based on the number of their attributes. Then, the access tree of the first document $F_1{}'$ is set as the first integrated access tree and the identifier, $f_1{}'$, of $F_1{}'$ is inserted to the root node of the tree. Given a set of integrated access trees, $S_{\mathcal{T}}$, we now discuss how to update the trees when a new document, $F_i{}'$, arrives. The attribute set of the new document $att(F_i{}')$ faces three cases based on its relation to the trees in $S_{\mathcal{T}}$. Specifically, $att(F_i{}')$ can match an existing tree, cover some trees, or $att(F_i{}')$ neither matches nor covers the existing trees. We first orderly scan the access trees in $S_{\mathcal{T}}$ to find a tree that matches $att(F_i{}')$. If the tree exists, we insert $f_i{}'$ to the root node of the tree. Otherwise, we orderly rescan the access trees in $S_{\mathcal{T}}$ to find a tree X which is covered by $att(F_i{}')$. If the tree X exists, we continue to search the trees to find a tree Y which is covered by attribute set $att(F_i{}') \setminus att(X)$. If the tree Y also exists, we continue to search the trees to find a tree which is covered by attribute set $att(F_i{}') \setminus att(X) \setminus att(Y)$. We iterate the above process until all the existing access trees are scanned. If all the attributes of the found trees together form $att(F_i{}')$, a larger access tree with root node r is constructed in which

Algorithm 1 BuildingAccessStructure

```
Input: Document collection $\mathcal{F} = F_1, F_2, \cdots, F_N$ with attribute sets $\{att(F_1), att(F_2), \cdots, att(F_N)\}$
Output: A set of integrated access trees $S_\mathcal{T}$
1:  Sort the files in $\mathcal{F}$ in ascending order based on the number of their attributes and obtain $F' = \{F_1{}', F_2{}', \cdots, F_N{}'\}$ with identifiers $\{f_1{}', f_2{}', \cdots, f_N{}'\}$;
2:  $S_\mathcal{T} = \{\}, C = \{\}$;
3:  for $i = 1 : N$ do
4:      $S = att(F_i')$;
5:      Scan the access trees in $S_\mathcal{T}$ in order;
6:      if $S$ matches an scanned access tree $X$, i.e., $S(X) = 0$ then
7:          Insert the identifier of $F_i'$ into the root node of $X$;
8:          break;
9:      end if
10:     Rescan the access trees in $S_\mathcal{T}$ in order;
11:     for a scanned access tree $Y$ in $S_\mathcal{T}$ do
12:         if $S$ covers $Y$, i.e., $S(Y) = 1$ then
13:             $C = C \cup Y, S = S \setminus att(Y)$;
14:         end if
15:     end for
16:     if $S$ is empty then
17:         Build a larger access tree $\mathcal{LT}$ with root node $r$ and all the access trees in $C$ are the child nodes of $r$;
18:         Insert $f_i'$ to $r$;
19:         Insert $\mathcal{LT}$ to $S_\mathcal{T}$ and delete all the trees in $C$ from $S_\mathcal{T}$;
20:     else
21:         Build a larger access tree $\mathcal{LT}$ with root node $r$ and all the access trees in $C$ are the child nodes of $r$; In addition, all the left attributes in $S$ are also inserted to the root node $r$ as leaves;
22:         Insert $f_i'$ to $r$;
23:         Insert $\mathcal{LT}$ to $S_\mathcal{T}$ and delete all the trees in $C$ from $S_\mathcal{T}$;
24:     end if
25: end for
```

all the found trees act as child nodes of r and the document identifier $f_i{}'$ is inserted to r. However, if all the attributes of the found trees together form a proper subset $att(F_i{}')'$ of $att(F_i{}')$, all the attributes in $att(F_i{}') \setminus att(F_i{}')'$ are also inserted to the root node r as leaves. As an example, there are two integrated access trees as presented in Fig. 5(a), then when a document with attribute set $\{A_1, A_2, A_3, A_4, A_5, A_6\}$ arrives, the updated access tree is shown in Fig. 5(b). It can be observed that, though the number of documents increases, the number of the integrated trees decreases. This is of great meaning the process of encrypting the document collection. At last, if $att(F_i{}')$ neither matches nor covers an existing access trees, we just set the access tree of $F_i{}'$ as an integrated access tree and insert the access tree of $F_i{}'$ is inserted into $S_\mathcal{T}$. The above process is iterated until all the document identifiers are inserted into the integrated access trees. All the integrated access trees in $S_\mathcal{T}$ compose the access structure of the whole document collection.

At last, we discuss how to set the numerical identifiers for the nodes in the access trees. A possible approach to construct the identifiers is presented as follows:

1) If x is a leaf node and associated with attribute A_i, then its numerical identifier is set as i.
2) If x is a non-leaf node and associated with a set of attributes $\{A_i, A_j, \cdots,$ $A_k\}, 1 \leqslant i < j < \cdots < k \leq M$, then its numerical identifier is set as $ij \cdots k$.

IV. DOCUMENT COLLECTION HIERARCHICAL ENCRYPTION

In this section, we present the detailed process of encrypting a document collection $\mathcal{F} = \{F_1, F_2, \cdots, F_N\}$ by CP-ABHE. First, each document in $\mathcal{F}$ is assigned with a set of attributes which are selected from $\mathcal{A}$ and the access structure, $S_\mathcal{T}$, of $\mathcal{F}$ is constructed based on Algorithm 1 presented in Section III. Then, for each document F_i in $\mathcal{F}$, a content key ck_i is randomly selected and we symmetrically encrypt F_i based on ck_i, i.e., $\mathcal{C}_i = E_{ck_i}(F_i), i = 1, 2, \cdots N$, where $\mathcal{C}_i$ is the ciphertext of F_i. We denote all the content keys as $ck = \{ck_1, ck_2, \cdots, ck_N\}$ and then all the content keys of the documents in an individual integrated access tree can be encrypted together. Now, we discuss how to hierarchically encrypt the content keys as follows.

Setup. The setup algorithm chooses a bilinear group $\mathbb{G}_0$ of prime order p with g as a generator, a bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ and two random numbers $\alpha, \beta \in \mathbb{Z}_p$. The public key is published as:

$$PK = (\mathbb{G}_0, g, h = g^\beta, e(g, g)^\alpha),$$

and the master secret key is set as:

$$MSK = (\beta, g^\alpha).$$

Encrypt(PK, ck, S$_\mathcal{T}$). We first need generate a secret number sk_x is for each node x in the trees. In each tree, these secret numbers for the nodes are chosen in a bottom-up manner, starting from the leaves to the root node. Specifically, we randomly select a secret number $s_i \in \mathbb{Z}_p$ for each attribute A_i in $\mathbb{A}$ and s_i is assigned to all the leaves with attribute A_i in all the trees in $S_\mathcal{T}$. In other words, the secret number sk_x of the leaf node x associated with attribute A_i is s_i. Then for the internal node x with a set of child nodes S_x, the secret number sk_x is computed as $sk_x = \sum_{z \in S_x} sk_z \Delta_{i,S_x'}(index(x))$ where $i = index(z)$, $S_x' = \{index(z), z \in S_x\}$, $index(x)$ is the numerical identifier of node x. By treating each child node z in S_x as a data point with coordinate $(index(z), sk_z)$, the Lagrange interpolation algorithm could be employed to construct a $|S_x| - 1$ order polynomial which crosses all the data points in S_x, where $|S_x|$ is the number of nodes in S_x. In this way, the secret number of node x can be calculated by plugging $index(x)$ into the polynomial. In theory, each child node maintains a share of the secret number of the parent node. To recover the secret number of a node, the data users need to collect all the necessary shares which are hidden in their secret keys. By iterating the above process, each node in the integrated access structure can be assigned with a secret number.

Then, we encrypt the content keys by the assigned secret numbers. Assume that the file identifiers $\{f_m, \cdots, f_n\}$ in a node x can be returned by function $file(x)$ and then we encrypt all the related content keys $\{ck_m, \cdots, ck_n\}$ based on the same secret number sk_x. Let Y be the set of the leaves in an integrated tree $\mathcal{T}$. All the content keys related with $\mathcal{T}$ are encrypted together and the ciphertext is constructed as follows:

$$CT_{\mathcal{T}} = (\mathcal{T}, \forall x \in \mathcal{T}, f_i \in file(x) : \tilde{C}_i = ck_i e(g, g)^{\alpha \cdot sk_x},$$
$$C_x{}^* = g^{sk_x}, \forall y \in Y : C_y = h^{sk_y}, C'_y = H(att(y))^{sk_y}).$$

For convenience, we call the ciphertext of all the content keys in an access tree $\mathcal{T}$ as the ciphertext of $\mathcal{T}$. By constructing the ciphertext for each integrated access tree in $S_{\mathcal{T}}$, we can get the ciphertext CT of the whole document collection as follows:

$$CT = \{\cup(CT_{\mathcal{T}}),\ \forall \mathcal{T} \in S_{\mathcal{T}}\}.$$

We further assume that $file(\mathcal{T})$ returns all the document identifiers in access tree $\mathcal{T}$ and denote the number of nodes, that contain document identifiers in $\mathcal{T}$, as $|\mathcal{T}|$. It can be observed that the ciphertext of $\mathcal{T}$ contains $|file(\mathcal{T})| + |\mathcal{T}| + 2 * |Y|$ elements in $\mathbb{G}_0$ and $\mathbb{G}_1$, where $|*|$ returns the number of elements in $*$. When encrypting a set of access trees, some redundant data can be deleted. Note that, C_y and C'_y are only related with sk_y and the further sk_y is only related with the attribute of leaf node y. As discussed previously, all the leaf nodes with the same attribute share the same secret number. Then, we can infer that the leaves, $y_1, y_2, \cdots, y_d$, of different access trees, $\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_d$, may share a same attribute A_i and hence, $C_{y_1} = C_{y_2} = \cdots = C_{y_d} = h^{s_i}$, $C_{y_1}{}' = C_{y_2}{}' = \cdots = C_{y_d}{}' = HA_i{}^{s_i}$. Therefore, when publishing the ciphertext of all the documents, only $2 * |\mathcal{A}|$ records of C_y and C'_y need to be published. Then, the total number of elements in CT can be theoretically calculated as $N + (\sum_{\mathcal{T} \in S_{\mathcal{T}}} |\mathcal{T}|) + 2 * |\mathcal{A}|$. Considering that $(\sum_{\mathcal{T} \in S_{\mathcal{T}}} |\mathcal{T}|)$ is naturally smaller than N and $|\mathcal{A}| \ll N$, we can infer that the number of total elements in the ciphertext is always smaller than $2 * N$.

In the KP-ABE scheme [11], the number of elements in the ciphertext is always $2 * N$ and it has the close performance with CP-ABHE. However, in CP-ABE and FH-CP-ABE, each access tree is treated as a whole and the secret numbers of the leaf nodes in different access trees are totally independent with each other. Therefore, in these two schemes, the ciphertext of $S_{\mathcal{T}}$ is the collection of all the individual access trees' ciphertexts and its size is much larger than that of CT in our scheme.

KeyGen(MSK, S). The key generation algorithm takes a set of attributes S as input and output a secret key for a data user who owns all the attributes in S. We first choose a random number $r \in \mathbb{Z}_p$, and then choose a random number $r_j \in \mathbb{Z}_p$ for each attribute $A_j \in S$. Then the secret keys are computed as follows:

$$SK = (D = g^{\alpha} \cdot h^r, \forall A_j \in S : D_j = g^r \cdot H(A_j)^{r_j}, D'_j = h^{r_j}).$$

It can be observed that, for different data users, the parameter r and r_j are different. Therefore, different data users cannot collude with each other to decrypt a ciphertext which cannot be decrypted by any data user alone. However, for one data user, the secret key can be treated as a set of fragments, i.e., D, D_j, D'_j, and the fragments can be flexibly combined to construct the secret keys for different access trees. Namely, the secret key of a data user in CP-ABHE is not designed for a specific access tree. The data users' secret keys in CP-ABE and FH-CP-ABE also have similar properties. This can be explained by the fact that all these three schemes embed the access structure of the documents into the ciphertext rather than the data users' secret keys. However, in KP-ABE, the access structure is embedded in the secret keys and each secret key is designed for a specific access tree. In other words, the fragments of a secret key are meaningless unless they are employed as a whole to decrypt a specific access tree. Therefore, our mechanism can greatly simplify the data users' secret keys compared with KP-ABE scheme.

Decrypt($\mathbf{CT}_{\mathcal{T}}$, SK). We employ a recursive algorithm $DecryptNode\ (CT_{\mathcal{T}}, SK, x)$ to decrypt the content keys encrypted by node x in the tree $\mathcal{T}$ step by step. This algorithm takes as input a ciphertext $CT_{\mathcal{T}}$, a private key SK which is associated with a set of attributes S, and a node x from $\mathcal{T}$. If node x is a leaf node with attribute A_i and $A_i \in S$, then the algorithm is defined as follows:

$$\begin{aligned} DecryptNode(CT_{\mathcal{T}}, SK, x) &= \frac{e(D_i, C_x)}{e(D'_i, C'_x)} \\ &= \frac{e(g^r \cdot H(A_i)^{r_i}, h^{sk_x})}{e(h^{r_i}, H(A_i)^{sk_x})} \\ &= \frac{e(g^r, h^{sk_x}) e(H(A_i)^{r_i}, h^{sk_x})}{e(h^{r_i}, H(A_i)^{sk_x})} \\ &= e(g, g)^{\gamma\beta \cdot sk_x}. \end{aligned}$$

However, if $A_i \notin S$, we define $DecryptNode(CT_{\mathcal{T}}, SK, x) = \perp$.

When x is an internal node, the algorithm is operated recursively. First, each node $z \in S_x$ calls the function $DecryptNode(CT_{\mathcal{T}}, SK, z)$ and stores the output of the algorithm as F_z. Here, S_x denotes the set of x's child nodes. If at least one $F_z = \perp$, the function $DecryptNode(CT_{\mathcal{T}}, SK, x)$ returns $\perp$. Otherwise, we denote $i = index(z)$, $S'_x = \{index(z), z \in S_x\}$ and compute F_x as follows:

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,S'_x}(index(x))} \\ &= \prod_{z \in S_x} (e(g, g)^{r\beta \cdot sk_z})^{\Delta_{i,S'_x}(index(x))} \\ &= e(g, g)^{r\beta \cdot \sum_{z \in S_x} sk_z \cdot \Delta_{i,S'_x}(index(x))} \\ &= e(g, g)^{r\beta \cdot sk_x} \end{aligned}$$

If a data user has an attribute set S which matches $att(x)$, the data user can calculate $A = F_x = e(g, g)^{r\beta \cdot sk_x}$ by iterating the above process. Then each content key ck_i encrypted by node x with sk_x can be decrypted as follows:

$$\tilde{C}_i/(e(C_x{}^*, D)/A) = \tilde{C}_i/(e(g^{sk_x}, g^{\alpha}h^{r})/e(g, g)^{r\beta \cdot sk_x}) = ck_i.$$

At last, all the documents encrypted by ck_i can be decrypted by the data user as:

$$F_i = D_{ck_i}(C_i), \forall f_i \in file(\mathcal{T}).$$

Otherwise, the data user cannot decrypt the encrypted documents.

V. SECURITY ANALYSIS

In this section, we mainly focus our attention on analyzing the security of CP-ABHE and other security problems in the document retrieval system are out of scope in this paper. Specifically, the documents are encrypted based on symmetric encryption schemes and they are assumed to be secure if the content keys are secure. Then, we mainly restrict our attention to the security of the content keys in CP-ABHE. Methodologically, we prove the security of CP-ABHE under the Selective-Set Security Game based on the Decisional BDH assumption provided in Section II-B.

Theorem 5.1: No polynomial adversary can win the Selective-Set Security Game of CP-ABHE with a non-negligible advantage if the Decisional BDH assumption holds.

Proof: We first assume that there is a polynomial adversary $\mathcal{A}dv$ that can break through the CP-ABHE scheme with an advantage ϵ. Under the above assumption, we can design a simulator $\mathcal{B}$ that can play the Decisional BDH game with an advantage $\epsilon/2$.

First, the challenger randomly chooses two multiplicative groups, $\mathbb{G}_0$,$\mathbb{G}_1$, of prime order p. Let g be a generator of $\mathbb{G}_0$ and let e be a bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$. Four random numbers, a, b, c, t, are chosen from $\mathbb{Z}_p$. Then the challenger flips a coin v and if $v = 0$, the challenger generates a BDH tuple $(g^a, g^b, g^c, e(g, g)^{abc})$; otherwise, if $v = 1$, it constructs a random 4-tuple $(g^a, g^b, g^c, e(g, g)^t)$. At last, all the chosen elements and the generated tuple are sent to the simulator. Simulator $\mathcal{B}$ plays the game as follows:

Init: The simulator $\mathcal{B}$ runs adversary $\mathcal{A}dv$ and let $\mathcal{A}dv$ submits a set of attributes S on which $\mathcal{A}dv$ is challenged.

Setup: The simulator sets $\alpha = ab+a'$ where a' is a random number in $\mathbb{Z}_p$. Then, the simulator computes $e(g, g)^{\alpha} = e(g, g)^{ab}e(g, g)^{a'}$. It further sets $h = g^{\beta} = g^b = B$ and sends $PK = (\mathbb{G}_0, g, B, e(g, g)^{ab}e(g, g)^{a'})$ to $\mathcal{A}dv$.

Query Phase 1: The adversary $\mathcal{A}dv$ can query the secret keys SK of any access structure $\mathbb{A}^*$ with a set of attributes S' as long as $S \nsubseteq S'$. To respond the query of $\mathcal{A}dv$, simulator $\mathcal{B}$ first selects a random number $r' \in \mathbb{Z}_p$ and sets $r = r' - a$. Then it calculates $D = g^{\alpha} \cdot h^{r} = B^{r'} \cdot g^{a'}$ and, for each attribute $A_j \in S'$, the simulator randomly chooses a number $r_j \in \mathbb{Z}_p$ and calculates $D_j = g^{(r'-a)}H(A_j)^{r_j} = \frac{g^{r'}}{A}H(A_j)^{r_j}$, $D_j' = B^{r_j}$. At last, $\mathcal{B}$ sends $SK = (B^{r'} \cdot g^{a'}, \forall A_j \in S' : \frac{g^{r'}}{A}H(A_j)^{r_j}, B^{r_j})$ to the adversary.

Challenge: For convenience sake, we assume that only one content key of a file is encrypted by CP-ABHE and the ciphertext is simplified as $CT_{\mathcal{T}} = (\mathcal{T}, C_x{}^*, \tilde{C}_i, \forall y \in S' : C_y = B^{sk_y}, C_y' = H(att(y))^{sk_y})$. In the challenge process, the adversary $\mathcal{A}dv$ first sends two messages M_0 and M_1 with equal lengths to $\mathcal{B}$. Then, simulator $\mathcal{B}$ flips a coin $\mu \in \{0, 1\}$ to randomly choose a message from M_0 and M_1, and the chosen message is encrypted as follows. Simulator $\mathcal{B}$ calculates $C_x{}^* = g^{sk_x} = g^c = C$. If $v = 0$, $\tilde{C}_i$ is calculated as $\tilde{C}_i = M_{\mu}e(g, g)^{\alpha c} = e(g, g)^{abc}e(g, g)^{a'c}$; otherwise, $\tilde{C}_i$ is calculated as $\tilde{C}_i = M_{\mu}e(g, g)^t$ which is a random element of $\mathbb{G}_1$ from $\mathcal{A}dv$'s view. Moreover, C_y and C_y' are also calculated by $\mathcal{B}$. At last, the ciphertext of the chosen message is sent to $\mathcal{A}dv$.

Query phase 2: The query phase 1 is repeated.

Guess: In this process, the adversary $\mathcal{A}dv$ needs to make a guess μ' of μ based on all the obtained information and the result is sent to the simulator $\mathcal{B}$. Then, simulator makes a guess v' of v based on the guess result of $\mathcal{A}dv$. Specifically, if $\mu' = \mu$, the simulator B outputs $v' = 0$ to indicates that it is given a BDH tuple by the challenger; otherwise, it will output $v' = 1$ to indicate that it is given a random 4-tuple. Then, we can theoretically calculate the advantage of simulator $\mathcal{B}$ in playing the Decisional BDH game.

If $\mu = 0$, the adversary $\mathcal{A}dv$ sees an encryption of M_{μ} and in this case $Pr(\mu' = \mu | v = 0) = 1/2 + \epsilon$ by the initial hypothesis. Since the simulator outputs $v' = 0$ when $\mu' = \mu$, we can infer that $Pr(v' = v | v = 0) = 1/2 + \epsilon$.

If $\mu = 1$, the adversary $\mathcal{A}dv$ gains on information about μ and hence $Pr(\mu' \neq \mu | v = 1) = 1/2$. Since the simulator outputs $v' = 1$ when $\mu' \neq \mu$, we can get $Pr(v' = v | v = 0) = 1/2$.

As a consequence, the advantage of $\mathcal{B}$ in playing the DBDH game can be calculated as follows:

$$\frac{1}{2}Pr(v' = v | v = 0) + \frac{1}{2}Pr(v' = v | \mu = 1) - \frac{1}{2}$$
$$= \frac{1}{2}(\frac{1}{2} + \epsilon) + \frac{1}{2} \times \frac{1}{2} - \frac{1}{2} = \frac{\epsilon}{2}$$

Considering that the Decisional BDH assumption holds, we can infer that ϵ is a negligible advantage. In other words, the adversary cannot win the Selective-Set Security Game of CP-ABHE with a non-negligible advantage. Consequently, our scheme is secure.

VI. EFFECTIVENESS OF INTEGRATED ACCESS TREES

A. GENERATION OF ATTRIBUTE SETS

As discussed in Section IV, the access structure, $S_{\mathcal{T}}$, of the document collection greatly affects the efficiency of CP-ABHE. In this section, we analyze the properties of $S_{\mathcal{T}}$ in detail. First, we design an attribute dispatcher to assign attributes to the documents. We assume that the attribute dictionary $\mathcal{A}$ is composed of 26 letters, i.e., $\mathcal{A} = A, B, \cdots, Z$. In simulation, the attributes in A are divided into 4 categories, i.e., $C_1 = \{A, B, \cdots, G\}$, $C_2 = \{H, I, \cdots, N\}$, $C_3 = \{O, P, \cdots, T\}$, $C_4 = \{U, V, \cdots, Z\}$. The attributes in the same category are assumed to be more related with each other and we employ a parameter p_r to reflect this. In this paper, parameter p_r ranges from 0.25 to 1. This is reasonable considering that the attributes are naturally divided into clusters in real document collections and it is likely that the related attributes are assigned to a document together. For example, if a document is related with attribute ''computer'', it is natural to infer that the document is more likely to be related with attribute ''network'' than other attributes such as ''economic'' and ''finance''.

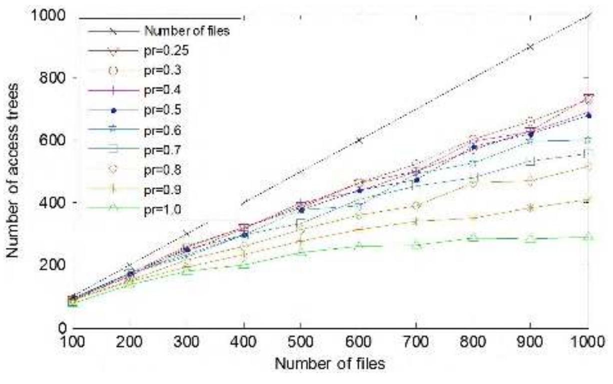
The process of generating the attribute sets for the documents is presented in Algorithm 2. Without loss of generality, we assume that each document has at least 1 attribute and at most 5 attributes. In the initial, we randomly choose the number of a document's attributes from $\{1, 2, \cdots, 5\}$ and then the first attribute A_n is uniformly randomly selected from $\mathcal{A}$. For the documents with more than 1 attribute, the next attribute is chosen by employing a random number $p_r{}'$. If the randomly generated $p_r{}'$ is smaller than p_r, the next attribute is selected in the same category of the first attribute. Otherwise, the next attribute is randomly selected from $\mathcal{A} \backslash A_n$. We iterate the above process until each document is assigned with an attribute set.

Algorithm 2 AttributeDispatcher

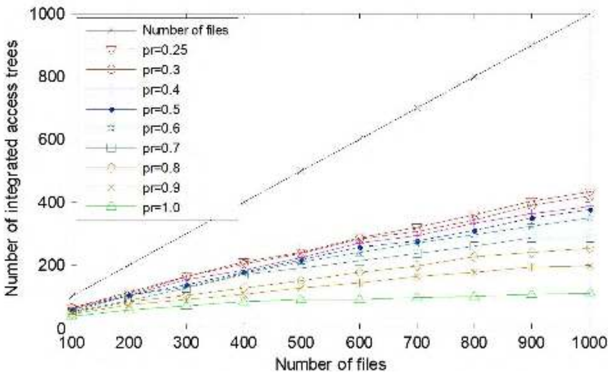
```
Input: $\mathcal{A} = C_1, C_2, C_3, C_4, \mathcal{F}, p_r (0.25 \leq p_r \leq 1)$
Output: The attribute set of each document
1:  for each document $F_i \in \mathcal{F}$ do
2:      $A \neq \emptyset$;
3:      Randomly select a number $m$ from $\{1, 2, 3, 4, 5\}$;
4:      Randomly select an attribute $A_n$ from $\mathcal{A}$ and we
        assume that $A_n \in C_k, k = 1, 2, 3, 4$;
5:      Insert $A_n$ to $\mathcal{A}$;
6:      for $i = 2 : m$ do
7:          Randomly generate a number $p'_r (0 \leq p'_r \leq 1)$ and
            if $p'_r \leq p_r$, randomly select an attribute $A_q$ from
            $C_k \setminus A_n$; otherwise, uniformly randomly select an
            attribute $A_q$ from $\mathcal{A} \setminus A_n$;
8:          Insert $A_q$ to $A$;
9:      end for
10:     The attributes in $A$ comprise the attribute set of docu-
        ment $F_i$;
11: end for
```

B. NUMBER OF INTEGRATED ACCESS TREES

Considering that each integrated access tree is encrypted as a whole, the number of trees in $S_{\mathcal{T}}$ strongly affects the encryption efficiency of CP-ABHE. Based on the assigned attribute sets generated in Section VI-A, we analyze the number of integrated access trees in $S_{\mathcal{T}}$. In KP-ABE and CP-ABE schemes, we assume that the documents with the same attribute set are encrypted together by a same secret key.



(a)



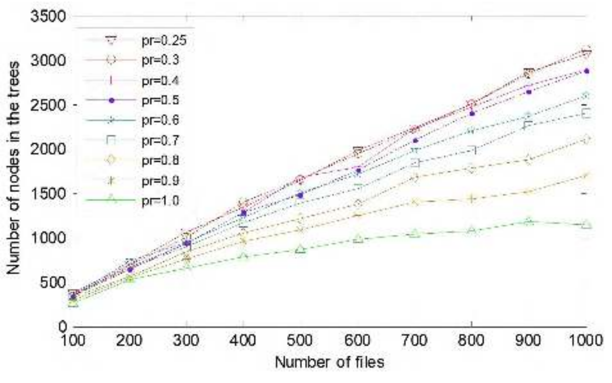
(b)

FIGURE 6. Number of access trees in KP-ABE/CP-ABE schemes. (b) Number of integrated access trees in CP-ABHE scheme.

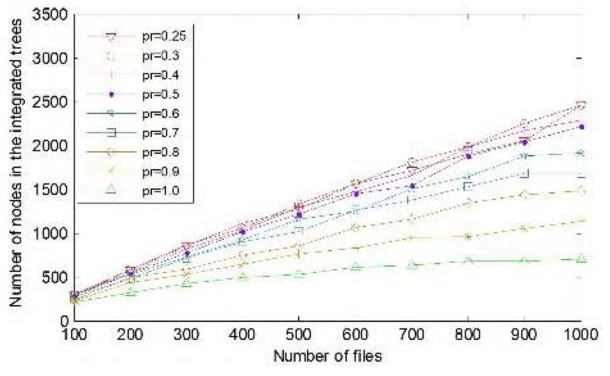
As shown in Fig. 6(a), the number of access trees in KP-ABE/CP-ABE schemes is naturally smaller than that of files. This is reasonable considering that some documents may share a same access tree. With the increasing of the number of files, the number of access trees also increases monotonously though the increasing speed decreases. This can be explained by the fact that with the increasing of the number of access trees, it is increasingly possible that the access tree of a new document is the same with an existing access tree.

The value of p_r also affects the number of access trees. When the attributes of a document are totally randomly selected from $\mathcal{A}$, i.e., $p_r = 0.25$, the attribute sets of the documents are greatly varied. As a consequence, the number of access trees is the largest. With the increasing of p_r, more and more documents share the same access trees and the total number of access trees decreases. For 1000 files, when $p_r = 0.25$, the number of access trees is about 760 and when $p_r = 1.0$, the number of access trees decreases to about 280.

The number of integrated access trees in CP-ABHE is presented in 6(b). Similar to KP-ABE and CP-ABE, the number of integrated access trees in our scheme also gradually increases with the increasing of the number of files and the increasing speed decreases. In addition, the number of integrated access trees decreases with the increasing of p_r.

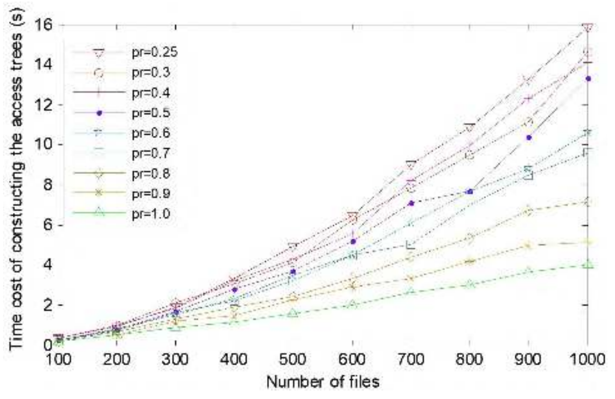


(a)

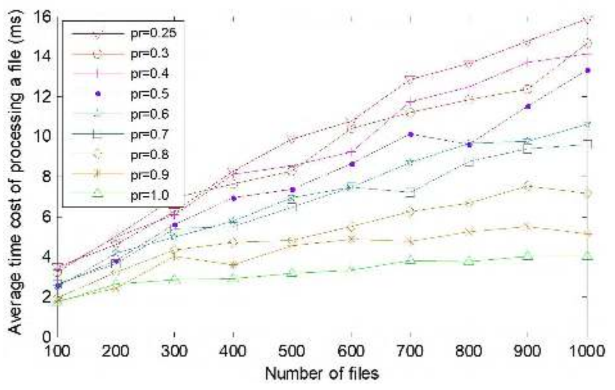


(b)

FIGURE 7. (a) Number of nodes in the access trees in KP-ABE/CP-ABE schemes. (b) Number of nodes in the integrated access trees in CP-ABHE scheme.



(a)



(b)

FIGURE 8. (a) Time cost of constructing the integrated access trees. (b) Average time cost of inserting a file identifier to the integrated access trees.

For 1000 files, when $p_r = 0.25$, the number of the integrated access trees is about 420 and when $p_r = 1.0$, the number of trees decreases to about 110. By comparing 6(a) and 6(b), we can find that the number of integrated access trees is much smaller than that of access trees in KP-ABE and CP-ABE schemes. Therefore, the simulation result illustrates that Algorithm 1 proposed in Section III performs very well in terms of decreasing the number of access trees.

C. NUMBER OF NODES IN THE TREES

As presented in Section IV, each node in the tree needs to be assigned a secret number. Each tree contains a root node, several intermediate nodes and a set of leaf nodes. Roughly speaking, the total number of nodes in the trees approximately linearly increases with the number of access trees and therefore it should have similar relations with the number of files and the value of p_r. As shown in Fig. 7, the number of nodes in KP-ABE and CP-ABE schemes ranges from 1200 to 3200 for different p_r. The number of nodes in the integrated access trees ranges from 700 to 2500. It can be observed that the number of nodes in the integrated trees is always smaller than that of the original access trees. In the process of encrypting the documents, all the schemes need to construct a secret number for each node in the trees. As a consequence, CP-ABHE consumes much fewer computation resources compared with that of KP-ABE and CP-ABE schemes.

D. TIME COST OF TREE CONSTRUCTION

The total time cost of constructing the integrated access trees is presented in Fig. 8(a). Apparently, the time cost increases with the increase of the number of files. For a small p_r, the total time cost increases fast. This can be explained by the fact that when p_r is small, the attribute sets of the files are very different and a large number of access trees need to be scanned before a new file identifier is inserted into the integrated access trees. On the contrary, when p_r is large, quite a number of files share the same integrated access trees and they can be inserted to the trees faster. In the worst case, i.e., $p_r = 0.25$, the time consumption of constructing the integrated access trees for 1000 documents is about 16 seconds. When $p_r = 1$, only about 4 seconds are consumed to construct the access structure of the document collection. The average time cost of inserting a file identifier to the trees is presented in Fig. 8(b).

E. DISTRIBUTION OF THE DOCUMENTS IN THE TREES

In this section, the number of documents is set as 1000. We first sort the trees in descent order based on the number of file identifiers stored in the trees. Then, we divide the trees into different sets and each set contains 25 trees. At last, we count the number of file identifiers stored in each set of

TABLE 1. Comparison of KP-ABE, CP-ABE and CP-ABHE.

Component	KP-ABE [11]	CP-ABE [1]	CP-ABHE
Encryption Time	$(\lvert att(F_1)\rvert + \cdots + \lvert att(F_N)\rvert)\mathbb{G}_0 + 2N\mathbb{G}_1$	$[2(\lvert att(F_1)\rvert + \cdots + \lvert att(F_N)\rvert) + N]\mathbb{G}_0 + 2N\mathbb{G}_1$	$(\sum_{\mathcal{T}\in S_{\mathcal{T}}} \lvert\mathcal{T}\rvert + 2\lvert\mathcal{A}\rvert)\mathbb{G}_0 + 2N\mathbb{G}_1$
Decryption Time	$[2(\lvert att(F_1)\rvert + \cdots + \lvert att(F_N)\rvert) + N]\mathbb{G}_1$ $+(\lvert att(F_1)\rvert + \cdots + \lvert att(F_N)\rvert)C_e$	$[3(\lvert att(F_1)\rvert + \cdots + \lvert att(F_N)\rvert) + 2N]\mathbb{G}_1$ $+[2(\lvert att(F_1)\rvert + \cdots + \lvert att(F_N)\rvert) + N]C_e$	$[2(\sum_{\mathcal{T}\in S_{\mathcal{T}}} \lvert att(\mathcal{T})\rvert + \lvert A\rvert + 2N]\mathbb{G}_1$ $+(2\lvert\mathcal{A}\rvert + N)C_e$
The Size of PK	$\lvert\mathcal{A}\rvert L_{\mathbb{G}_0} + L_{\mathbb{G}_1}$	$3L_{\mathbb{G}_0} + L_{\mathbb{G}_1}$	$3L_{\mathbb{G}_0} + L_{\mathbb{G}_1}$
The Size of MSK	$(\lvert\mathcal{A}\rvert + 1)L_{\mathbb{Z}_p}$	$L_{\mathbb{Z}_p} + L_{\mathbb{G}_0}$	$L_{\mathbb{Z}_p} + L_{\mathbb{G}_0}$
The Size of SK	$(\lvert att(F_1)\rvert + \cdots + \lvert att(F_N)\rvert)L_{\mathbb{G}_0}$	$(2\lvert\mathcal{A}\rvert + 1)L_{\mathbb{G}_0}$	$(2\lvert\mathcal{A}\rvert + 1)L_{\mathbb{G}_0}$
The Size of CT	$(\lvert att(F_1)\rvert + \cdots + \lvert att(F_N)\rvert)L_{\mathbb{G}_0} + NL_{\mathbb{G}_1}$	$[2(\lvert att(F_1)\rvert + \cdots + \lvert att(F_N)\rvert) + N]L_{\mathbb{G}_0} + NL_{\mathbb{G}_1}$	$(2\lvert\mathcal{A}\rvert + \sum_{\mathcal{T}\in S_{\mathcal{T}}} \lvert\mathcal{T}\rvert)L_{\mathbb{G}_0} + NL_{\mathbb{G}_1}$

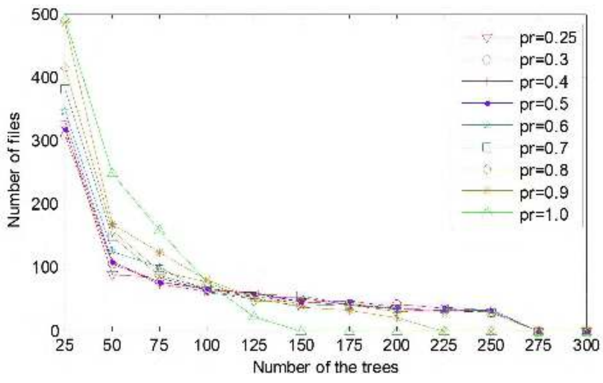


FIGURE 9. File distribution in the access trees.

the trees. As shown in Fig. 9, quite a number of file identifiers are contained in the first 50 integrated access trees and the proportion ranges from 40% to 75% for different p_r. Then there exists a long tail for the rest of the trees. In addition, a larger p_r leads a shorter tail and when $p_r = 1.0$, almost all the files are stored in the first 150 integrated access trees. This is reasonable considering that more and more files share the same attribute sets when the value of p_r increases. Though the number of the trees are larger than 200 when p_r ranges from 0.25 to 0.8 as shown in Fig. 6(a), almost all the document identifiers are stored in the largest 200 trees. If ignoring a small number of documents is acceptable for some applications, the number of the trees greatly decreases and hence the efficiency of the schemes can be further improved.

VII. EFFICIENCY OF CP-ABHE

A. PERFORMANCE ANALYSIS

We theoretically compare the proposed scheme with KP-ABE and CP-ABE schemes in terms of encryption/decryption efficiency and storage space. For convenience, some basic definitions are presented first. We assume that $\mathbb{G}_i (i = 0, 1)$ is a group or the time cost of a basic operation on the group such as exponentiation or multiplication. Let $\mathbb{Z}_p$ be the group $\{0, 1, \cdots, p-1\}$ and C_e be the time cost of a bilinear map operation e. In addition, we define $|*|$ as the number of elements in $*$, L_* as the length of an element in $*$.

We assume that the data owner encrypts the content keys $ck = \{ck_1, ck_2, \cdots, ck_N\}$ by KP-ABE, CP-ABE and our scheme, respectively. Let $att(F_i)$ returns the attribute set of F_i, $att(\mathcal{T})$ be the attribute set of the access tree $\mathcal{T}$. We denote the number of nodes which contain at least one document identifiers in $\mathcal{T}$ as $|\mathcal{T}|$. The time consumptions of constructing the

TABLE 2. The rank of the three schemes' performance.

Component	KP-ABE	CP-ABE	CP-ABHE
Encryption Time	2nd	3rd	1st
Decryption Time	2nd	3rd	1st
The Size of PK	2nd	1st	1st
The Size of MSK	2nd	1st	1st
The Size of SK	2nd	1st	1st
The Size of CT	2nd	3rd	1st
Flexibility	2nd	1st	1st

polynomials when generating the secret numbers are ignored. We further assume that a data user needs to decrypt all the documents. Under the above assumptions, a theoretical comparison between these three schemes is presented in Table 1.

By basic analysis, we can infer that $|att(F_1)| + \cdots + |att(F_N)| \gg \{N, \sum_{\mathcal{T}\in S_{\mathcal{T}}} |\mathcal{T}|, \sum_{\mathcal{T}\in S_{\mathcal{T}}} |att(\mathcal{T})|\} \gg |\mathcal{A}|$ for a large document collection. Then, we can rank the performance of these three schemes in terms of different measurements as shown in Table 2. It can be observed that the CP-ABHE performs the best in terms of all the measurements. The KP-ABE scheme performs better than CP-ABE in terms of encryption/decryption efficiency and the size of CT. However, a huge disadvantage of the KP-ABE scheme is the **secret key expanding problem**. The CP-ABE scheme performs better than KP-ABE in terms of the size of PK, MSK, and SK. However, the size of the ciphertext is much larger than that of KP-ABE scheme. When sending the ciphertext to the data users, the data transmission amount in CP-ABE is much larger and it is a challenge for the networks. In addition, CP-ABE scheme and CP-ABHE scheme are more flexible than the KP-ABE scheme in real life. In conclusion, theoretical analysis shows that both KP-ABE and CP-ABE have their disadvantages and CP-ABHE always performs the best.

B. PERFORMANCE EVALUATION

To further evaluate the performance of these three document encryption schemes, we implement the CP-ABHE scheme based on the cpabe toolkit and the Java Pairing-Based Cryptography library (JPBC) [4]. We employ a 160-bit elliptic curve group based on the supersingular curve $y^2 = x^3 + x$ over a 512-bit finite field. In addition, the KP-ABE scheme in [11] and the CP-ABE scheme in [1] are also implemented. All the above schemes are simulated on a 2.60 GHZ Intel Core processor, Windows 7 operating system with a RAM of 4 GB. The number of documents in the collection ranges from 100 to 1000. As presented in Section VI, the attribute dictionary is defined as $\mathcal{A} = \{A, B, \cdots, Z\}$ and each

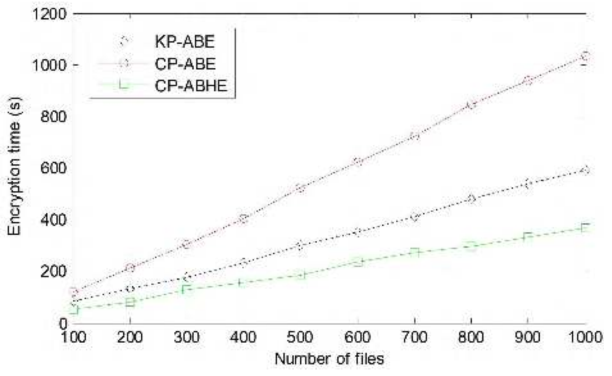


FIGURE 10. Encryption time.

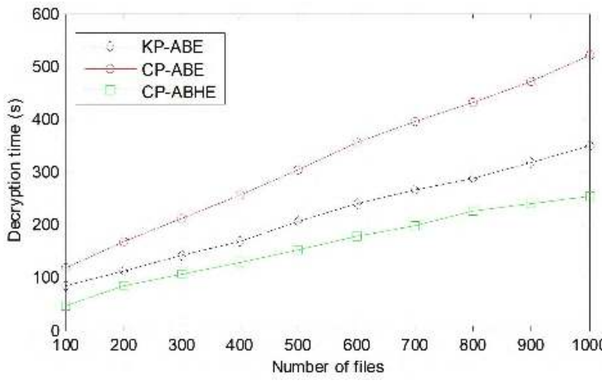


FIGURE 11. Decryption time.

document is assigned with an attribute set through Algorithm 2. Similar to [13] and [33], the encryption/decryption time and the storage cost of ciphertext are employed to measure the performance of these schemes. In addition, we also use the secret key storage space to reflect the **secret key expanding problem**. Each simulation is executed for 10 times and the average simulation results are presented in the following.

1) ENCRYPTION EFFICIENCY

The encryption time of the three schemes with different number of documents is presented in Fig. 10. To obtain $ck_ie(g, g)^{\alpha s}$ of a document, CP-ABE needs to execute two operations on $\mathbb{G}_1$. In addition, the scheme needs to execute $2 * att(F_i) + 1$ operations on $\mathbb{G}_0$ to get h^s, C_j and C'_j. When a new document arrives, all the encryption process needs to be re-executed for one time. Therefore, the encryption time of CP-ABE is the largest in all the three schemes. The KP-ABE scheme also needs to encrypt each document singly. However, the KP-ABE scheme needs to execute only $att(F_i)$ operations in $\mathbb{G}_0$ and hence it performs better than CP-ABE in terms of encryption time. In CP-ABHE, all the ciphertexts of the documents share the same C_y and C'_y which can greatly decrease the computation complexity. As shown in Fig. 10, CP-ABHE improves the encryption efficiency by about 60% compared with CP-ABE and it also outperforms KP-ABE scheme.

2) DECYPTION EFFICIENCY

As shown in Fig. 11, the decryption time of all the three schemes approximately linearly increases with the expanding of the document collection. For a constant document collection, CP-ABHE improves the decryption efficiency by about 50% compared with the CP-ABE scheme and it also outperforms KP-ABE. To decrypt all the encrypted documents, the KP-ABE scheme and CP-ABE scheme need to decrypt the access trees one by one. For each access tree, they first need to decrypt the leaf nodes and then decrypt the root node by an iterating process. At last, the content secret key ck_i hidden in the access tree is decrypted. It can be observed that most time is consumed in the process of decrypting the nodes in the trees. For different access trees, the secret numbers of in the leaf nodes are independent with each other. To decrypt a leaf node, the CP-ABE scheme needs to execute bilinear map operation two times and a operation in $\mathbb{G}_1$. However, in the KP-ABE scheme, only a bilinear map is needed to decrypt a leaf node. Considering that the rest decryption process of KP-ABE and CP-ABE schemes are similar to each other, we can conclude that the KP-ABE scheme outperforms CP-ABE scheme in terms of decryption efficiency. CP-ABHE performs the best in all the three schemes and this can be explained by the fact that only M (i.e., $|\mathcal{A}|$) leaf nodes need to be decrypted.

3) CIPHERTEXT STORAGE EFFICIENCY

In this section, we restrict our attention to the ciphertext of the encrypted content keys. As shown in Fig. 12, the ciphertext in CP-ABE scheme consumes the most storage space. For each access tree, the ciphertext includes an element in $\mathbb{G}_1$ (i.e., $ck_ie(g, g)^{\alpha s}$) and $2 * |att(F_i)| + 1$ elements in $\mathbb{G}_0$ (i.e., C_j, C'_j and $C = h^s$). For different access trees, their ciphertexts are totally independent and we cannot save any storage cost by publishing the ciphertexts together. Though the ciphertexts of different access trees in the KP-ABE scheme are also independent with each other, the ciphertext of an access tree includes only an element in $\mathbb{G}_1$ (i.e., $ck_ie(g, g)^{\alpha s}$) and $|att(F_i)|$ elements in $\mathbb{G}_0$ (i.e., E_j). Therefore, the size of ciphertext in the KP-ABE scheme is much smaller than that of the ciphertext in the CP-ABE scheme. CP-ABHE scheme performs the best in all the three schemes and consumes the least storage space. In CP-ABHE, the content keys are encrypted together and the ciphertext of the keys includes N elements in $\mathbb{G}_1$ and $2\,|\mathcal{A}| + \sum_{\mathcal{T} \in S_{\mathcal{T}}} |\mathcal{T}|$ elements in $\mathbb{G}_0$. As discussed in section VII-A, the ciphertext of CP-ABHE consumes much smaller storage space compared with that of KP-ABE scheme and CP-ABE scheme. The simulation result demonstrates the correctness of our theoretical analysis.

4) SECRET KEY STORAGE EFFICIENCY

The total storage cost of the secret keys is presented in Fig. 13. In CP-ABE and CP-ABHE schemes, the secret key of a data user is related with his attribute set only and doesn't expand with the increasing of the document collection.

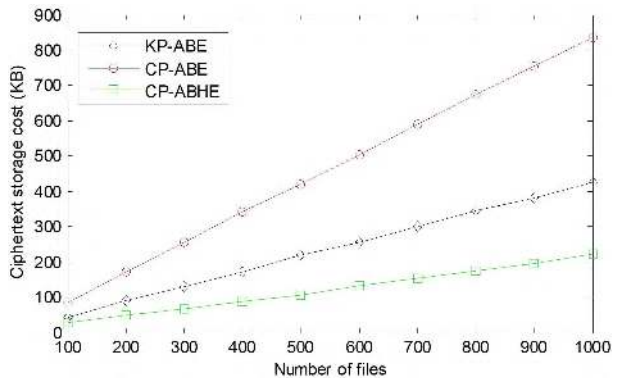


FIGURE 12. Ciphertext storage space.

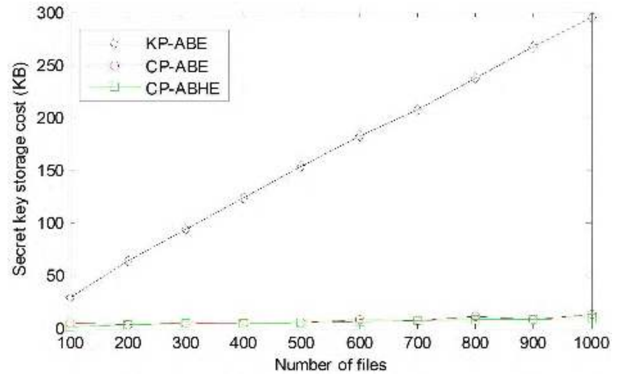


FIGURE 13. Secret key storage space.

However, in the KP-ABE scheme, each secret key is generated for a specific access tree. With the increasing of the document collection's size, the number of access trees increases and, as a consequence, the size of a data user's secret key is extremely large. It can be observed from Fig. 13 that the storage space of the secret key of a data user is linearly increasing with the document collection's size. When the document collection contains 1000 documents, the size of the secret key in KP-ABE is about 300 KB which is much larger than that of the secret keys in CP-ABE and CP-ABHE schemes.

C. PERFORMANCE COMPARISON

In conclusion, CP-ABHE scheme always performs the best in terms of encryption/decryption time, the secret key storage cost and ciphertext storage cost. KP-ABE scheme performs better than the CP-ABE scheme in terms of encryption/decryption time and cipher-text storage cost. However, a huge disadvantage of the KP-ABE scheme is the **secret key expanding problem**. As we step into the era of mobile internet, more and more data users tend to access the documents through mobile devices which are of very limited resources. In this case, storing a large number of secret keys is impractical. Though CP-ABE scheme has a larger cost in terms of encryption/decryption and ciphertext storage, it is more convenient for the data owners to set the access structures and the data users need to store a small number of secret keys.

VIII. RELATED WORK

Attribute-based encryption schemes have been widely researched in the literatures. The fuzzy identity-based encryption (Fuzzy IBE) scheme proposed by Sahai and Waters [28] is widely treated as the origin of attribute-based encryption (ABE). Sahai and Waters first employ the term "attribute-based encryption (ABE)" in the field of information security. Inspired by Fuzzy IBE, many ABE schemes are designed including KP-ABE schemes and CP-ABE schemes. Goyal *et al.* extend the Fuzzy IBE scheme and propose the key-policy attribute-based-encryption (KP-ABE) in [11]. Though KP-ABE can provide fine-grained access control, it restricts its attention to the monotone access structure only. In [25], Ostrovsky *et al.* construct a KP-ABE scheme which allows a user's private key can be expressed in terms of any access formula over attributes. Further, they prove the scheme's security based on decisional bilinear Diffie-Hellman assumption. Yang *et al.* [38] propose a scheme which performs well in terms of both access structure expressivity and security. CP-ABE schemes are more flexible and suitable for general applications and many varieties of CP-ABE schemes have been proposed in the literatures [1], [10], [34]. In CP-ABE schemes, the access structures are embedded in the ciphertext and each data user is assigned with a set of attributes. A data user can decrypt a ciphertext if and only if their be matched with each other.

Recently, ABE schemes have been widely employed to securely store and share data in cloud computing. Pirretti *et al.* [26] introduce a novel secure information management architecture based on ABE primitives. A policy system which meets the needs of different data users is designed and used to encrypt distributed file systems. The hierarchical ABE (HABE) scheme [32] is proposed by combining a hierarchical IBE scheme and a CP-ABE scheme. HABE scheme can help the enterprise users to efficiently share confidential data in cloud computing by simultaneously achieving fine-grained access control, high performance, practicability, and scalability. Zhu *et al.* [39] also propose a file sharing scheme in cloud computing based on ABE and the security and efficiency of the scheme are evaluated. Li *et al.* [17] provide a CP-ABE scheme with efficient data user revocation for cloud storage. KSF-OABE scheme [18] integrates the keyword search function into the ABE scheme which can improve the search efficiency of ciphertexts. Though all the above proposed schemes can be used in cloud computing, they are designed for encrypting a single document. They cannot be directly employed to encrypt a large document collection, because the encryption/decryption efficiency is low if we encrypt each file singly. To our knowledge, the most related work to our scheme is FH-CP-ABE [33] and however, this scheme can only hierarchically encrypt a set of documents together whose attribute sets need to nicely comprise an integrated access structure. This is impractical for a large document collection considering that the attribute sets of the documents are random.

IX. CONCLUSION

In this paper, we design a hierarchical document collection encryption scheme. We first design an incremental algorithm to construct the integrated access trees of the documents and decrease the number of trees. Then, each integrated access tree is encrypted together and the documents in a tree can be decrypted at a time. Different to existing schemes, we construct the secret numbers for the nodes of the trees in a bottom-up manner. In this way, the sizes of ciphertext and secret keys significantly decrease. At last, a thorough performance evaluation is provided including security analysis, efficiency analysis, and simulation. Results show that the proposed scheme outperforms KP-ABE and CP-ABE schemes in terms of encryption/decryption efficiency and storage space.

Our scheme can be further improved in several aspects: First, the access policy discussed in Section III assumes that the access trees are composed of only "AND" gates. Extending the flexibility and versatility of the access policy is one of the most important research directions. Second, the documents are encrypted before outsourcing and a promising task is how to efficiently search the interested documents over the ciphertexts. At last, we focus our attention on the static document collection and how to efficiently encrypt/decrypt a dynamic document collection will be also researched in the future.

REFERENCES

[1] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy*, May 2007, pp. 321–334.

[2] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. TCC*, 2007, pp. 535–554.

[3] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2014.

[4] A. D. Caro and V. Iovino, "jPBC: Java pairing based cryptography," in *Proc. IEEE Symp. Comput. Commun. IEEE Comput. Soc.*, Jun. 2011, pp. 850–855.

[5] C. Chen *et al.*, "An efficient privacy-preserving ranked keyword search method," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 4, pp. 951–963, Apr. 2016.

[6] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. ACM CCS*, 2006, pp. 79–88.

[7] H. Deng *et al.*, "Ciphertext-policy hierarchical attribute-based encryption with short ciphertexts," *Inf. Sci.*, vol. 275, pp. 370–384, Aug. 2014.

[8] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2546–2559, Sep. 2016.

[9] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Proc. ACNS*, 2004, pp. 31–45.

[10] V. Goyal, A. Jain, O. Pandey, and A. Sahai, "Bounded ciphertext policy attribute based encryption," in *Automata, Languages and Programming*. Berlin, Germany: Springer, 2008, pp. 579–591.

[11] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 89–98.

[12] J. Han, W. Susilo, Y. Mu, and J. Yan, "Privacy-preserving decentralized key-policy attribute-based encryption," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 11, pp. 2150–2162, Nov. 2012.

[13] J. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 8, pp. 1343–1354, Aug. 2013.

[14] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Proc. EUROCRYPT*, 2010, pp. 62–91.

[15] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer-Verlag, 2010, pp. 62–91.

[16] J. Li, C. Jia, J. Li, and X. Chen, "Outsourcing encryption of attribute-based encryption with MapReduce," *Proc. Int. Conf. Inf. Commun. Secur.* Berlin, Germany: Springer-Verlag, 2012, pp. 191–201.

[17] J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 785–796, Sep./Oct. 2017.

[18] X. N. Lin *et al.*, "Outsourced attribute-based encryption with keyword search function for cloud storage," *IEEE Trans. Services Comput.*, to be published.

[19] L. Cheung and C. Newport, "Provably secure ciphertext policy ABE," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2007, pp. 456–465.

[20] W. Liu, J. Liu, Q. Wu, B. Qin, and Y. Zhou, "Practical direct chosen ciphertext secure key-policy attribute-based encryption with public ciphertext test," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2014, pp. 91–108.

[21] Z. Liu, Z. Cao, and D. S. Wong, "Traceable CP-ABE: How to trace decryption devices found in the wild," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 1, pp. 55–68, Jan. 2015.

[22] Z. Liu, Z. Cao, and D. S. Wong, "White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 1, pp. 76–88, Jan. 2013.

[23] J. Ning, Z. Cao, X. Dong, L. Wei, and X. Lin, "Large universe ciphertext-policy attribute-based encryption with white-box traceability," in *Proc. Eur. Symp. Res. Comput. Secur.*, vol. 8713, 2014, pp. 55–72.

[24] T. Okamoto and K. Takashima, "Fully secure functional encryption with general relations from the decisional linear assumption," in *Proc. CRYPTO*. Berlin, Germany: Springer, 2010, pp. 191–208.

[25] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 195–203.

[26] M. Pirretti, P. Traynor, P. McDaniel, "Secure attribute-based systems," *J. Comput. Secur.*, vol. 18, no. 5, pp. 799–837, 2010.

[27] H. Qian, J. Li, and Y. Zhang, "Privacy-preserving decentralized ciphertext-policy attribute-based encryption with fully hidden access structure," in *Information and Communications Security*. Cham, Switzerland: Springer, 2013, pp. 363–372.

[28] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. EUROCRYPT*, 2005, pp. 457–473.

[29] S. Subashini and V. Kavitha, "A survey on security issues in service delivery models of cloud computing," *J. Netw. Comput. Appl.*, vol. 34, no. 1, pp. 1–11, 2011.

[30] A. Swaminathan *et al.*, "Confidentiality-preserving rank-ordered search," in *Proc. ACM Workshop Storage Secur. Survivability*, 2007, pp. 7–12.

[31] C. Wang, N. Cao, K. Ren, and W. Lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1467–1479, Aug. 2012.

[32] G. Wang, Q. Liu, and J. Wu, "Hierarchical attribute-based encryption for fine-grained access control in cloud storage services," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2010, pp. 735–737.

[33] S. Wang, J. Zhou, J. K. Liu, J. Yu, J. Chen, and W. Xie, "An efficient file hierarchy attribute-based encryption scheme in cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1265–1277, Jun. 2016.

[34] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography—PKC*. Berlin, Germany: Springer, 2011, pp. 53–70.

[35] W. K. Wong, D. W. Cheung, B. Kao, and N. Mamoulis, "Secure kNN computation on encrypted databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2009, pp. 139–152.

[36] F. Xhafa, J. Wang, X. Chen, J. K. Liu, J. Li, and P. Krause, "An efficient PHR service system supporting fuzzy keyword search and fine-grained access control," *Soft Comput.*, vol. 18, no. 9, pp. 1795–1802, 2014.

[37] Z. Xia, X. Wang, X. Sun, and Q. Wang, "A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 2, pp. 340–352, Feb. 2016.

[38] X. Yang, W. Du, X. Wang, and L. Wang, "Fully secure attribute-based encryption with non-monotonic access structures," in *Proc. IEEE Int. Conf. Intell. Netw. Collaborative Syst.*, Sep. 2013, pp. 521–527.

[39] S. Zhu, X. Yang, and X. Wu, "Secure cloud file system with attribute based encryption," *Proc. IEEE Int. Conf. Intell. Netw. Collaborative Syst. Comput. Soc.*, Sep. 2013, pp. 99–102.

JUNSONG FU received the Ph.D. degree in communication and information system from Beijing Jiaotong University, in 2018. He is currently an Assistant Professor with the School of Cyberspace Security, Beijing University of Posts and Telecommunications. His research interests include network data processing, network security, and information privacy issues in distributed systems and the Internet of Things.

NA WANG received the Ph.D. degree from the School of Mathematical Sciences, Xiamen University, in 2018. She is currently a Postdoctoral Fellow with the School of computer Science, Beijing University of Posts and Telecommunications. Her research interests include cryptography, message sharing, and information security issues in distributed and cloud systems.