Journal of Parallel and Distributed Computing 135 (2020) 56–69

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

A fine-grained authorized keyword secure search scheme with efficient search permission update in cloud computing

Hui Yin [a], Zheng Qin [b,*], Jixin Zhang [b], Hua Deng [b], Fangmin Li [a], Keqin Li [c]

[a] *College of Computer Engineering and Applied Mathematics, Changsha University, Changsha, Hunan, 410022, China*
[b] *College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan, 410082, China*
[c] *Department of Computer Science, State University of New York, New Paltz, NY, 12561, USA*

ARTICLE INFO

Article history:
Received 22 April 2019
Received in revised form 27 August 2019
Accepted 17 September 2019
Available online 23 September 2019

Keywords:
Authorized keyword search
Attribute-based encryption
Cloud computing
Search permission update
Secure search

ABSTRACT

With the rapid development of cloud computing, secure search has become a hot research spot, which is a promising technique that allows a data user to perform privacy-preserving keyword-based search over encrypted cloud data. In this paper, we further consider the secure search problem based on a practical application scenario that a data owner needs to grant different keyword query permissions for different data users to achieve flexible access control on outsourced encrypted data in the cloud computing environment. To address this problem, we propose a fine-grained authorized keyword secure search scheme by leveraging the ciphertext policy attribute-based encryption (ABE), which not only supports privacy-preserving keyword-based search over encrypted data, but also inherits flexible and fine-grained data privilege control properties of ABE. Moreover, our proposed scheme is able to achieve fine-grained search permission update with very small communication and computation cost. By running the attribute revocation sub-protocol and attribute addition sub-protocol, the data owner can flexibly and efficiently update a data user's keyword search permissions when the data user's system role changes. We provide detailed performance analysis and rigorous security proof for our scheme. Extensive experiments demonstrate the correctness and practicality of the proposed scheme.

1. Introduction

1.1. Motivation

Cloud computing is an emerging and promising information technology and service paradigm by providing for its customers low-cost and elastic resources, convenient data sharing and accesses, on-demand high quality application services, and the powerful parallel processing abilities with minimal management effort [37]. Compared with the traditional information technology, abundant advantages brought by the cloud computing have motivated more and more individual users and enterprises to migrate their data and computation task to the cloud server.

Data outsourcing is the most attractive and popular application scheme in the cloud computing. Cloud customs usually wish to outsource their data to the cloud server to enjoy scalable and flexible data storage and management services with minimal capital outlays and management overhead [7]. However, once data is uploaded to cloud servers, data owners also correspondingly lose the direct physical control on their data since the cloud servers are operated by remote commercial organizations, which are far away from data owners [29]. Consequentially, the outsourced data may face the dual-threat from the cloud server and external attackers. Therefore, the data security problem is the most important factor considered for the data owners to determine whether to outsource their private data to the cloud server. Researchers propose that the data encryption is the simplest and most effective way to guarantee the security of outsourced cloud data [17,28]. However, data encryption makes the conventional plaintext keyword based query techniques unavailable [29].

Searchable encryption provides a mechanism that allows a user to perform encrypted keyword query over encrypted data stored at a remote server. The first practical searchable encryption scheme was proposed by Song et al. in [32]. The continual improvements in [1,3,4,8,10,13,14,18,19,21] make the searchable encryption more efficient, securer, and more flexible to support query results verification and data dynamic update in the ciphertext environment. Recently, with the rapid prevalence of cloud computing, researchers begin to explore efficient and feature-rich secure search techniques over outsourced encrypted cloud data such as secure ranked multi-keyword search, secure fuzzy keyword search, secure similarity search, and secure personalized search. To achieve these goals, some excellent schemes have been proposed in [5,6,12,22,23,33,34,38,39,42–45]. However, all above schemes assume that an authorized data user has sufficient

* Corresponding author.
E-mail address: zqin@hnu.edu.cn (Z. Qin).

https://doi.org/10.1016/j.jpdc.2019.09.011
0743-7315/© 2019 Elsevier Inc. All rights reserved.

access to search all outsourced data files. In other words, the data owner allows a data user to use arbitrary keywords to obtain all eligible data files without enforcing any data access control on outsourced data files. In some real-world application scenarios, the data owner (e.g., data executives of an enterprise) may desire to grant different keyword query permissions for different data users in a fine-grained manner to flexibly and strictly determine which users have the access privilege to which data files [15]. For example, a technical manager of a company may be allowed to search technique documents of all products while an ordinary programmer can only obtain the development documents related to his current projects yet both of them are prohibited from accessing financial statements of the company. Carrying out thus a flexible and fine-grained data access control is not a trivial work for the data owner, since the data is outsourced to the remote cloud server in the encryption form in the system. Moreover, the dynamics of data users make the flexible data access authorization more challenging. For example, a programmer can be promoted to the technical manager. Once the role of the data user changes, the data owner should update the data user's data access privilege immediately.

1.2. Our contributions

In this paper, we mainly make three key contributions, which can be summarized as follows.

1. We present a fine-grained authorized keyword secure search scheme with efficient search permission update by evolving the popular ciphertext policy attribute-based encryption (CP-ABE) scheme proposed in [2]. Therefore, our proposed scheme can be regarded as a new primitive called *searchable ciphertext policy attribute-based encryption with search permission update (SCP-ABE-SPU)*.
2. We propose an efficient and fine-grained search permission update protocol, which includes two sub-protocols: *attribute revocation sub-protocol* and *attribute addition sub-protocol*. By running attribute update protocol, the data owner is able to flexibly update a data user's keyword query permissions in a fine-grained manner when the data user's system role changes.
3. We make detailed performance analyses for our proposed authorized keyword search scheme and search permission update protocol. The formal security proof and thorough security analysis are provided. Extensive experimental results on a real data set demonstrate the correctness and practicality of the proposed scheme.

The remainder of this paper is organized as follows. We review the related work in Section 2. The related background, including system model, threat model, and several basic techniques to be used in the paper, is described in Section 3. We formally construct the authorized keyword secure search scheme and search permission update protocol in Sections 4 and 5, respectively. Thorough performance and security analyses are provided in Sections 6 and 7, respectively. We make experimental evaluation for our scheme in Section 8 and conclude the paper in Section 9.

2. Related work

2.1. Secure search over encrypted cloud data

Wang et al. [38] first used encrypted keyword relevance score as ranking criterion to implement single keyword top-*k* secure search over encrypted cloud data. They extends their conference version by putting on ranked search result authentication mechanism in [39]. To achieve multi-keyword ranked search over encrypted cloud data, Cao et al. [5] used encrypted space vector model to design a "coordinate matching"-based multi-keyword ranked secure search scheme. They improved the ranked accuracy in [6] by introducing TF×IDF rule. Sun et al. [34] proposed to use the tree-based index structure to improve multi-keyword search efficiency and use cosine similarity measure to achieve higher search result ranking accuracy under the space vector model. Sun et al. [33] extended their scheme [34] to make it enable authenticity check over the returned search results. Similar to Sun et al.'s works [34] and [33], Xia et al. [43] also constructed tree-based index under the space vector and TF×IDF model to achieve efficient and accurate multi-keyword ranked search. An important difference from [34] and [33] is that Xia et al.'s scheme is able to deal with the deletion and insertion of data files flexibly and efficiently. Based on the same technical route as above multi-keyword schemes, Li et al. [23] proposed a very efficient multi-keyword search scheme over encrypted cloud data by supporting classified sub-dictionary computation. By building a search interest model according to a data user's search history, Fu et al. [12] proposed a personalized multi-keyword secure search scheme to improve user search experience. To tolerate minor typos, fuzzy keyword search schemes over encrypted cloud data were also proposed in [22] and [42]. Yin et al. fully considered the data security and search privacy in multiple data owners scenario and proposed a secure conjunctive multi-keyword ranked search scheme over encrypted cloud data in [45].

2.2. Authorized keyword secure search based on attribute encryption

Zheng et al.'s work designed a novel cryptographic primitive to achieve fine-grained query authorization by exploiting the attribute-based encryption (ABE) [30], called attribute-based keyword search (ABKS) [47] and Sun et al. exploited ABE to realize fine-grained owner-enforced search authorization in the multiple data owners model [35]. Later, several authorized keyword secure search schemes are proposed in [9,24–27,31,40,41,46]. However, each of these schemes has respective limitations. Specifically, the schemes [35] and [40] used the access structure that only allows AND policies and schemes [9,24,27,41] considered AND and OR gates of attributes but do not support threshold gate. The authors proposed an efficient ciphertext-policy searchable attribute-based encryption scheme in [46], which supports AND, OR, and threshold gate. However, the trapdoor cannot achieve unlinkability security property due to the deterministic encryption. In [26], miao et al. proposed the attribute-based keyword search over hierarchical data with tree access structure supporting AND, OR, and threshold gate, but disenabling the search permission update. Schemes [25] and [31] use composite-order group to set up pairing environment, which incurs the impractical search complexity [9]. In addition, we emphasize that none of the above schemes achieve fine-grained search permission update. In some application scenarios, flexibly updating a data user's keyword search permission may be more practical and attractive.

3. Background

3.1. System model

As shown in Fig. 1, there are three entities in our system model. They are the data owner, the cloud server, and data users. To guarantee data confidentiality, before outsourcing data files, the data owner encrypts data files as well as builds secure searchable indexes for enabling the efficient search over encrypted data. Each secure searchable index is embedded an access control policy by the data owner, which defines what type of users can search on this index. An authorization data user with a set of attributes submits a query trapdoor (i.e., encrypted query keyword) to cloud server to obtain the interested data files. Upon receiving

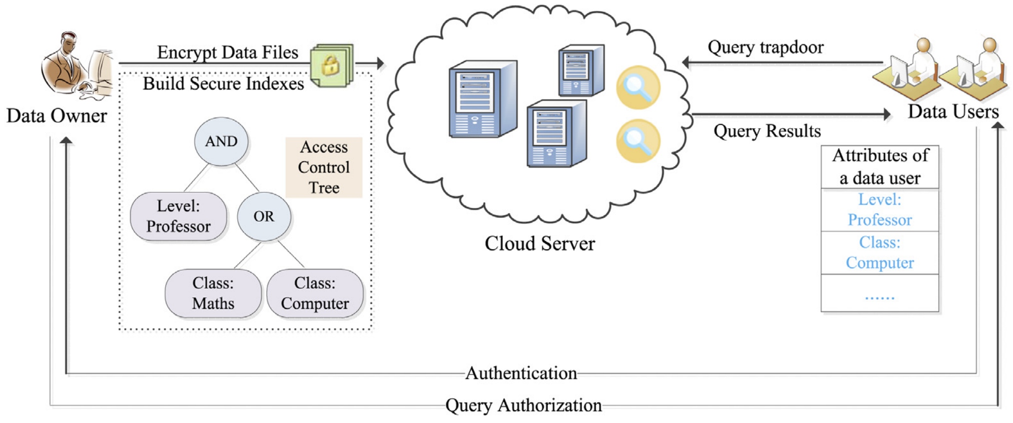


Fig. 1. A system model of search over encrypted data on a remote cloud center.

the query trapdoor, the cloud server is responsible for performing search over encrypted outsourced data. Specifically, only when the data user's attribute set satisfies the access control policy embedded in an index, the cloud server can perform correct query over the secure index. Finally, the cloud server returns back the encrypted query results to the data user.

In addition, assume that secure channels exist between the data owner and data users, by which the system authentication can be conducted correctly and secretly. After passing the system authentication, the data user obtains some important components from the data owner such as data file decryption key and query permissions via existed secure channels.

3.2. Threat model

In this paper, the data owner and data users are the trusted entities. Similar to previous related works, we consider the cloud server as "honest-but-curious" in the sense that the cloud server could strictly obey data escrow protocols and correctly fulfill the functional responsibilities while it may try to obtain as many contents as possible from encrypted data files, secure searchable index, and query trapdoor. We also assume that the cloud server cannot launch collusion attacks with the data owner and data users.

In general, the security mainly captures the notion that the cloud server cannot obtain any underlying plaintext information from outsourced data files, searchable index, and query trapdoors. Now, we first give the formal security definitions through the following games between a challenger $\mathcal{B}$ and the adversary $\mathcal{A}$.

Adaptively Chosen-Keyword Attack Game:

Setup. The challenger $\mathcal{B}$ initializes running environment and sends public parameters to the adversary $\mathcal{A}$.

Phase 1. $\mathcal{A}$ adaptively requests search trapdoor $\mathcal{T}_{\mathcal{A}}(w)$ for any keyword w for polynomially many times from $\mathcal{B}$ with the attribute sets $S_1, \ldots, S_q$.

Challenge. $\mathcal{A}$ define a challenge access tree T^* such that none of the attribute sets $S_1, \ldots, S_q$ from Phase 1 satisfy T^*. $\mathcal{A}$ submits two keywords w_0, w_1 and T^* to $\mathcal{B}$. $\mathcal{B}$ flips a random binary coin $b \in \{0, 1\}$ and encrypts w_b with T^* as $\mathcal{I}_{w_b}$, which is sent to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ continues to query the search trapdoor $\mathcal{T}_{\mathcal{A}}(w)$ for chosen keyword w (including w_0 and w_1) with the attribute set S_{q_w} from $\mathcal{B}$. The only restriction is that if the attribute set S_{q_w} in $\mathcal{T}_{\mathcal{A}}(w)$ satisfies T^*, then $w \neq w_0, w_1$ (in other word, if $w = w_0$ or $w = w_1$, then $S_{q_{w_0}}$ or $S_{q_{w_1}}$ does not satisfy T^*).

Guess. Finally, $\mathcal{A}$ outputs a guess b' of b.

The advantage that a probabilistic polynomial time adversary $\mathcal{A}$ wins the above game is defined as $Adv = \mathbf{Pr}[b' = b] - \frac{1}{2}$.

Definition 1. The proposed fine-grained authorized keyword secure search scheme is semantically secure against an adaptively chosen-keyword attack if the advantage Adv that any probabilistic polynomial time adversary wins the above game is negligible.

Chosen-Plaintext Keyword Attack Game:

Setup The challenger $\mathcal{B}$ initializes running environment and public parameters to the adversary $\mathcal{A}$.

Phase 1 The adversary is allowed to access trapdoor encryption oracle, denoted by E, for many times and inputs two keywords w_0 and w_1.

Challenge The adversary sends w_0 and w_1 to the challenger. The challenger flips a random binary coin $b \in \{0, 1\}$ and encrypts w_b as $E(w_b)$, which is given the adversary.

Phase 2 The adversary continues to access the encryption oracle E.

Guess The adversary inputs the guess b' of b.

The advantage that a probabilistic polynomial time adversary $\mathcal{A}$ wins the above game is defined as $Adv = \mathbf{Pr}[b' = b] - \frac{1}{2}$.

Definition 2. The proposed fine-grained authorized keyword secure search scheme is semantically secure against a chosen-plaintext keyword attack if the advantage Adv that any probabilistic polynomial time adversary wins the above game is negligible.

3.3. Basic techniques

3.3.1. Bilinear pairing map

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ denote two cyclic multiplicative groups of order q. A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ satisfies the following properties:

- Computable: For any $Q, Z \in \mathbb{G}_1$, there is a polynomial time algorithm to compute $e(Q, Z) \in \mathbb{G}_2$.
- Bilinear: For all $x, y \in \mathbb{Z}_q^*$ and $Q, Z \in \mathbb{G}_1$, the equality $e(Q^x, Z^y) = e(Q, Z)^{xy}$ holds.
- Non-degenerate: If g, h are generators of $\mathbb{G}_1$, then $e(g, h)$ is a generator of $\mathbb{G}_2$.

3.3.2. Attribute-based encryption

Attribute-Based encryption (ABE) is able to effectively enforce access control over encrypted data in a fine-grained manner by leveraging an access control policy. Depending on that the access control policy is embedded into the decryption key or ciphertext, the ABE can generally be classified into two categories: KP-ABE (key-policy ABE) [15] and CP-ABE (ciphertext-policy ABE) [2]. Since an access tree can flexibly achieve AND, OR, and threshold operations among attributes, it is widely used to represent the access control policy in the ABE system.

1. Access Tree T In the access tree T, a non-leaf node and a leaf node represent a threshold gate and an attribute, respectively. Each node x in T contains two values: num_x and k_x. num_x is the number of child nodes of the node x and k_x is its threshold value $(1 \leq k_x \leq num_x)$. If x is a non-leaf node, $k_x = 1$ means the node is an OR gate and $k_x = num_x$ represents an AND gate. If x is a leaf node, then $k_x = 1$. For ease of description, several notations about an access tree are defined as follows.

- *parent(x)*: the parent of node x.
- *index(x)*: the label of node x. The child nodes of a node y in T are labeled from 1 to num.
- *attr(x)*: the attribute associated with the leaf node x.

2. Satisfying access tree T Let T_x be the subtree of T rooted at the node x. If a set of attributes S satisfies T_x, we denote it as $T_x(S) = 1$. $T_x(S)$ can be computed as follows. If x is a non-leaf node, evaluate $T_{x'}(S)$ for all children x' of node x. $T_x(S)$ return 1 if and only if at least k_x children return 1. If x is a leaf node, $T_x(S)$ returns 1 if and only if $attr(x) \in S$. Thus, according to the above recursive computation, if set S satisfies T, then $T_r(S) = 1$, where r is the root node of T.

3.3.3. Decisional Diffie–Hellman (DDH) problem and assumption

DDH Problem: Let g represent a generator of the group $\mathbb{G}$ with order q. There are three random elements a, b, c in $\mathbb{Z}_q^*$. Given (g, g^a, g^b), the problem is to distinguish the valid element g^{ab} from the random element g^c. A PPT algorithm $\mathcal{A}$ has an advantage $Adv_{\mathcal{A}}^{\text{DDHP}}$ in solving DDHP if:

$$Adv_{\mathcal{A}}^{\text{DDHP}} \leq |Pr[\mathcal{A}(g^a, g^b, g^{ab}) = 1] - Pr[\mathcal{A}(g^a, g^b, g^c) = 1]|$$

DDH assumption: for any probabilistic polynomial time algorithm $\mathcal{A}$, $Adv_{\mathcal{A}}^{\text{DDH}}$ is negligible.

3.3.4. Decisional Bilinear Diffie–Hellman (DBDH) problem and assumption

DBDH Problem: Let g represent a generator of the group $\mathbb{G}_1$ with order q. Suppose a challenger chooses four elements $a, b, c, z \in \mathbb{Z}_q^*$ at random and computes $g^a, g^b, g^c, e(g, g)^{abc}$, $e(g, g)^z$, where e is a bilinear map. The DBDH problem is to find a probabilistic polynomial time algorithm $\mathcal{A}$ has advantage $Adv_{\mathcal{A}}^{DBDH}$ in distinguishing the tuple $(g^a, g^b, g^c, e(g, g)^{abc})$ from the tuple $(g^a, g^b, g^c, e(g, g)^z)$ if

$$\begin{aligned} Adv_{\mathcal{A}}^{\text{DBDH}} \leq |&Pr[\mathcal{A}(g^a, g^b, g^c, e(g, g)^{abc}) = 1] \\ &- Pr[\mathcal{A}(g^a, g^b, g^c, e(g, g)^z) = 1]| \end{aligned}$$

DBDH Assumption: for any probabilistic polynomial time algorithm $\mathcal{A}$, $Adv_{\mathcal{A}}^{\text{DBDh}}$ is negligible.

4. Fine-grained authorized keyword based secure search scheme

In this section, we construct a fine-grained authorized keyword secure search scheme.

To achieve fine-grained authorized keyword search for different data users, our basic idea is that the data owner encrypts an index keyword under an access tree. An authorized data user whose attributes satisfy the access tree will be able to use the keyword to perform data searching. By defining different access trees for different index keywords, the data owner is able to flexibly determine which keywords can be searched by which authorized data users in a fine-grained manner. In addition, we organize the encrypted index keywords and encrypted data files as secure inverted index for achieving sub-linear search complexity [10,20].

4.1. Setting initialization

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic multiplicative groups with large prime order q. g is a generator of group $\mathbb{G}_1$. A bilinear map is denoted as $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with aforementioned three properties. We define two cryptography hash functions $H_1 : \{0, 1\}^* \rightarrow \mathbb{Z}_q^*$ hashing an arbitrary length string to an element in $\mathbb{Z}_q^*$ and $H_2 : \{0, 1\}^* \rightarrow \mathbb{G}_1$ hashing an arbitrary length string to a group element in $\mathbb{G}_1$. Finally, we define Lagrange coefficients as follows:

$$\triangle_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x - j}{i - j}$$

where S denotes a set of elements in $\mathbb{Z}_q^*$ and $i, j \in \mathbb{Z}_q^*$.

4.2. Key generation

In this key generation phase, the data owner chooses two random elements α, β from $\mathbb{Z}_q^*$ and generates two keys k_1, k_2 as follows:

$$\mathbb{KEY} = \begin{cases} k_1 = \left(\beta, \dfrac{1}{\beta}, g^\alpha\right) \\ k_2 = (e(g, g)^\alpha, h = g^\beta) \end{cases}$$

In our scheme, the data owner uses k_1 to grant query privilege for data users and uses k_2 to encrypted index keywords for constructing secure searchable index, respectively, where $\frac{1}{\beta} \in \mathbb{Z}_q^*$ denotes an inverse of element β.

4.3. Index keyword encryption and data outsourcing

Without loss of generality, we use notation w to denote an index keyword. The data owner takes the following two steps to generate encrypted posting list of the keyword w. First, the data owner defines an access tree T_w and chooses a polynomial q_x for each node x in T_w. These polynomials are chosen in a top-down manner, starting from the root node R. For each polynomial q_x of x, the data owner sets the degree d_x of q_x to be one less than the threshold value k_x of the node x, i.e., $d_x = k_x - 1$. For the root node R, he chooses a random element $s \in \mathbb{Z}_q^*$ and sets $q_R(0) = s$ and then sets d_R other points of q_R randomly to completely define it. For any other node x in T_w, he sets $q_x(0) = q_{parent(x)}(index(x))$ and chooses d_x other points to completely define q_x. Let Y be the set of leaf nodes T_w. The data owner uses k_2 and T_w to encrypt index keyword w as follows:

$$\begin{aligned} \mathcal{I}_w =&(T_w, \widetilde{\mathcal{I}}_w' = e(g, g)^{\alpha s H_1(w)}, \widetilde{\mathcal{I}}_w'' = h^{s H_1(w)}, \\ &\forall y \in Y : I_y = g^{q_y(0)}, I_y' = H_2(attr(y))^{q_y(0)}) \end{aligned}$$

Second, the data owner encrypts data files containing the keyword w by using semantically secure AES encryption scheme under the key k and then uses $\mathcal{I}_w$ to associate with all these encrypted data files to form the corresponding posting list.

4.4. Index keyword re-encryption

After receiving the encrypted index, the cloud server re-encrypts $\mathcal{I}_w$ for the keyword w to achieve effective and flexible data user search permission update in our scheme. To achieve this goal, the data owner needs to generate an attribute key K_a for each attribute a in the attribute universe $\mathbb{A}$. We use

$$\mathbb{AK} = \{\forall a \in \mathbb{A} : K_a\}$$

to denote the attribute key universe. The data owner uploads $\mathbb{AK}$ to the cloud server and the cloud server uses it to re-encrypt each posting list $\mathcal{I}_w$ as follows:

$$\begin{aligned}\mathcal{I}_w^* =&(T_w, \widetilde{I}_w' = e(g,g)^{\alpha s H_1(w)}, \widetilde{I}_w'' = h^{sH_1(w)},\\ &\forall y \in Y \wedge \forall K_{attr(y)} \in \mathbb{AK} : I_y = g^{q_y(0)},\\ &I_y' = (H_2(attr(y))^{q_y(0)})^{K_{attr(y)}})\end{aligned}$$

We emphasize that the purpose of introducing $\mathbb{AK}$ and re-encrypting each index keyword is to achieve effective search permission update in our query system. We defer the discussion of how to implement this property in the latter section.

4.5. Data user grant

When a data user wishes to join the system, the data user first needs to obtain the search privilege from the data owner. In this phase, a group of keys are distributed to the data user, by which the data user can generate legal query trapdoor for a certain query keyword.

Assume there is a new data user u, the process of granting search permission for u includes the following steps. First, the data owner assigns a set of attributes S for u and uses k_1 to generate the key based on S as:

$$\begin{aligned}\mathcal{G}_u =&(G_1 = g^{\frac{\alpha}{\beta}}, G_2 = g^{\frac{1}{\beta}}\\ &\forall a \in S, G_a = H_2(a)^{r_a}, G_a' = g^{r_a})\end{aligned}$$

where $r_a \in \mathbb{Z}_q^*$ is a randomly chosen element for each attribute $a \in S$. Second, the data owner generates an attribute key subset according to the attribute key universe $\mathbb{AK}$ and S as:

$$\overline{\mathbb{AK}} = \{\forall a \in S : K_a\} \subset \mathbb{AK}$$

Finally, the data owner sends tuple $(k, S, \mathcal{G}_u, \overline{\mathbb{AK}})$ to the data user through secure communication channels, where k is the symmetric encryption key of data files.

Upon receiving the tuple $(k, S, \mathcal{G}_u, \overline{\mathbb{AK}})$ from the data owner, u employs the attribute key subset $\overline{\mathbb{AK}}$ to update the $\mathcal{G}_u$ as follows:

$$\begin{aligned}\mathcal{G}_u^* =&(G_1 = g^{\frac{\alpha}{\beta}}, G_2 = g^{\frac{1}{\beta}},\\ &\forall a \in S \wedge \forall K_a \in \overline{\mathbb{AK}} : G_a = H_2(a)^{r_a},\\ &G_a' = (g^{r_a})^{\frac{1}{K_a}})\end{aligned}$$

In this paper, we consider an authorized data user to be trusted, which means he does not leak the received security parameters to other illegal users.

4.6. Trapdoor generation

Assume u wants to obtain all data files that contain keyword w_0, u uses $\mathcal{G}_u^*$ to encrypt w_0 to generate query trapdoor by doing the following. First, u randomly chooses an element $r \in \mathbb{Z}_q^*$ and computes $\lambda_1 = r \cdot H_1(w_0)$, $\lambda_2 = g^{\lambda_1} = g^{rH_1(w_0)}$, $T_1 = G_1(G_2)^r = g^{\frac{\alpha}{\beta}} \cdot g^{\frac{r}{\beta}} = g^{\frac{\alpha+r}{\beta}}$. Second, for each $a \in S$, u computes $T_a = \lambda_2 G_a = g^{rH_1(w_0)} H_2(a)^{r_a}$. Eventually, the trapdoor of query keyword w_0 is denoted as:

$$\begin{aligned}\mathcal{T}_u(w_0) =&(T_1 = g^{\frac{\alpha+r}{\beta}},\\ &\forall a \in S \wedge \forall K_a \in \overline{\mathbb{AK}} : T_a = g^{rH_1(w_0)} H_2(a)^{r_a},\\ &T_a' = G_a' = (g^{r_a})^{\frac{1}{K_a}}),\end{aligned}$$

which will be submitted to the cloud server.

4.7. Secure search over encrypted inverted index

Cloud is equipped with powerful storage capacity and computation power, besides storing all encrypted data files and secure searchable index, it is also responsible for performing search on behalf of data users. Upon receiving a query trapdoor $\mathcal{T}_u(w_0)$ of the query keyword w_0 submitted by data user u, the cloud will perform search over encrypted inverted index and return encrypted data files that contain the keyword w_0 if and only if the following two conditions must be satisfied simultaneously: (1) the cloud can find an encrypted posting list with keyword w that satisfies $w = w_0$ and (2) the data user u must hold the query privilege of the keyword w, i.e., u's attribute set assigned by the data owner must satisfy the access tree T_w. Otherwise, u cannot obtain any data files from the cloud server. Moreover, in the whole query process, the cloud cannot know any useful information about data files, inverted index, and u's query keyword.

In what follows, we describe the secure search process performed by the cloud server in detail. Given the query trapdoor $\mathcal{T}_u(w_0)$ and an encrypted posting list $\mathcal{I}_w^*$, for ease of understanding, we regard the whole search process as two sub-procedures, which are actually transparent for both cloud server and data users.

The first sub-procedure is to determine whether the data user u has access to $\mathcal{I}_w^*$, i.e, the cloud server decides whether u's attribute set S satisfies the access tree T_w. The detailed process can be described as follows.

Let x be a node from T_w,

- For each leaf node x, let $a = attr(x)$ denote the attribute associated with x, if $a \in S$, then compute:

$$\begin{aligned}F_x &= \frac{e(T_a, I_x)}{e(T_a', I_x')}\\ &= \frac{e(g^{rH_1(w_0)} \cdot H_2(a)^{r_a}, g^{q_x(0)})}{e((g^{r_a})^{\frac{1}{K_a}}, (H_2(a)^{q_x(0)})^{K_a})}\\ &= \frac{e(g^{rH_1(w_0)}, g^{q_x(0)}) \cdot e(H_2(a), g)^{r_a \cdot q_x(0)}}{e(H_2(a), g)^{r_a \cdot q_x(0) \cdot K_a \cdot \frac{1}{K_a}}}\\ &= e(g,g)^{rH_1(w_0) q_x(0)}\end{aligned}$$

  if $a \notin S$, we define $F_x = \perp$.
- For each non-leaf node x in T_{w_0} (In a down-top and recursive manner), let S_x denote an arbitrary k_x-sized set of child nodes z such that $F_z \neq \perp$; if no such set exists then the node is not satisfied by u's attribute set S and define $F_x = \perp$; otherwise, compute using Lagrange interpolation:

$$\begin{aligned}F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,S_x'}(0)}\\ &= \prod_{z \in S_x} (e(g,g)^{rH_1(w_0) \cdot q_z(0)})^{\Delta_{i,S_x'}(0)}\\ &= \prod_{z \in S_x} (e(g,g)^{rH_1(w_0) \cdot q_{parent(z)}(index(z))})^{\Delta_{i,S_x'}(0)}\\ &= \prod_{z \in S_x} e(g,g)^{rH_1(w_0) \cdot q_x(i) \cdot \Delta_{i,S_x'}(0)}\\ &= e(g,g)^{rH_1(w_0) \cdot q_x(0)}\end{aligned}$$

  where $i = index(z)$, $S_x' = (\forall z \in S_x : index(z))$, and $\Delta_{i,S_x'}$ is the Lagrange coefficient.
- For the root node R of the access tree T_w, according to the above recursive operation, if $F_R = \perp$ then T is not satisfied by the u's attribute set S; otherwise,

$$F_R = e(g,g)^{rH_1(w_0) \cdot q_R(0)} = e(g,g)^{rH_1(w_0)s}$$

After finishing the above recursive operations, if $F_R = \perp$, then this means that u's attribute set S does not satisfy T_w; otherwise, the cloud server continues to perform the second sub-procedure to judge whether the query keyword w_0 satisfies $w_0 = w$ by verifying whether the following equation is true:

$$\widetilde{I}'_w = \frac{e(\widetilde{I}''_w, T_1)}{F_R}$$

If the equation is true, this means that $w = w_0$ holds. u obtains all encrypted data files that contain keyword $w(w_0)$ according to the posting list $\mathcal{I}^*_w$ from the cloud server. Finally, u uses the symmetric key k to decrypt them locally.

Now, we verify the correctness of the search by the following derivation:

$$\begin{aligned}\frac{e(\widetilde{I}''_w, T_1)}{F_R} &= \frac{e(h^{sH_1(w)}, g^{\frac{\alpha+r}{\beta}})}{e(g,g)^{rH_1(w_0)s}} \\ &= \frac{e(g^{\beta sH_1(w)}, g^{\frac{\alpha+r}{\beta}})}{e(g,g)^{rH_1(w_0)s}} \\ &= \frac{e(g^{sH_1(w)}, g^{\alpha+r})}{e(g,g)^{rH_1(w_0)s}} \\ &= \frac{e(g^{sH_1(w)}, g^{\alpha})e(g^{sH_1(w)}, g^{r})}{e(g,g)^{rH_1(w_0)s}} \\ &= \frac{e(g,g)^{\alpha sH_1(w)}e(g,g)^{rH_1(w)s}}{e(g,g)^{rH_1(w_0)s}}\end{aligned}$$

If the query keyword w_0 is exactly the same as w (i.e., $H_1(w) = H_1(w_0)$) then,

$$\begin{aligned}\frac{e(\widetilde{I}''_w, T_1)}{F_R} &= \frac{e(g,g)^{\alpha sH_1(w)}e(g,g)^{rH_1(w)s}}{e(g,g)^{rH_1(w_0)s}} \\ &= e(g,g)^{\alpha sH_1(w)} \\ &= \widetilde{I}'_w\end{aligned}$$

5. User search permission update

In this section, we design two attribute update sub-protocols: an attribute addition sub-protocol and an attribute revocation sub-protocol. By the proposed attribute update sub-protocols, the data owner is able to flexibly update search permission for a data user when the data user's role changes in the system. We first present the construction of our protocol and then make a discussion about it from functionality, performance, and correctness three aspects.

5.1. Attribute revocation sub-protocol

When the data owner decides to revoke a previously assigned attribute a from a specified data user u, the system needs to run the attribute revocation protocol among the data user, the cloud server, and other data users except u. We describe running details of this protocol from different entities as follows.

- **Data Owner:** The data owner generates a new attribute key K'_a for a to replace the old attribute key K_a in the set $\mathbb{AK}$ and then sends K'_a to the cloud server and data users who have been assigned the attribute a except u.
- **Cloud Server:** Upon receiving the K'_a, the cloud server first adds it to $\mathbb{AK}$ and deletes the old key K_a for the attribute a. Then, for each encrypted posting list with the keyword w, if existing one leaf node in the access tree T_w associates with the attribute a, then the cloud server updates the value I'_a of $\mathcal{I}^*_w$ as $(I'_a)^{K'_a}$ for the attribute a and keeping other leaf nodes unchanged. The new $\mathcal{I}^*_w$ is denoted as follows (assume that the lead node x in T_w associates the attribute a i.e., $a = attr(x)$):

$$\begin{aligned}\mathcal{I}^*_w =&(T_w, \widetilde{I}'_w = e(g,g)^{\alpha sH_1(w)}, \widetilde{I}''_w = h^{sH_1(w)}, \\ &I_x = g^{q_x(0)}, I'_x = (H_2(attr(x))^{q_x(0)})^{K_{attr(x)}K'_{attr(x)}} \\ &\forall y \in Y \wedge \forall K_{attr(y)} \in \mathbb{AK} \setminus \{x\} : I_y = g^{q_y(0)}, \\ &I'_y = (H_2(attr(y))^{q_y(0)})^{K_{attr(y)}})\end{aligned}$$

- **Data Users:** when a data user u' who has been assigned the attribute a receives the updated attribute key K'_a from the data owner, he first updates his attribute key subset $\overline{\mathbb{AK}}$ by replacing the previous key K_a using K'_a and then updates the previous G'_a as $(G'_a)^{\frac{1}{K'_a}}$ and keeps other attribute keys unchanged. The new $\mathcal{G}^*_{u'}$ is denoted by:

$$\begin{aligned}\mathcal{G}^*_{u'} &= (G_1 = g^{\frac{\alpha}{\beta}}, G_2 = g^{\frac{1}{\beta}}, \\ &G_a = H_2(a)^{r_a}, G'_a = (g^{r_a})^{\frac{1}{K_aK'_a}} \\ &\forall b \in S \wedge \forall K_b \in \overline{\mathbb{AK}} \setminus \{a\}, G_b = H_2(b)^{r_b} \\ &G'_b = (g^{r_b})^{\frac{1}{K_b}})\end{aligned}$$

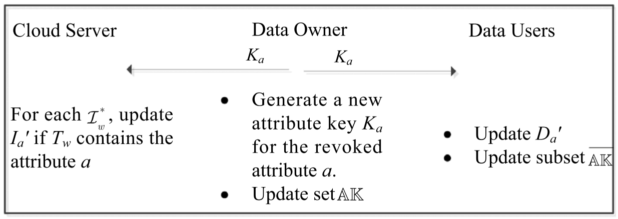

Cloud Server
Data Owner
Data Users
K_a
K_a
For each $\mathcal{I}^*_w$, update I_a' if T_w contains the attribute a
Generate a new attribute key K_a for the revoked attribute a.
Update set $\mathbb{AK}$
Update D_a'
Update subset $\overline{\mathbb{AK}}$

Fig. 2. Attribute revocation sub-protocol.

A high-level description of the attribute revocation protocol is given in Fig. 2.

5.2. Attribute addition sub-protocol

When the data owner decides to add a new attribute b to a specified data user u, the system runs the attribute addition protocol between the data owner and the data user u. We describe running details of this protocol from the two different entities as follows.

- **Data Owner:** The data owner randomly chooses an element $r_b \in \mathbb{Z}^*_q$ and uses attribute key $K_b \in \mathbb{AK}$ of attribute b to compute $G_b = H_2(b)^{r_b}$ and $G'_b = (g^{r_b})^{\frac{1}{K_b}}$. The tuple (b, K_b, G_b, G'_b) is sent to u via secure communication channels.
- **Data User:** Upon receiving tuple (b, K_b, G_b, G'_b) from the data owner, u updates following sets:
  (1) Update previous query keyword encryption secret key set $\mathcal{G}^*_u$ as

$$\begin{aligned}\mathcal{G}^*_u =&(G_1 = g^{\frac{\alpha}{\beta}}, G_2 = g^{\frac{1}{\beta}}, \\ &(\forall a \in S \wedge \forall K_a \in \overline{\mathbb{AK}} : G_a = H_2(a)^{r_a}, \\ &G'_a = (g^{r_a})^{\frac{1}{K_a}}) \bigcup (G_b = H_2(b)^{r_b}, G'_b = (g^{r_b})^{\frac{1}{K_b}}))\end{aligned}$$

  (2) Update previous attribute set S as $S := S \cup \{b\}$.
  (3) Update previous attribute key subset $\overline{\mathbb{AK}} := \overline{\mathbb{AK}} \cup \{K_b\}$.

A high-level description of the attribute addition protocol is given in Fig. 3.

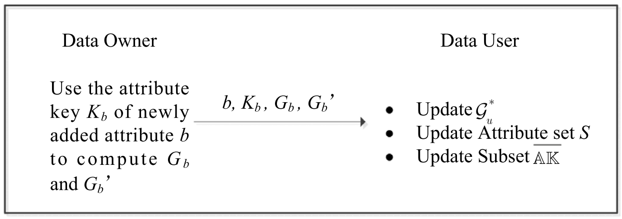

Data Owner
Data User
Use the attribute key K_b of newly added attribute b to compute G_b and G_b'
b, K_b, G_b, G_b'
Update $\mathcal{G}_u^*$
Update Attribute set S
Update Subset $\mathbb{AK}$

Fig. 3. Attribute addition sub-protocol.

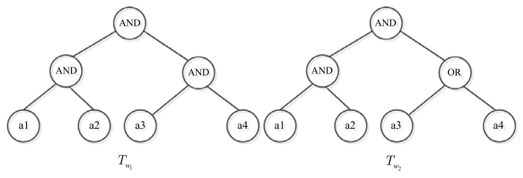


Fig. 4. Access tree of encrypting index keyword w_1 and w_2, respectively.

5.3. Discussion

5.3.1. Functionality

Our proposed search permission update protocol allows the data owner to achieve flexible and fine-grained keyword search privilege control by revoking or adding attributes from or to a data user when the data user's role changes in the system. For example, for an encrypted posting list $\mathcal{I}_w^*$, once certain attributes are revoked from a data user that leads to his remaining attributes no more satisfy the access tree T_w. Accordingly, the data user loses the search privilege of the posting list with keyword w; meanwhile, the data user still keeps search privilege of other posting lists as long as his remaining attributes satisfy corresponding access trees. On the other hand, adding new attributes for a data user is equivalent to granting new keyword search permissions for the data user. Obviously, a data user can be easily revoked from the search system thoroughly by revoking his all attributes previously assigned.

5.3.2. Performance

The proposed search permission update protocol is extremely efficient and only needs a few computation cost and communication cost for the data owner, the cloud server, and data users. We can see that, in these two sub-protocols, the data owner does not need to re-encrypt data files and re-build secure index, which significantly releases the data owner's operations. As shown in Figs. 2 and 3, when the user attribute update occurs, we can see that the data owner, the cloud server, and data users may need to perform a few operations. We will numerically describe performance cost of these operations in Sections 6 and 8.

5.3.3. Correctness

Since the correctness of attribute adding protocol is obvious, we use an example to illustrate the correctness of the attribute revocation protocol. Assume that there are a data user *Alice* who has attributes $\{a_1, a_2, a_3, a_4\}$ and two encrypted keywords $\mathcal{I}_{w_1}^*$ and $\mathcal{I}_{w_2}^*$, which is embedded the access tree T_{w_1} and T_{w_2}, respectively, as shown in Fig. 4. Obviously, *Alice* has the query permissions of keywords w_1 and w_2, since her attribute set satisfies access trees T_{w_1} and T_{w_2} simultaneously.

At some point, if attribute $a3$ is revoked from *Alice*, *Alice* loses the query permission of keyword w_1 even she still owns a_3 from his own perspective. This is because that *Alice* does not obtain the updated attribute key K'_{a_3} from the data owner and cannot update the element G'_{a_3} of her query trapdoor generation key $\mathcal{G}^*_{Alice}$. When

Table 1
Notations used in performance analysis.

Notations	Description
P	Pairing operation
$M_{\mathbb{G}}$	Multiplication operation in group $\mathbb{G}$
$M_{\mathbb{Z}_q^*}$	Multiplication operation in $\mathbb{Z}_q^*$
$E_{\mathbb{G}}$	Exponentiation operation in group $\mathbb{G}$
H	Hash computation
$\|\alpha\|$	Bit length of α, if α is a string; Cardinality of α, if α is a set.
Y	Leaf-node set of an access tree
S	Attribute set of a data user
N	Least interior nodes satisfying an access tree

Table 2
The computation cost and output size of each phase in our secure search scheme.

Phase	Computation cost	Output size
KG	$P + 2E_{\mathbb{G}_1}$	$2\|\mathbb{Z}_q^*\|+2\|\mathbb{G}_1\|+\|\mathbb{G}_2\|$
IKE	$(2\|Y\|+1)E_{\mathbb{G}_1} + E_{\mathbb{G}_2} + M_{\mathbb{Z}_q^*} + \|Y\|H_2$	$(2Y + 1)\|\mathbb{G}_1\|+\|\mathbb{G}_2\|$
IKR	$\|Y\|E_{\mathbb{G}_1}$	$(2Y + 1)\|\mathbb{G}_1\|+\|\mathbb{G}_2\|$
DUG	$(3\|S\|+2)E_{\mathbb{G}_1} + \|S\|H_2$	$2(\|S\|+1)\|\mathbb{G}_1\|$
TG	$H_1 + M_{\mathbb{Z}_q^*} + 2E_{\mathbb{G}_1} + (\|S\|+1)M_{\mathbb{G}_1}$	$2(\|S\|+1)\|\mathbb{G}_1\|$
SS	$(2\|S\|+1)P + (\|N\|+1)M_{\mathbb{G}_2} + \|N\|E_{\mathbb{G}_2}$	0

Alice continues to use $\mathcal{G}^*_{Alice}$ to encrypt keyword w_1 to request data files from the cloud server, the cloud server first needs to compute F_x for each leaf node x in T_{w_1}. Correspondingly, the leaf node a_3 is computed as follows:

$$\begin{aligned}
F_{a_3} &= \frac{e(T_{a_3}, I_{a_3})}{e(T'_{a_3}, I'_{a_3})} \\
&= \frac{e(g^{rH_1(w_1)} \cdot H_2(a_3)^{r_{a_3}}, g^{q_{a_3}(0)})}{e((g^{r_{a_3}})^{\frac{1}{K_{a_3}}}, (H_2(a_3)^{q_{a_3}(0)})^{K_{a_3}K'_{a_3}})} \\
&= \frac{e(g^{rH_1(w_1)}, g^{q_{a_3}(0)}) \cdot e(H_2(a_3), g)^{r_{a_3}\cdot q_{a_3}(0)}}{e(H_2(a_3), g)^{r_{a_3}\cdot q_{a_3}(0)\cdot K_{a_3}\cdot K'_{a_3}\cdot \frac{1}{K_{a_3}}}} \\
&= \frac{e(g^{rH_1(w_1)}, g^{q_{a_3}(0)}) \cdot e(H_2(a_3), g)^{r_{a_3}\cdot q_{a_3}(0)}}{e(H_2(a_3), g)^{r_{a_3}\cdot q_{a_3}(0)\cdot K'_{a_3}}} \\
&\neq e(g, g)^{rH_1(w_1)q_{a_3}(0)}
\end{aligned}$$

We can see that the parent node of a_3 and a_4 is an AND gate. According to our proposed scheme, $F_{a_3} \neq e(g, g)^{rH_1(w_1)q_{a_3}(0)}$ will lead to $F_{R_1} = \perp$ for the root node R_1 of T_{w_1} in the recursion process. That is, *Alice*'s attribute set no longer satisfies T_{w_1}. On the other hand, *Alice* can still search the keyword w_2, since the correct value $F_{R_2} = e(g, g)^{rH_1(w_1)\cdot q_{R_2}(0)}$ can be computed by node a_4 for the OR gate.

6. Performance analysis

This section provides the performance analysis for the proposed authorized keyword search scheme and the search permission update protocol. For clear and accurate performance description, we first define several necessary notations as shown in Table 1.

6.1. Performance of authorized keyword search scheme

From the view of system level operations, the proposed fine-grained authorized keyword secure search scheme described in Section 4 includes *Key Generation (KG)*, *Index Keyword Encryption (IKE)*, *Index Keyword Re-encryption (IKR)*, *Data User Grant (DUG)*, *Trapdoor Generation (TG)*, *Secure Search (SS)*. For ease of reading, we describe the computation cost and output size of each phase in Table 2.

Table 3
The computation cost and output size of attribute update protocol.

Protocol	Entity	Computation cost	Output size
ARP	*Data owner*	0	$\|\mathbb{Z}_q^*\|$
	Cloud server	$n_a E_{\mathbb{G}_1}$	$n_a\|\mathbb{G}_1\|$
	Data user	$E_{\mathbb{G}_1}$	$\|\mathbb{G}_1\|$
AAP	*Data owner*	$2E_{\mathbb{G}_1}$	$2\|\mathbb{G}_1\|+2\|\mathbb{Z}_q^*\|$
	Data user	0	0

6.2. Performance of search permission update protocol

The search permission update protocol includes two subprotocols: *attribute revocation protocol (ARP)* and *attribute addition protocol (AAP)*, which are run among entities when data user's system role changes. For ease of reading, we describe the computation cost and output size of each entity in Table 3.

In Table 3, the notation n_a denotes the total number of encrypted posting lists whose access trees contain attribute a and we ignore the time cost of set union operation.

7. Security proof and analysis

In this section, we first provide the security proofs of the searchable index and trapdoor construction and then give thorough security analysis for our proposed scheme.

7.1. Security proof

Let $Search(\mathcal{I}_w, \mathcal{T}(w'))$ denote the secure search algorithm, which takes an index $\mathcal{I}_w$ and a trapdoor $\mathcal{T}(w')$ as input, and outputs 1 if the attribute set in $\mathcal{T}(w')$ satisfies the access tree in $\mathcal{I}_w$ and $w = w'$; otherwise, it outputs 0 (It's concrete implementation is present in Section 4.7).

Intuitively, the adversary $\mathcal{A}$ has two approaches to distinguish the challenge ciphertext $\mathcal{I}_{w_0}$ from $\mathcal{I}_{w_1}$ (i.e., decide $b = 1$ or $b = 0$): one is to access the search algorithm and judge $Search(\mathcal{I}_{w_b}, \mathcal{T}(w_0)) = 1$ or $Search(\mathcal{I}_{w_b}, \mathcal{T}(w_1)) = 1$; the other is to directly recover the message w_b from the original ciphertext $\mathcal{I}_{w_b}$. Now, we prove that in the above two games $\mathcal{A}$ cannot break our proposed scheme with a non-negligible advantage.

Theorem 1. *Our proposed fine-grained authorized keyword secure search scheme is semantically secure against an adaptively chosen-keyword attack in the generic bilinear group model.*

Proof. We prove that our proposed scheme is semantically secure in the generic bilinear group model based on the following adaptive chosen-keyword security game between the challenger $\mathcal{B}$ and the adversary $\mathcal{A}$, where H_2 is modeled as a random oracle and H_1 is a secure hash function.

Setup. The challenger $\mathcal{B}$ initializes running environment and sends $(\mathbb{G}_1, \mathbb{G}_2, e, H_1, H_2, g, q, h = g^{\beta}, e(g,g)^{\alpha})$ to the adversary $\mathcal{A}$.

Phase 1. When $\mathcal{A}$ requests the value of H_2 on any string a, $\mathcal{B}$ generate a new random value $t_a \in \mathbb{Z}_q^*$ and provides g^{t_a} as the response to $H_2(a)$.

On a trapdoor query corresponding to the attribute set $S_j \in [S_1, S_q]$, a new random value $r^{(j)} \in \mathbb{Z}_q^*$ is chosen, for each attribute $a_i \in S_j$, $\mathcal{B}$ randomly chooses a new value $r_a^{(j)}$ and computes $G = g^{\frac{\alpha+r^{(j)}}{\beta}}, G' = g^{r^{(j)}}, G_a = g^{t_a r_a^{(j)}}, G'_a = g^{r_a^{(j)}}$. Finally, for any keyword w chosen by $\mathcal{A}$, $\mathcal{A}$ generates the trapdoor as:

$$\mathcal{T}_{\mathcal{A}}(w) = (T_1 = g^{\frac{\alpha+r^{(j)}}{\beta}},$$
$$\forall a \in S_j : T_a = g^{r^{(j)}H_1(w)} g^{t_a r_a^{(j)}}, T'_a = g^{r_a^{(j)}}),$$

Table 4
Encryption functions used in the security proof.

Function	Construction	Public key	Private key	Random value
$E1$	$e(g,g)^{\alpha s H_1(w)}$	$e(g,g)^{\alpha}$	α	s
$E2$	$g^{\beta s H_1(w)}$	g^{β}	β	s
$E3$	$g^{r H_1(w)} H_2(a)^{r_a}$	g	α, β	r, r_a

which is sent to $\mathcal{A}$. Obviously, this is a legal trapdoor construction of w from $\mathcal{A}$'s perspective, where we do not consider the attribute key subset, since it does not influence the security.

Challenge. $\mathcal{A}$ defines an access tree T^* such that none of the attribute sets $S_1, \ldots, S_q$ from Phase 1 satisfy it and chooses two challenge keywords w_0 and sends w_1. w_0, w_1, and T^* to $\mathcal{B}$. $\mathcal{B}$ randomly chooses a bit $b \in \{0,1\}$ and encrypts w_b with T^* as:

$$\mathcal{I}_{w_b} = (\mathrm{T}^*, \widetilde{\mathcal{I}}'_{w_b} = e(g,g)^{\alpha s H_1(w_b)}, \widetilde{\mathcal{I}}''_w = h^{s H_1(w_b)},$$
$$\forall a \in Y : I_a = g^{q_a(0)}, I'_a = g^{t_a q_a(0)}),$$

where Y is the leaf node set of T^*.

Phase 2. $\mathcal{A}$ repeats Phase 1 for any keyword w with attribute set S_{q_w}. The only restriction if $w = w_0$ or $w = w_1$, then $S_{q_{w_0}}$ or $S_{q_{w_1}}$ does not satisfy T^*.

Guess. In our scheme, intuitively the adversary can output correct b by using the trapdoor $\mathcal{T}_{\mathcal{A}}(w)$ with attribute set S_{q_w} and to access the search algorithm $Search(\mathcal{I}_{w_b}, \mathcal{T}_{\mathcal{A}}(w))$ to decide $b = 0$ or $b = 1$. There exist two cases as below.

(1) If $w = w_0/w_1$, we claim that the adversary cannot output correct b by letting $Search(\mathcal{I}_{w_b}, \mathcal{T}_{\mathcal{A}}(w_0)) = 1$ or $Search(\mathcal{I}_{w_b}, \mathcal{T}_{\mathcal{A}}(w_1)) = 1$, since neither $\mathcal{S}_{q_{w_0}}$ nor $\mathcal{S}_{q_{w_1}}$ satisfies T^* encrypting $\mathcal{I}_{w_b}$. The intrinsic reason is that in CP-ABE, the adversary can never construct a query for $e(g,g)^{\gamma\alpha s}$, where γ can be set as $H_1(w_b)$ in this paper, when the attribute set does not satisfy the access tree. Detailed proof has been provided in [2]. Therefore, in our scheme, the adversary can never obtain the correct $e(g,g)^{H_1(w_b)\alpha s}$ by $e(\widetilde{I}''_{w_b}, T_1)/F_R$. Without the value and a correct comparison between the value and $\widetilde{I}'_{w_b}$, the *Search* algorithm always outputs 0 even $w = w_0/w_1$.

(2) If $w \neq w_0/w_1$ and the attribute set $\mathcal{S}_{q_w}$ satisfies T^*. Since $\mathcal{S}_{q_w}$ satisfies T^*, $\mathcal{A}$ can compute $e(\widetilde{I}''_{w_b}, T_1)/F_R = e(g,g)^{\alpha s H_1(w_b)} e(g,g)^{r H_1(w_b)s} / e(g,g)^{r H_1(w)s}$. Obviously, the value $e(g,g)^{H_1(w_b)\alpha s}$ cannot be effectively calculated due to $w \neq w_0/w_1$, which causes that *Search* algorithm always outputs 0. □

On the other hand, the adversary $\mathcal{A}$ can disclose the keyword information from ciphertexts $\widetilde{I}'_w = e(g,g)^{\alpha s H_1(w)}, \widetilde{I}''_w = h^{s H_1(w)} = g^{\beta s H_1(w)}, T_a = g^{r H_1(w)} H_2(a)^{r_a}$ from index $\mathcal{I}_w$ and trapdoor $\mathcal{T}_u(w)$. Next, we prove that these encryptions are semantically secure against chosen-plaintext attack. For ease of description, we define three encryption functions according to the above ciphertext constructions, as shown in Table 4, which are modeled as the encryption oracle in our proof.

Theorem 2. *The index keyword encryption E_1 is semantically secure against chosen-plaintext keyword attack if DBDH assumption holds.*

Proof. Suppose there exists a probabilistic polynomial time adversary $\mathcal{A}$ that has a non-negligible advantage ϵ to break E_1, we can construct a simulator $\mathcal{B}$ who can solve the DBDH problem with a non-negligible advantage $\frac{\epsilon}{2}$.

The challenger $\mathcal{C}$ first flips a binary coin μ. If $\mu = 0$, $\mathcal{C}$ sets tuple $t_0 : (g, A = g^a, B = g^b, C = g^c, Z = e(g,g)^{abc})$; if $\mu = 1$, he sets tuple $t_1 : (g, A = g^a, B = g^b, C = g^c, Z = e(g,g)^z)$, where a, b, c, z are chosen from $\mathbb{Z}_q^*$ at random uniformly. Tuple t_μ is sent to simulator $\mathcal{B}$. The simulator $\mathcal{B}$ plays the following game with adversary $\mathcal{A}$ on behalf of challenger $\mathcal{C}$.

Setup $\mathcal{B}$ sends the public parameter $(\mathbb{G}_1, \mathbb{G}_2, g, q, e, H_1, H_2)$ to $\mathcal{A}$.

Phase 1 $\mathcal{A}$ accesses the encryption function E_1 many times using arbitrary keywords to ask corresponding ciphertext. Finally, he outputs two keywords w_1 and w_2 and sends them to $\mathcal{B}$.
Challenge $\mathcal{B}$ flips a binary coin γ and encrypts keyword w_γ as $\mathcal{E} = (Z)^{H_1(w_\gamma)}$.

If $\mu = 0$, $Z = e(g,g)^{abc}$. Since α and s are also randomly chosen elements in encryption block E_1, we let $ab = \alpha$ and $c = s$. Thus, we have $\mathcal{E} = (Z)^{H_1(w_\gamma)} = (e(g,g)^{ab})^{cH_1(w_\gamma)} = e(g,g)^{\alpha s H_1(w_\gamma)}$. Therefore, $\mathcal{E}$ is a valid index keyword encryption using E_1. If $\mu = 1$, $Z = e(g,g)^z$. Then we have $\mathcal{E} = e(g,g)^{zH_1(w_\gamma)}$. Since z is a random element, therefore $\mathcal{E}$ is a random element in $\mathbb{G}_2$ from $\mathcal{A}$'s perspective and contains no information about w_γ.
Phase 2 $\mathcal{A}$ continues to ask the encryption oracle E_1.
Guess $\mathcal{A}$ outputs a guess γ' of γ. If $\gamma' = \gamma$, then $\mathcal{B}$ outputs the guess $\mu' = 0$ of μ. This means $\mathcal{C}$ sent the valid encryption tuple $t_0 : (g, A = g^a, B = g^b, C = g^c, Z = e(g,g)^{abc})$ to $\mathcal{B}$. Since $\mathcal{A}$ has advantage ϵ to break E_1, therefore, the probability $\mathcal{A}$ outputs guess γ' of γ satisfying $\gamma' = \gamma$ is $\frac{1}{2} + \epsilon$. Correspondingly, the probability that $\mathcal{B}$ outputs guess μ' of μ satisfying $\mu' = \mu = 0$ is $\frac{1}{2} + \epsilon$. If $\gamma' \neq \gamma$, then $\mathcal{B}$ outputs the guess $\mu' = 1$ of μ. This means random tuple t_1 was sent to $\mathcal{B}$. Therefore, the probability $\mathcal{A}$ outputs guess γ' of γ satisfying $\gamma' = \gamma$ is $\frac{1}{2}$. Correspondingly, the probability that $\mathcal{B}$ outputs guess μ' of μ satisfying $\mu' = \mu = 1$ is $\frac{1}{2}$.

Hence, the overall advantage that $\mathcal{B}$ solves the DBDH problem can be computed:

$$\left|\frac{1}{2}Pr[\mu = \mu'|\mu = 0] + \frac{1}{2}Pr[\mu = \mu'|\mu = 1] - \frac{1}{2}\right|$$
$$= \left|\left[\frac{1}{2}\left(\frac{1}{2} + \epsilon\right) + \frac{1}{2}\cdot\frac{1}{2}\right] - \frac{1}{2}\right| = \frac{\epsilon}{2}$$

Since ϵ is non-negligible, therefore $\frac{\epsilon}{2}$ is also non-negligible. This conclusion means $\mathcal{B}$ is able to solve the DBDH problem with a non-negligible advantage, which contradicts DBDH problem assumption. □

Theorem 3. *The index keyword encryption E_2 is semantically secure against chosen-plaintext keyword attack if DDH assumption holds.*

Proof. Suppose there exists a probabilistic polynomial time adversary $\mathcal{A}$ has a non-negligible advantage ϵ to break E_2, we can construct a simulator $\mathcal{B}$ who can solve the DDH problem with a non-negligible advantage $\frac{\epsilon}{2}$.

The challenger $\mathcal{C}$ first flips a binary coin μ. If $\mu = 0$, $\mathcal{C}$ sets tuple $t_0 : (g, A = g^a, B = g^b, C = g^{ab})$; if $\mu = 1$, he sets tuple $t_1 : (g, A = g^a, B = g^b, C = g^c)$, where a, b, c, are chosen from $\mathbb{Z}_q^*$ at random uniformly. Tuple t_μ is sent to simulator $\mathcal{B}$. The simulator $\mathcal{B}$ plays the following game with adversary $\mathcal{A}$ on behalf of challenger $\mathcal{C}$.
Setup $\mathcal{B}$ sends the public parameter $(\mathbb{G}_1, \mathbb{G}_2, g, q, e, H_1, H_2)$ to $\mathcal{A}$.
Phase 1 $\mathcal{A}$ accesses the encryption function E_2 many times using arbitrary keywords to ask corresponding ciphertext. Finally, he outputs two keywords w_1 and w_2 and sends them to $\mathcal{B}$.
Challenge $\mathcal{B}$ flips a binary coin γ and encrypts keyword w_γ as $\mathcal{E} = (C)^{H_1(w_\gamma)}$. If $\mu = 0$, $C = g^{ab}$. Since β and s are also randomly chosen elements in encryption block E_2, we let $a = \beta$ and $b = s$. Thus, we have $\mathcal{E} = (C)^{H_1(w_\gamma)} = (g^{ab})^{H_1(w_\gamma)} = g^{\beta s H_1(w_\gamma)}$. Therefore, $\mathcal{E}$ is a valid index keyword encryption using E_2. If $\mu = 1$, $C = g^c$. Then we have $\mathcal{E} = g^{cH_1(w_\gamma)}$. Since c is a random element, therefore $\mathcal{E}$ is a random element in $\mathbb{G}_1$ from $\mathcal{A}$'s perspective and contains no information about w_γ.
Phase 2 $\mathcal{A}$ continues to ask the encryption oracle E_2.
Guess $\mathcal{A}$ outputs a guess γ' of γ. If $\gamma' = \gamma$, then $\mathcal{B}$ outputs the guess $\mu' = 0$ of μ. This means $\mathcal{C}$ sent the valid encryption tuple $t_0 : (g, A = g^a, B = g^b, C = g^{ab})$ to $\mathcal{B}$. Since $\mathcal{A}$ has advantage ϵ to break E_2, therefore, the probability $\mathcal{A}$ outputs guess γ' of γ satisfying $\gamma' = \gamma$ is $\frac{1}{2} + \epsilon$. Correspondingly, the probability that $\mathcal{B}$ outputs guess μ' of μ satisfying $\mu' = \mu = 0$ is $\frac{1}{2} + \epsilon$. If $\gamma' \neq \gamma$, then $\mathcal{B}$ outputs the guess $\mu' = 1$ of μ. This means random tuple t_1 was sent to $\mathcal{B}$. Therefore, the probability $\mathcal{A}$ outputs guess γ' of γ satisfying $\gamma' = \gamma$ is $\frac{1}{2}$. Correspondingly, the probability that $\mathcal{B}$ outputs guess μ' of μ satisfying $\mu' = \mu = 1$ is $\frac{1}{2}$.

Hence, the overall advantage that $\mathcal{B}$ solves the DDH problem can be computed:

$$\left|\frac{1}{2}Pr[\mu = \mu'|\mu = 0] + \frac{1}{2}Pr[\mu = \mu'|\mu = 1] - \frac{1}{2}\right|$$
$$= \left|\left[\frac{1}{2}\left(\frac{1}{2} + \epsilon\right) + \frac{1}{2}\cdot\frac{1}{2}\right] - \frac{1}{2}\right| = \frac{\epsilon}{2}$$

Since ϵ is non-negligible, therefore $\frac{\epsilon}{2}$ is also non-negligible. This conclusion means $\mathcal{B}$ is able to solve the DDH problem with a non-negligible advantage, which contradicts DDH problem assumption. □

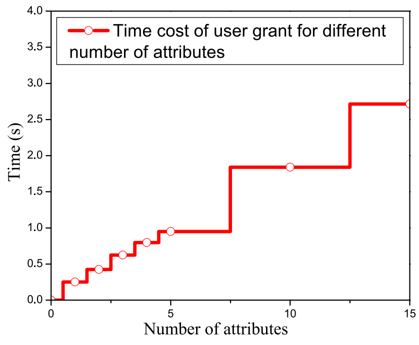
Theorem 4. *The query keyword encryption E_3 is semantically secure against chosen-plaintext keyword attack if DDH assumption holds.*

Proof. The proof of this theorem is similar to that of Theorem 3. The only difference is in the **Challenge** phase due to different encryption construction. We only describe this part of contents as follows.
Challenge $\mathcal{B}$ flips a binary coin γ and encrypts keyword w_γ as $\mathcal{E} = (C)^{H_1(w_\gamma)}H_2(a)^{r_a}$, where $H_2(a)^{r_a}$ can be considered as the public parameter from $\mathcal{B}$'s perspective, although the data user never publish it to anyone else in practice.
If $\mu = 0$, $C = g^{ab}$. Since r is also randomly chosen elements in encryption block E_3, we let $r = ab$. Thus, we have $\mathcal{E} = (C)^{H_1(w_\gamma)}H_2(a)^{r_a} = \mathcal{E} = (g^{ab})^{H_1(w_\gamma)}H_2(a)^{r_a} = \mathcal{E} = g^{rH_1(w_\gamma)}H_2(a)^{r_a}$. Therefore, $\mathcal{E}$ is a valid index keyword encryption using E_3. If $\mu = 1$, $C = g^c$. Then we have $\mathcal{E} = g^{cH_1(w_\gamma)}H_2(a)^{r_a}$. Since c is a random element, therefore $\mathcal{E}$ is a random element in $\mathbb{G}_1$ from $\mathcal{A}$'s perspective and contains no information about w_γ. □

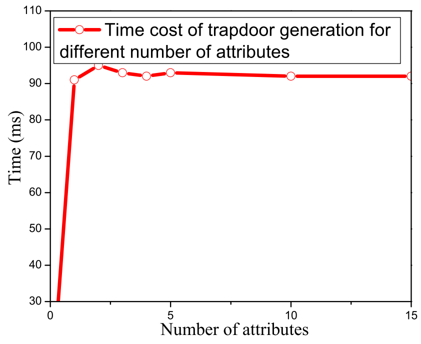
7.2. Security analysis

We provide a thorough security analysis for our scheme from four aspects, i.e., data files security, searchable index security, search security, and trapdoor privacy and unlinkability.

- **Security of data files**: The security of data files can be well guaranteed by using the semantically secure symmetric encryption such as AES.
- **Security of search index**: In each encrypted posting list, the corresponding index keyword is contained in encryption block E_1 and E_2, which has been proved to be semantically secure against chosen-plaintext keyword attack under the DBDH and DDH assumption, respectively. This indicates that the cloud server cannot recover the index keyword information from each encrypted posting list as long as DBDH and DDH assumption hold.
- **Search security**: According to our search algorithm, in a complete search processes, the cloud server is able to obtain several intermediate computation results containing underlying keyword information such as $F_x = e(g,g)^{rH_1(w)q_x(0)}$, $F_R = e(g,g)^{rH_1(w)s}$, and $\frac{e(\tilde{I}''_w, T_1)}{F_R} = e(g,g)^{H_1(w)\alpha s}$ where x and R denote a non-leaf node and the root node of a access tree respectively, $r, q_x(0), s, \alpha$ are random elements in terms of the cloud server. Obviously, all of them are the encryption of E_1, which is semantically secure against chosen plaint-keyword attack as proved in Theorem 2. This means that



(a) Time cost of user grant for different number of attributes.



(b) Time cost of trapdoor generation for different number of attributes.

Fig. 5. Time cost of user grant and trapdoor generation.

it is difficult to recover keyword information from these intermediate results, which guarantees that the cloud server cannot obtain any useful information from search processes.

- **Trapdoor privacy and unlinkability**: Given the query trapdoor $\mathcal{T}_u(w)$ of query keyword w, the cloud server cannot recover w from each $T_a = g^{rH_1(w_0)}H_2(a)^{r_a}, a \in S$, where S is u's attribute set, due to the intractability of encryption block E_3 under the DDH assumption (Theorem 4), which guarantees query trapdoor privacy. Since E_3 is able to effectively defense chosen-plaintext keyword attack by introducing random element r, therefore it is a nondeterministic encryption, which achieves query trapdoor unlinkability property. That is, the identical query keywords bear totally different query trapdoors.

8. Experimental evaluation

We conduct experimental evaluation for the proposed scheme on Enron Email Dataset [11] and randomly select 4000 text files to build our experimental subset, from which 800 index keywords are extracted by using Hermetic Word Frequency Counter [36]. All programs are developed in Java language and leveraging JPBC library [16]. The elliptic curve group *Type A* [16] is used in our experiments. The client running configuration is a Windows 7 desktop system with 2.30 GHz Intel Core i5-6200 CPU, 4 GB memory and the server is an Ubuntu 16.04 system with 3.60 GHz Intel Core i7-7700 CPU, 16 GB memory. The client performs "Key Generation", "User Grant", "Index Keywords Encryption", "Trapdoor Generation" and server side performs search over encrypted data.

8.1. Time cost of user grant and trapdoor generation

A data user's search authorization is that the data owner assigns the legal query keyword encryption key to the data user according to the data user's system role. Fig. 5(a) shows that the time cost of user grant is linear to the number of user attributes. That is, the more attribute the data user has, the more time the data owner needs to spend to generate query keyword encryption key for the data user. Once the data user obtains the query key from data owner, he is able to generate legal query trapdoor with respect to query keyword. Moreover, the trapdoor generation time has nothing to do with the number of user's attributes, as shown Fig. 5(b). This is because that, in this process, only two relatively consuming-time exponentiation operations are involved regardless of how many the data user has attributes.

Table 5
The time cost of encrypting one index keyword for different number of attributes.

l	2	3	4	5	6	7	8	9
t	0.437	0.594	0.764	0.953	1.107	1.280	1.499	1.623

8.2. Time cost of secure index construction

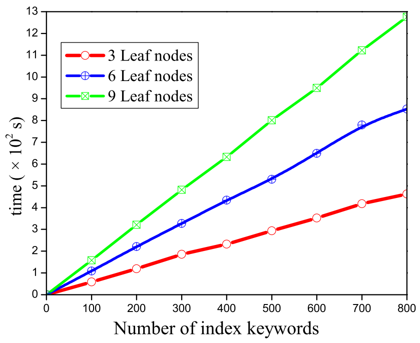
Secure index construction is to encrypt each index keyword with specified access tree to form corresponding encrypted posting list. Table 5 shows the time cost t (in seconds), of encrypting one index keyword when varying the number l, of leaf nodes (attributes) of the specified access tree. We can see that the time cost continually increases with the number of leaf nodes. When $l = 9$, the encryption time achieves about 1.623 s, it is a relatively time-consuming process.

To comprehensively evaluate the time cost of secure index construction over given data set, we conduct several groups of experiments for different scales of index keyword set and data file set, as shown Figs. 6(a) and 6(b). For ease of testing, each group experiment adopts the same access tree to encrypt index keywords, the number of leaf nodes in the access tree is 3, 6, and 9, respectively. From Fig. 6(a), we can see that the cost time of secure index construction linearly increases with the number of index keywords when fixing the size of the data file set. Fig. 6(b) shows the size of the data file set has little influence on secure index construction, which is a good property since the size of data file set is generally much larger than the number of index keywords in practice. In addition, the more leaf nodes (attributes) are in the access tree, the more time is spent to construct secure index.

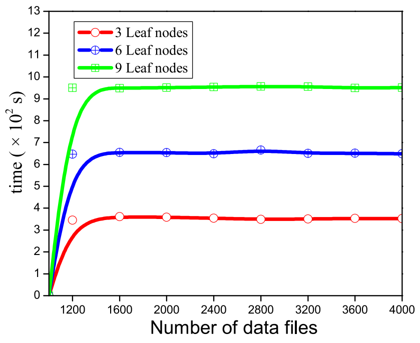
In conclusion, although the secure index construction is a relatively time-consuming process for the data owner, it is a one-time operation.

8.3. The time cost of secure index re-encryption

Upon receiving the secure index from the data owner, the cloud server needs to re-encrypt them for achieving fine-grained and efficient user attribute update. Fig. 7(a) shows that, when fixing the number of data files ($n = 2000$), the time cost of re-encrypting secure index for the cloud server linearly increases with the number of index keywords. Fig. 7(b) shows that, when fixing the number of index keywords ($u = 600$), the scale of the data files does not almost affect the time cost of secure index re-encryption. Similar to the secure index construction, the more leaf nodes (attributes) are in the access tree, the more time is spent to re-encrypt secure index for the cloud server, as shown in Figs. 7(a) and 7(b).

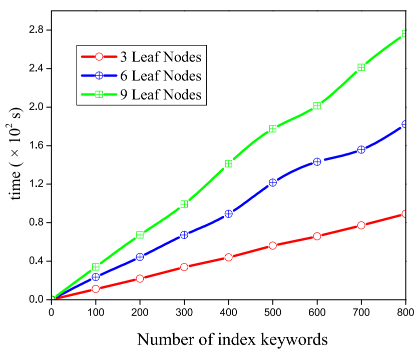


(a) For different number of index keywords with fixed scale of data files, n=2000.

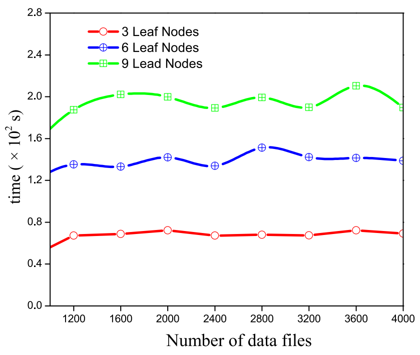


(b) For different number of data files with fixed scale of index keywords, u=600.

Fig. 6. Time cost of secure index construction for the data owner.



(a) For different number of index keywords with fixed scale of data files, n=2000.



(b) For different number of data files with fixed scale of index keywords, u=600.

Fig. 7. Time cost of secure index re-encryption for the cloud server.

8.4. The time cost of search

In order to clearly demonstrate the relationship between the search time and the number l of leaf nodes in the access tree and the number s of data user's attributes, we adopt a simple access tree construction with only one interior node to encrypt an index keyword as shown in Fig. 8, where $a_1, \ldots, a_l$ denotes l leaf nodes (each corresponds to an attribute) and $1 \leq m \leq l$. Obviously, a data user has a set of attributes S, if $|S \cap L| = m$, then the data user has the query permission of the index keyword encrypted by the access tree, where $L = \{attr(a_1), \ldots, attr(a_l)\}$, $|S| = s$, and $|L| = l$.

Figs. 9(a) and 9(b) show the time cost of search over one encrypted posting list when using different values of m, l, and s. Experimental results demonstrate that, when the data user's attributes satisfy the access tree (i.e., $|S \cap L| = m$), the search time is only determined by value m and irrelevant to l and s. The larger the value m is, the more time needs to be spent for the cloud server. In addition, when the data user's attributes do not satisfy the access tree (i.e., $|S \cap L| \neq m$), the search time is almost zero, since, in this case, no time-consuming computation is involved such as pairing operation and exponential operation.

Fig. 10 shows the average time cost of search in our experimental data set. Fig. 10(a) shows that, when fixing the number of data files ($n = 2000$), the time cost of search increases linearly with an increasing number of index keywords. While keeping the number of index keywords unchanged ($u = 400$), the number of data files have no influence on search time, as

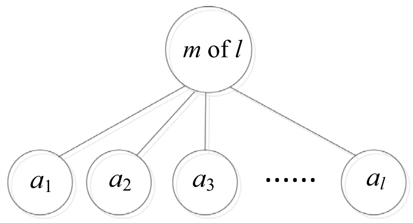
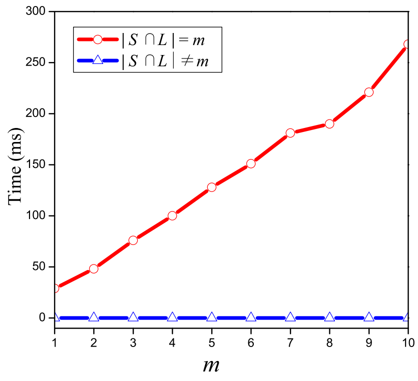


Fig. 8. Access tree construction used to encrypt an index keyword in the experiments.

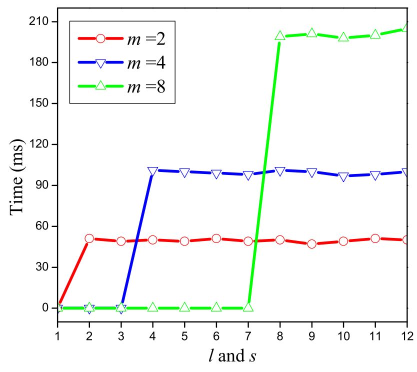
shown in Fig. 10(b). We also observe that the time cost can be reduced remarkably by increasing the number of threads when the programs are run in a parallel computing environment. Note that, in our experiments, we assume the data user holds all index keywords' query permissions. Therefore, in practical application, the actual search time cost should be less than our experimental results. An extreme case is that when a data user's attributes do not satisfy all access trees, the search time cost is almost zero regardless of the number of index keywords.

8.5. The time cost of search permission update

A user's search permission update would take place when attributes are added/revoked to/from the user. The processes mainly rely on our designed attribute addition protocol and attribute revocation protocol. Therefore, we evaluate the time cost

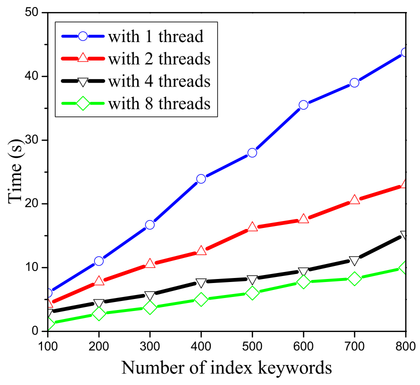


(a) For different number of m with fixed l and s, $l = 10$ and $s = 10$.

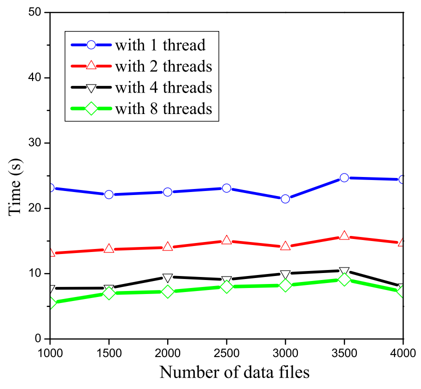


(b) For different number of l and s with fixed number of m.

Fig. 9. Time cost of search over one encrypted posting list.



(a) For different number of index keywords (posting lists) with fixed number of data files (n=2000), $l = s = 10$ and $m = 5$.



(b) For different number of data files with fixed number of index keywords (u=400), $l = s = 10$ and $m = 5$.

Fig. 10. Average time of search over experimental data set.

of search permission update by running these two sub-protocol implementations. Table 6 demonstrates each party's time cost (in seconds) of running designed attribute revocation protocol, where a denotes a revoked attribute from a data user u and n_a denotes the total number of encrypted posting lists, whose access trees contain attribute a, stored at the cloud server. We can see that, in the process of performing the protocol, the time cost at the cloud server side linearly increases with n_a and each other data user except u spends about 0.047 s to complete the protocol, while the execute time at the data owner side is almost zero. Table 7 shows the time cost (in seconds) of attribute addition protocol, where b denotes an added attribute to a data user u and n_b denotes the total number of encrypted posting lists, whose access trees contain attribute b, stored at the cloud server. We can see that it is an extremely efficient protocol, the time cost occurrence is only at the data owner side, who spends about 0.094 s to complete the protocol regardless of n_b.

Table 6
The time cost of running attribute revocation protocol.

n_a	100	200	300	400	500	600
Data owner	0	0	0	0	0	0
Cloud server	1.02	1.762	2.656	3.289	4.112	4.983
Data user	0.047	0.048	0.047	0.047	0.045	0.047

Table 7
The time cost of running attribute addition protocol.

n_b	100	200	300	400	500	600
Data owner	0.094	0.095	0.095	0.092	0.094	0.095
Cloud server	0	0	0	0	0	0
Data user	0	0	0	0	0	0

9. Conclusion

In this paper, we propose a fine-grained authorized keyword secure search scheme over encrypted cloud data by evolving the off-the-peg CP-ABE scheme proposed in [2]. Our scheme allows the data owner to flexibly grant keyword search permissions with different data users in a fine-grained manner by embedding different access control policies into different index keywords. Moreover, our scheme supports efficient and fine-grained user search permission update by two designed attribute update sub-protocols. Finally, we valuate the performance for our scheme by extensive experiments in the real data set. How to hide access tree construction and user's attributes in our search system to achieve full privacy-protection will be our future work.

Acknowledgments

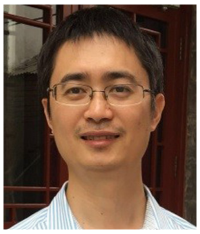
The work is supported by the National Natural Science Foundation of China under Grant Nos. 61972058, 61902123, 61772088, and 61772191; Science and Technology Key Projects of Hunan Province under Grant No. 2016JC2012; Outstanding Youth Research Project of Provincial Education Department of Hunan under Grant No. 17B030; Natural Science Foundation of Hunan Province under Grant No. 2017JJ3371.

Declaration of competing interest

No author associated with this paper has disclosed any potential or pertinent conflicts which may be perceived to have impending conflict with this work. For full disclosure statements refer to https://doi.org/10.1016/j.jpdc.2019.09.011.

References

[1] L. Ballard, S. Kamara, F. Monrose, Achieving efficient conjunctive keyword searches over encrypted data, in: IEEE ICICS, 2005, pp. 414–426.
[2] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: IEEE S&P, 2007, pp. 321–334.
[3] D. Boneh, G.D. Crescenzo, R. Ostrovsky, G. Persiano, Public-key encryption with keyword search, in: EUROCRYPR, 2004, pp. 506–522.
[4] D. Boneh, B. Waters, Conjunctive, subset, and range queries on encrypted data, in: Spinger TCC, 2007, pp. 535–554.
[5] N. Cao, C. Wang, M. Li, K. Ren, W. Lou, Privacy-preserving multi-keyword ranked search over encrypted cloud data, in: IEEE INFOCOM, 2011, pp. 829–837.
[6] N. Cao, C. Wang, M. Li, K. Ren, W. Lou, Privacy-preserving multi-keyword ranked search over encrypted cloud data, IEEE Trans. Parallel Distrib. Syst. 25 (1) (2014) 222–233.
[7] N. Cao, S. Yu, Z. Yang, W. Lou, Y. Hou, LT codes-based secure and reliable cloud storage service, in: IEEE INFOCOM, 2012, pp. 693–701.
[8] Y.-C. Chang, M. Mitzenmacher, Privacy Preserving Keyword Searches on Remote Encrypted Data, in: ACNS, 2005, pp. 442–455.
[9] H. Cui, Z. Wan, R. Deng, G. Wang, Y. Li, Efficient and expressive keyword search over encrypted data in cloud, IEEE Trans. Dependable Secure Comput. 15 (3) (2018) 409–422.
[10] R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky, Searchable symmetric encryption: improved deinitions and efficient constructions, in: ACM CCS, vol. 19, 2006, pp. 79–88.
[11] Enron dataset, http://nlp.cs.aueb.gr/software.
[12] Z. Fu, K. Ren, J. Shu, X. Sun, F. Huang, Enabling personalized search over encrypted outsourced data with efficiency improvement, IEEE Trans. Parallel Distrib. Syst. 27 (9) (2016) 2546–2559.
[13] E.-J. Goh, Secure Indexes, Tech. rep., 2003, IACR ePrint Cryptography Archive, http://eprint.iacr.org/2003/216.
[14] P. Golle, J. Staddon, B. Waters, Secure conjunctive keyword search over encrypted data, in: Springer ACNS, 2004, pp. 31–45.
[15] V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encryption data, in: ACM CCS, 2006, pp. 89–98.
[16] http://gas.dia.unisa.it/projects/jpbc/index.html.
[17] S. Kamara, K. Lauter, Cryptographic cloud storage, in: Springer RLCPS, 2010, pp. 136–149.
[18] S. Kamara, C. Papamanthou, Parallel and dynamic searchable symmetric encryption, in: Financial Cryptography and Data Security, Springer Berlin Heidelberg, 2013, pp. 258–274.
[19] S. Kamara, T. Roeder, C. Papamanthou, Dynamic searchable symmetric encryption, in: ACM CCS, 2012, pp. 965–976.
[20] K.S. Kim, M. Kim, B. Waters, D. Lee, J.H. Park, W.H. Kim, Forward secure dynamic searchable symmetric encryption with efficient updates, in: ACM CCS, 2017, pp. 1449–1463.
[21] K. Kurosawa, Y. Ohtaki, UC-secure searchable symmetric encryption, in: Financial Cryptography and Data Security, Springer Berlin Heidelberg, 2012, pp. 285–298.
[22] J. Li, Q. Wang, N. Cao, K. Ren, W. Lou, Fuzzy keyword search over encrypted data in cloud computing, in: IEEE INFOCOM, 2010.
[23] H. Li, Y. Yang, T. Luan, X. Liang, L. Zhou, X. Shen, Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data, IEEE Trans. Dependable Secure Comput. 13 (3) (2016) 312–325.
[24] K. Liang, W. Susilo, Searchable attribute-based mechanism with efficient data sharing for secure cloud storage, IEEE Trans. Inf. Forensics Secur. 10 (9) (2015) 1981–1992.
[25] Z. Lv, C. Hong, M. Zhang, D. Feng, Expressive and secure searchable encryption in the public key setting, in: International Information Security Conference, 2014, pp. 364–376.
[26] Y. Miao, J. Ma, X. Liu, X. Li, J. Q, J. Zhang, Attribute-based keyword search over hierarchical data in cloud computing, IEEE Trans. Serv. Comput. (2017) doi: 10.1109/TSC.2017.2757467.
[27] T. Peng, L. Qin, B. Hu, J. Liu, J. Zhu, Dynamic keyword search with hierarchical attributes in cloud computing, IEEE Access 6 (2018) 68948–68960.
[28] Y. Peng, W. Zhao, F. Xie, Z. Dai, Y. Gao, D. Chen, Secure cloud storage based on cryptographic techniques, J. China Univ. Posts Telecommun. 19 (11) (2012) 182–189.
[29] K. Ren, C. Wang, Q. Wang, Security challenges for the public cloud, IEEE Internet Comput. 16 (1) (2012) 69–73.
[30] A. Sahai, B. Waters, Fuzzy identity-based encryption, in: EUROCRYPT, 2005, pp. 457–473.
[31] J. Shi, J. Lai, Y. Liu, R.H. Deng, J. Weng, Authorized keyword search on encrypted data, in: ESORICS, 2014, pp. 419–435.
[32] D. Song, D. Wagner, A. Perrig, Practical techniques for searches on encrypted data, in: IEEE Symposiumon Security and Privacy, vol. 8, 2000, pp. 44–55.
[33] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y.T. Hou, Verifiable privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking, IEEE Trans. Parallel Distrib. Syst. 25 (11) (2014) 3025–3035.
[34] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y.T. Hou, H. Li, Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking, in: ASIA CCS, 2013, pp. 71–82.
[35] W. Sun, S. Yu, W. Lou, Y.T. Hou, H. Li, Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud, in: IEEE INFOCOM, 2014, pp. 226–234.
[36] H. Systems, Hermetic word frequency counter, http://www.hermetic.ch/wfc/wfc.htm.
[37] L.M. Vaquero, L. Rodero-Merino, J. Caceres, M. Lindner, A break in the clouds: Towards a cloud definition, ACM SIGCOMM Comput. Commun. Rev. 39 (1) (2009) 50–55.
[38] C. Wang, N. Cao, J. Li, K. Ren, W. Lou, Secure ranked keyword search over encrypted cloud data, in: IEEE ICDCS, 2010, pp. 253–262.
[39] C. Wang, N. Cao, K. Ren, W. lou, Enabling secure and efficient ranked keyword search over outsourced cloud data, IEEE Trans. Parallel Distrib. Syst. 23 (8) (2012) 1467–1479.
[40] H. Wang, X. Dong, Z. Cao, Multi-value-independent ciphertext-policy attribute based encryption with fast keyword search, IEEE Trans. Serv. Comput. (2017) http://dx.doi.org/10.1109/TSC.2017.2753231.
[41] S. Wang, J. Ye, Y. Zhang, Searchable attribute-based encryption scheme with attribute revocation in cloud storage, PloS One 12 (8) (2018) e0183459.
[42] B. Wang, S. Yu, W. Lou, Y.T. Hou, Privacy-preserving multi-keyword fuzzy search over encrypted data in the cloud, in: IEEE INFOCOM, 2014, pp. 2112–2120.
[43] Z. Xia, X. Wang, X. Sun, Q. Wang, A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data, IEEE Trans. Parallel Distrib. Syst. 27 (2) (2016) 340–352.
[44] H. Yin, Z. Qin, L. Ou, K. Li, A query privacy-enhanced and secure search scheme over encrypted data in cloud computing, J. Comput. System Sci. 90 (2017) 14–27.
[45] H. Yin, Z. Qin, J. Zhang, L. Ou, F. Li, K. Li, Secure conjunctive multi-keyword ranked search over encrypted cloud data for multiple data owners, Future Gener. Comput. Syst. 100 (2019) 689–700.
[46] H. Yin, J. Zhang, Y. Xiong, L. Ou, F. Li, S. Liao, K. Li, CP-ABSE: A ciphertext-policy attribute-based searchable encryption scheme, IEEE Access 7 (1) (2019) 5682–5694.
[47] Q. Zheng, S. Xu, G. Ateniese, Vabks: Verifiable attribute-based keyword search over outsourced encrypted data, in: IEEE INFOCOM, 2014, pp. 522–530.



Hui Yin received his BS in Computer Science from Hunan Normal University, China, in 2002; MS in Computer Software and Theory from Central South University, China, in 2008; and his PhD degree from the College of Information Science and Engineering at Hunan University, China, in 2018. He is currently an assistant professor at the College of Applied Mathematics and Computer Engineering, Changsha University, China. His interests are information security, privacy protection, and applied cryptography.

Zheng Qin received the PhD degree in computer software and theory from Chongqing University, China, in 2001. He is a professor of computer science and technology in Hunan University, China. He is a member of China Computer Federation (CCF) and ACM respectively. His main interests are computer network and information security, cloud computing, big data processing and software engineering. He has accumulated rich experience in products development and application services, such as in the area of financial, medical, military and education sectors.

JiXin Zhang received the BS degree from Hubei Polytechnic University in Mathematics, and the MS degree from Wuhan University of Technology in computer science and technology, in 2007 and 2014, respectively. Currently, he is pursuing the PhD degree in the College of Information Science and Engineering at Hunan University, China. His research focuses on polymorphic malware detection, privacy protection and big data.

Hua Deng received his MS degree in cryptography from Southwest Jiaotong University, China, in 2010 and Ph.D. degree in information security from Wuhan University, China, in 2015. Now he is a postdoctoral research fellow at the College of Computer Science and Electronic Engineering, Hunan University, China. His research interests include applied cryptography, data security and privacy, cloud security.

Fangmin Li received the B.Sc. degree from the Huazhong University of Science and Technology, Wuhan, China, in 1990, the M.Sc. degree from the National University of Defense Technology, Changsha, China, in 1997, and the Ph.D. degree from Zhejiang University, Hangzhou, China, in 2001, all in computer science. He is currently a Professor with the School of Computer Engineering and Applied Mathematics, Changsha University. He has authored several books on embedded systems and over 50 academic papers in wireless networks, and also holds ten Chinese patents. His current research interests include wireless communications and networks security, computer systems and architectures, and embedded systems. Dr. Li is a Senior Member of China Computer Federation (CCF), and a Committee Member of the Technical Committee on Sensor Network, CCF.

Keqin Li is a SUNY Distinguished Professor of computer science with the State University of New York. He is also a Distinguished Professor at Hunan University, China. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber-physical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU-GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He has published over 680 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He currently serves or has served on the editorial boards of the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Computers, the IEEE Transactions on Cloud Computing, the IEEE Transactions on Services Computing, and the IEEE Transactions on Sustainable Computing. He is an IEEE Fellow.