Received May 12, 2020, accepted May 19, 2020, date of publication May 25, 2020, date of current version June 8, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2996940

An Efficient Attribute-Based Multi-Keyword Search Scheme in Encrypted Keyword Generation

YUANBO CUI[1], FEI GAO[1], YIJIE SHI[1], WEI YIN[2], EMMANOUIL PANAOUSIS[3], (Member, IEEE), AND KAITAI LIANG[3], (Member, IEEE)

[1]State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China
[2]National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100020, China
[3]Department of Computer Science, University of Surrey, Guildford GU2 7XH, U.K.

Corresponding author: Yijie Shi (yijieshi2000@bupt.edu.cn)

This work was supported in part by the National Natural Science Foundation of China (NSFC) under Grant 61672110, Grant 61671082, Grant 61976024, and Grant 61972048, and in part by the Fundamental Research Funds for the Central Universities under Grant 2019XD-A01.

ABSTRACT With the growing popularity of cloud computing in recent years, data owners (DOs) now prefer to outsource their data to cloud servers and allow the specific data users (DUs) to retrieve the data. Searchable encryption is an important tool to provide secure search over the encrypted cloud data without infringing data confidentiality and data privacy. In this work, we consider a secure search service providing fine-grained and search functionality, called attribute-based multiple keyword search (ABMKS), which can be seen as an extension of searchable encryption. In the existing ABMKS schemes, the computation operations in the encrypted keyword index generation are time-consuming modular exponentiation, and the number of which is linearly growing with the factor m. Here m is the number of keywords embedded in a file. To reduce the computation overhead, in this paper, we propose an ABMKS with only multiplication operations in encrypted keyword index generation. As a result, the computation cost of the encrypted keyword index generation is more efficient than the existing schemes. In addition, the encrypted keyword indexes are aggregated into one item, which is regardless of the number of underlying keywords in a file data. Finally, the security and the performance analysis demonstrate that our scheme is both efficient and secure.

INDEX TERMS Searchable encryption, modular exponentiation, multiplication, attribute-based multiple keywords search, cloud computing.

I. INTRODUCTION

With the flexibility and benefits provided by the cloud storage [1], [2] and cloud computing [3], [4], data owners (DOs) prefer to outsource the management of their data to the cloud service provider (CSP) and rent the strong computation ability of CSP. Because DOs pay more attention and consideration to the privacy of data, DOs will encrypt their sensitive data before outsourcing it to the CSP. However, data encryption will cause a huge cost in terms of data usability, as the existing solutions of keyword-based information retrieval on plaintext data cannot be applied directly to the encrypted data. It is impractical to download all the encrypted data from CSP and decrypt it locally.

To realize the keyword search over encrypted data, some solutions are proposed using fully-homomorphic encryption [5] or oblivious RAMs [6], but these techniques will bring huge computation overhead on both CSP and users. On the contrary, searchable encryption [7] is a practical solution, which allows the CSP to search over encrypted data on behalf of the authorized users with a keyword search trapdoor provided by the users, and the CSP can retrieve the matching data without learning information about the underlying plaintext.

Searchable encryption (SE) can be realized in both symmetric and asymmetric encryption settings. Abound of research works [8], [9], [11]–[14] have been proposed to realize various search functionalities, such as single keyword search, multiple keywords search, ranked search etc. In the symmetric searchable encryption (SSE) schemes, DOs have to distribute a session key to DUs, which brings complicated secret key distribution/management overhead to DOs.

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

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/

Boneh *et al*. [15] first introduced the definition of searchable encryption in public-key setting, which can resolve the issue of distributing the session key in the SSE. Zheng *et al*. [16] proposed new primitive called attribute-based keyword search (ABKS) by integrating SE and attribute-based encryption [17]–[24]. ABKS is extended on basis of SE to realize keyword search and access control simultaneously. Follow-up [25], [26] are proposed to achieve multiple keyword search based on [16]. Among the public-key encryption with keyword search schemes, attribute-based multi-keyword search (ABMKS) achieves more and more attention for its practical applicability [25].

In ABMKS, DOs can realize the access control on their data, which means only the authorized DUs can access to it. More precisely, DOs encrypt their data based on an access policy with attribute-based encryption and build encrypted keyword indexes corresponding to the keyword extracted from the data. If the data users' attributes satisfy the access policy and the trapdoor maths with the encrypted keyword indexes simultaneously, they can retrieve and decrypt the matching data. The existing ABMKS scheme [26] can support keyword search and comparable attributes through utilizing 0-encoding and 1-encoding. However, in [26], the computation operations in the encrypted keyword index generation are mainly modular exponentiations, which are time-consuming compared to multiplication.

In this paper, we design an efficient ABMKS scheme, called attribute-based multiple keyword search scheme with only multiplication (ABMKS-WM) in encrypted keyword generation by using Binary Vector and Polynomial (the details of them shown as in Sections III-C-III-D), thus the time cost of the encrypted keyword index generation is more efficient compared to existing schemes, e.g., [25]–[27]. In addition, the encrypted keyword indexes are aggregated into one item, which is regardless of the number of underlying keywords in a file. The contribution of this paper are summarized as follows:

1) We design a secure ABMKS without exponentiation in the encrypted keyword index generation. Unlike the existing ABMKS schemes, e.g., [25]–[27], the computation operations in index generation are only multiplication, which is more efficient than exponentiation. In addition, the encrypted keyword indexes are aggregated into one item, which is regardless of the number of underlying keywords in a file. To our best knowledge, our design is the first of its type that achieves the encrypted keyword index generation without exponentiation in the model of ABMKS.
2) We prove that our design is secure against the chosen-keyword attacks via the formal security analysis, and our performance evaluation proves that the scheme is efficient in terms of both the computation and communication overhead, in particular, the time cost of the ciphertext generation and data retrieval are more efficient than that of the existing ABMKS schemes.

The remainder of this paper is organized as follows. We briefly review related work in Section 2. In Section 3, we present some basic primitives used in this paper and the main building blocks for our construction. We describe the system and threat models, the construction, and the security model of our scheme in Section 4. In Section 5, we give the design of our scheme. Section 6 presents the security analysis of the proposed scheme. The experimental analysis and the comparison with some related works are presented in Section 7. We conclude the paper in Section 8.

II. RELATED WORK

Searchable encryption enables DOs or DUs to execute keyword search over encrypted data. Based on the different cryptography primitives, searchable encryption can be roughly classified into symmetric SE and asymmetric ones.

Song *et al*. [7] first proposed the notion of symmetric SE and presented a construction of it. Subsequently, a variety of symmetric SE schemes have been proposed. Goh [28] put forward the security definitions for SSE and proposed a construction based on Bloom filter. Bao *et al*. [10] proposed a SSE scheme in multi-user model. Boneh *et al*. [15] first put forward the notion of public-key encryption with keyword search (PKES), and Abdalla *et al*. [30] enhance the foundations of PEKS. Abundant of PEKS schemes [31]–[38] are proposed to achieve various functionalities. Li *et al*. [38] proposed a scheme which supports both abilities and provides flexible keyword update service. Yin *et al* [39] proposed an ingenious secure query scheme to guarantee data security and system flexibility in the multiple data owners model, which allows each DO to adopt randomly chosen temporary keys to build secure indexes for different files. By leveraging the attribute-based encryption primitive, Yin *et al*. [41] put forward a fine-grained authorized keyword secure search scheme in which the access policy supports AND, OR, and threshold gates and Yin *et al*. [42] proposed a ABE scheme allows the DO to conduct a fine-grained search authorization for a DU. Liang and Susilo [40]present an ABE with keyword search schemes which allow DO to warrant keyword search capability to authorized DU.

Chen *et al*. [36] proposed a public key encryption with keyword search in dual-server model, which can resist the inside keyword guessing attack [43]. In order to check the correctness of retrieving results from the semi-trusted CSP, Liu *et al*. [35] and Miao *et al*. [26] presented the verifiable SE schemes, respectively.

To realize the access control and keyword search on the data at the same time, Zheng *et al*. [16] proposed a new search service called attribute-based keyword search (ABKS), which formed by combining of attribute-based encryption (ABE) and SE. And later [25], [26] were proposed to achieve extended functionalities based on [16]. For example, Miao *et al*. [25] proposed an ABKS with user revocation in multi-owner settings.

Golle *et al*. [29] proposed the concept of conjunctive keyword search in the SE system. Later on, Park *et al*.

[44] extended the notion into public key system. Attribute-Based Multiple Keyword Searchable supports multi-keyword search and access control on encrypted data simultaneously. However, in most existing ABMKS schemes, e.g., [25], [26], the computation operations in encrypted keyword index generation are time-consuming exponentiation, and the number of which is growing linearly with the factor m. Here m is the number of keyword in a file. Li *et al.* [27] proposed Towards Privacy-Preserving and Efficient Attribute-Based Multi-Keyword Search (TPPE-ABMKS) through utilizing keyword dictionary tree and the subset cover, which can achieve multi-keyword search with fine-grained access control, and the number of the encrypted keyword index is relatively small. However, the computation operations in the encrypted keyword index generation are pairing operations, which are also time-consuming.

III. PRELIMINARY

We present a brief review of some basic primitives used in this work in III-A and III-B. We also define the main building blocks for constructing our scheme in Sections III-C and III-D.

A. BILINEAR MAP

As in [17], [18], we let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p, g a generator of the group $\mathbb{G}$ and a bilinear mapping e: $\mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ satisfies following properties:

- Bilinearity: $e(g^a, g^b) = e(g, g)^{ab}, \forall g \in \mathbb{G}, \forall a, b \in \mathbb{Z}_p^*$.
- Nondegeneracy: $e(g, g) \neq 1$.
- Computability: the $e(g^a, g^b)$ can be computed by an efficient algorithm, $\forall g \in \mathbb{G}, \forall a, b \in \mathbb{Z}_p^*$

B. ACCESS TREE $\mathcal{T}$

Let $\mathcal{T}$ be a tree representing an access policy [17], [18]. In an access policy $\mathcal{T}$, each non-leaf of $\mathcal{T}$ represents a threshold gate, each non-leaf is described by its children and a threshold value. Let num_v denote the number of children of a node v, and the children from left to right are labeled as $1, \ldots, num_v$, while k_v ($k_v \leq num_v$) denotes the threshold value associated with the node v, when $k_v = 1$, the threshold gate is an OR gate and when $k_v = num_v$, it is an AND gate. Each leaf node of $\mathcal{T}$ is described by an attribute and a threshold value $k_v = 1$.

To better understand the access tree, we define a few functions as follow, let parent(v) represent the parent of node v. If v is a leaf node, att(v) represents the attribute associated with the leaf node v. Let index(v) denote the label of the node v, and $\mathcal{T}_v$ represent the subtree of $\mathcal{T}$ rooted at node v.

Let $\mathcal{T}_v(\gamma) = 1$ indicate that the attribute set γ satisfies the access tree $\mathcal{T}_v$. If v is a non-leaf node, we can compare $\mathcal{T}_v(\gamma)$ recursively as follows, compute $\mathcal{T}_{v'}(\gamma)$ for all children v' of the node v, if at least k_v children of the node v return 1, then $\mathcal{T}_v(\gamma) = 1$. If v is a leaf node and $att(v) \in \gamma$, then $\mathcal{T}_v(\gamma) = 1$.

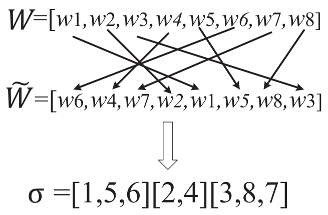


FIGURE 1. **An example of permutation σ Generation.**

C. KEYWORD DICTIONARY AND BINARY VECTOR

Let $W = [w_1, \ldots, w_n]$ be the keywords dictionary in the system, and W_Q be the query keywords set chosen by DU, e.g., $W_Q = \{w_1, w_3, w_5, w_7\}$. We represent W_Q with a **binary vector** $\vec{Q}$ based on the keyword dictionary W as follows:

$$\vec{Q} = [q_1, \ldots, q_n]$$

where

$$\begin{cases} q_i = 1 : w_i \in W_Q \\ q_i = 0 : w_i \notin W_Q \end{cases}$$

For easy to understand, hereafter, we set $n = 8$ in all the examples, e.g., when $W = [w_1, \ldots, w_8]$ and $W_Q = \{w_1, w_3, w_5, w_7\}$, then $\vec{Q} = [1, 0, 1, 0, 1, 0, 1, 0]$.

Let W_D be a keywords set that appears in a file, we represent W_D with a **binary vector** $\vec{D}$ based on the keyword dictionary W as follows:

$$\vec{D} = [d_1, \ldots, d_n]$$

where

$$\begin{cases} d_i = 1 : w_i \in W_D \\ d_i = 0 : w_i \notin W_D \end{cases}$$

For example, when $W = [w_1, \ldots, w_8]$ and $W_D = \{w_1, w_3, w_5, w_6, w_7\}$, then $\vec{D} = [1, 0, 1, 0, 1, 1, 1, 0]$.

Definition 1: Given the **binary vectors** *$\vec{Q}$ and $\vec{D}$, if for all $i = 1$ to n, $q_i \leq d_i$, we write $\vec{Q} \subseteq \vec{D}$, it is to say that $W_Q \subseteq W_D$.*

D. MAIN IDEA

1) INDEX GENERATION

Before building the encrypted keyword indexes for the keywords set W_D of a file $\widehat{f}$, DO first randomizes the order of keyword in the keyword dictionary W to get the new keyword dictionary $\widetilde{W} = [\widetilde{w}_1, \ldots, \widetilde{w}_n]$, then obtains the permutation σ.

We give an example of permutation generation as shown in Fig. 1, $\sigma = [1, 5, 6][2, 4][3, 8, 7]$, where $[1, 5, 6]$ means $\widetilde{w}_5 = w_1$, $\widetilde{w}_6 = w_5$, $\widetilde{w}_1 = w_6$, it's to say that the keyword w_1 in W is shifted to the 5-th position of $\widetilde{W}$, keyword w_5 in W is shifted to the 6-th position of $\widetilde{W}$, keyword w_6 in W is shifted to the 1-th position of $\widetilde{W}$. $[2, 4]$ and $[3, 8, 7]$ follow the same rule as [1, 5, 6], and where [2, 4] means $\widetilde{w}_4 = w_2$, $\widetilde{w}_2 = w_4$; where [3, 8, 7] means $\widetilde{w}_8 = w_3$, $\widetilde{w}_7 = w_8$ and $\widetilde{w}_3 = w_7$. We can see that the total number of possible σ is $n! = n \times (n-1) \times \ldots \times 2 \times 1$. As a result, an adversary has the probability of $1/n!$ to guess it, e.g., when $n = 500$, then the probability is negligible. The DO will generate the encrypted keyword indexes for the keywords set W_D with $\widetilde{W}$ and σ as follows:

First, as shown in Section III-C, DO will generate a binary vector $\overrightarrow{D} = [d_1, \ldots, d_n]$ for W_D based on the new keyword dictionary $\widetilde{W}$. For example, when $\widetilde{W} = [w_6, w_4, w_7, w_2, w_1, w_5, w_8, w_3]$, $W_D = \{w_1, w_3, w_5, w_6, w_7\}$, then $\overrightarrow{D} = [1, 0, 1, 0, 1, 1, 0, 1]$ and the permutation $\sigma = [1, 6, 5][2, 4][3, 7, 8]$.

Second, with the binary vector $\overrightarrow{D} = [d_1, \ldots, d_n]$, compute the function $f(x, \overrightarrow{D})$ and the encrypted keyword index I for the keywords set W_D as follows:

$$f(x, \overrightarrow{D}) = \prod_{j=1}^{n} (x - H(\sigma||j))^{d_j} \tag{1}$$

$$I = f(x, \overrightarrow{D})_{|x=H(Encode(\sigma||K))} \tag{2}$$

where H is hash function, K is a random symmetric key which is used to encrypt the file $\widehat{f}$. Note that the operations in $I = f(x, \overrightarrow{D})_{|x=H(Encode(\sigma||K))}$ is multiplication but not modular multiplication.

At last, DO chooses an access policy tree $\mathcal{T}$ and encrypts the permutation σ and K based on the access policy tree by using ABE scheme.

2) TRAPDOOR GENERATION

If a data user's attributes set satisfies the access policy tree $\mathcal{T}$, then he/she can get the permutation σ and the symmetric key K. Then the DU can generate the new keyword dictionary $\widetilde{W}$ by using the σ.

As shown in Section III-C, let W_D be the query keywords set chosen by a DU, and DU then generates the binary vector $\overrightarrow{Q} = [q_1, \ldots, q_n]$ for W_Q based on the new keyword dictionary $\widetilde{W}$. For example $\widetilde{W} = [w_6, w_4, w_7, w_2, w_1, w_5, w_8, w_3]$, $W_Q = \{w_1, w_3, w_5, w_7\}$, then $\overrightarrow{Q} = [0, 0, 1, 0, 1, 1, 0, 1]$.

Then, DU can compute the function $f(x, \overrightarrow{Q})$ and the trapdoor T corresponding to keywords set W_Q as follows:

$$f(x, \overrightarrow{Q}) = \prod_{j=1}^{n} (x - H(\sigma||j))^{q_j} \tag{3}$$

$$T = f(x, \overrightarrow{Q})_{|x=H(Encode(\sigma||K))} \tag{4}$$

where H is a hash function, K is a random symmetric key which is used to encrypt the file $\widehat{f}$. Note that the operation in $T = f(x, \overrightarrow{Q})_{|x=H(Encode(\sigma||K))}$ is multiplication but not modular multiplication.

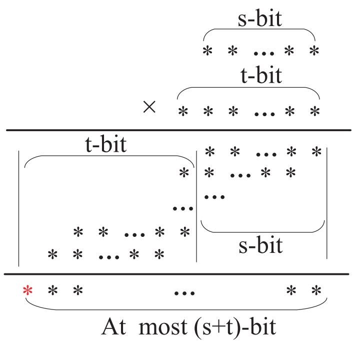


FIGURE 2. s-bit multiply by t-bit.

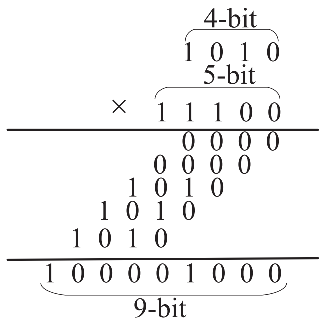


FIGURE 3. An example of 5-bit multiply by 4-bit.

3) MATCH PROCESS

If $W_Q \subseteq W_D$, then $\overrightarrow{Q} \subseteq \overrightarrow{D}$, thus for all $i = 1$ to n, $q_i \leq d_i$. We can note that $\frac{f(x,\overrightarrow{D})}{f(x,\overrightarrow{Q})} = \prod_{j=1}^{n} (x - H(\sigma||j))^{d_j - q_j}$ is a polynomial function in x. For that, $\frac{I}{T} = \frac{f(x,\overrightarrow{D_i})}{f(x,\overrightarrow{Q})}{}_{|x=H(Encode(\sigma||K))}$ is a integer, it means that I can be exactly divisible by T, then $I \ mod \ T = 0$.

Remark: Based on the method above to generate encrypted keyword indexes, the sizes of the encrypted keyword indexes and the trapdoor are aggregated into one item. In addition, the computation operations in encrypted keyword index and the trapdoor generation are multiplications. As described above, the encrypted keyword index is I, which is a large number. We give the theorem of the bit long for the I as follow.

Theorem 1: Let H be the hash function $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p^*$, the bit length of $I = f(x, \overrightarrow{D})_{|x=H(Encode(\sigma||K))}$ is at most $m \cdot |\mathbb{Z}_p^*|$, here m is the number of the keyword in W_D, $|\mathbb{Z}_p^*|$ is the bit length of an element in $\mathbb{Z}_p^*$.

Proof: First, as shown in the Fig. 2, we prove that if a s-bit number multiply by a t-bit number, then the bit length of the product is at most s+t.

The Fig. 2 describes the rule of the binary multiplication, the $*$ denotes 0 or 1. We give a example of the 4-bit number (1010) multiply by a 5-bit number (11100) in Fig. 3, the computing rule of binary multiplication is the same as the decimal

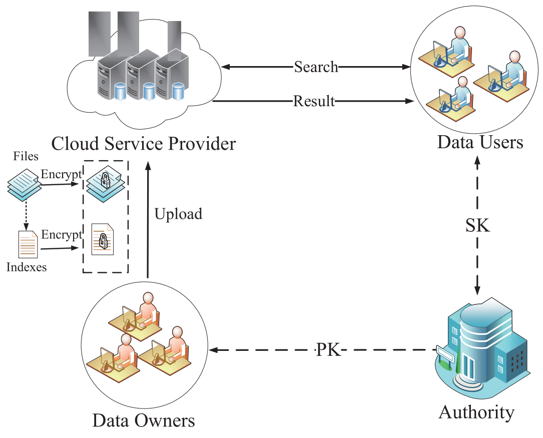


FIGURE 4. The basic framework of our scheme.

multiplication, the product of (0×0) is 0, the product of (1×1) is 1, the product of (0×1) is 0, we can see that the bit length of the product of (4-bit) and (5-bit) is at most 9-bit. As the computing rule in the Fig. 3, we can note that the bit length of product of s-bit number $\times$ t-bit number is at most s+t when the red $*$ is 1.

Note that I is the product of m $|\mathbb{Z}_p^*|$-bit numbers multiply continuously, then the bit length of I is at most $m\cdot|\mathbb{Z}_p^*|$. $\square$

IV. PROBLEM FORMULATIONS

In this section, we present the system model, the threat model, the scheme definition, and the security model, respectively.

A. SYSTEM AND THREAT MODEL

We present the system model of our scheme in Fig. 4, which is the same as the one in [25]. Our scheme involves four entities: DO, DU, CSP and Authority. The DO will encrypt the data files set F as well as corresponding keyword sets with an access policy before uploading them to the CSP. The CSP provides the storage services and executes keyword search operations on behalf of the DU. When a DU wants to make a search query over the encrypted data, he/she generates a trapdoor by his/her specified query keywords and submits it to CSP. On receiving the trapdoor, CSP retrieves the appropriate data file by using the trapdoor, if the user's attributes satisfy the access policy and the trapdoor matches the encrypted keyword index I. The role of Authority is to issue credentials (PK/SK) to the data owners/users, the credentials are sent over secure communication channel.

The threat model of our system is as follows: DO, Authority and the authorized data users are trusted, but the CSP is a trusted-but-curious entity which honestly executes the protocol but attempts to learn some sensitive information, e.g., the query keyword information.

B. THE CONSTRUCTION OF ABMKS-WM SCHEME

In this section, we present the overview of our scheme, it is composed of eight algorithms as follows:

- **Setup**(λ) Take as input the security parameter λ, output the master key MSK and the public key PK.
- **KeyGen**(PK, MSK, S) Take as input user's attribute set S, MSK and PK, output the private key SK for the user.
- **Encrypt**$(PK, \mathcal{T}, F, W_D)$ Take as input a files set $F = f_1, \ldots, f_N$. Let W_D be the keywords set of the file $\widehat{f}$. For each file $\widehat{f} \in F$, DO generates a permutation σ respect to the file $\widehat{f}$, and then generates ciphertexts C of the file $\widehat{f}$ and index I of the keywords set W_D according to the symmetric key K and the access tree $\mathcal{T}$, respectively. At last sends the ciphertext $CT = \{C, I\}$ to the CSP.
- **GenTK**(PK, SK) Take as input public key PK and the private key SK for user's attribute set S, output the transformation key TK and the corresponding retrieving key RK.
- **Transform**(CT, TK) Take as input the ciphertext CT and the transformation key TK, output a partially decrypted ciphertext CT'.
- **Decrypt**(CT', RK) Take as input the transformed ciphertext CT' and the retrieving key RK, output the permutation σ and the symmetric key K.
- **Trapdoor**$(PK, SK, RK, W_Q, \sigma, K)$ Take as input the public key PK, private key SK and corresponding key RK, query keywords set W_Q, the permutation σ, output the trapdoor T for the query keyword W_Q.
- **Retrieve**(PK, CT, T) Take as input the ciphertext CT and the trapdoor T_{W_Q} for query keywords set W_Q. CSP checks whether T_{W_Q} satisfies the ciphertext CT, if it holds, then returns the search results C to user, otherwise, outputs $\perp$.

C. SECURITY MODEL

In this section, we present the security model of our scheme as follows.

The one goal of our scheme is that it can resist the chosen-keyword attack (CKA) [26]. As described in threat model, only the CSP is honest-but-curious. Intuitively, CKA means that the CSP (an adversary $\mathcal{A}$) cannot learn anything information about plaintext keywords set from the keywords set ciphertext except for the search tokens and the results. We present the security model by utilizing the chosen-keyword attack (CKA) game as follows:

Definition 2: Chosen-Keyword Attack Game:

- ***Setup**: The challenger $\mathcal{C}$ executes the Setup algorithm to get the public parameters PK and master key MSK, then sends the public parameters PK to the adversary $\mathcal{A}$. The adversary $\mathcal{A}$ chooses an access tree T, which is sent to the challenger.*
- ***Phase 1**: $\mathcal{A}$ can adaptively query the following oracles for polynomially many times, and the challenger $\mathcal{C}$ initializes an empty keyword list L_{kw} and an empty set D.*

  1) *$\mathcal{O}_{KeyGen}(S)$: On input a set of attributes S, the challenger $\mathcal{C}$ runs the KeyGen algorithm to get SK_S and sets $D = D\bigcup S$. It then returns it to adversary $\mathcal{A}$.*

2) $\mathcal{O}_{GenTK}(SK)$: *On input a set of attributes S, if $S \in D$, the challenger $\mathcal{C}$ runs the GenTK algorithm to get TK_S. Otherwise, the challenge runs the KeyGen algorithm to get SK_S, and runs GenTK to get TK_S. It then returns the TK_S to adversary $\mathcal{A}$.*
3) $\mathcal{O}_{Trapdoor}(SK, W_Q)$: *On input a set of keyword W_Q and the SK, the challenger $\mathcal{C}$ runs the Trapdoor algorithm to get T_{W_Q} and sets $L_{kw} = L_{kw} \bigcup W_Q$, if the attributes set S satisfies the policy tree $\mathcal{T}$. It then returns it to adversary $\mathcal{A}$.*

- ***Challenge***: *$\mathcal{A}$ randomly chooses two keywords set W_0^* and W_1^*, where $W_0^*, W_1^* \notin L_{kw}$, it means that W_0^*, W_1^* cannot be queried in $\mathcal{O}_{Trapdoor}$. Then, the challenger $\mathcal{C}$ picks a random $b \in \{0, 1\}$ and encrypts W_b^* as CT^* by using **Encrypt** algorithm. Finally, $\mathcal{C}$ returns CT^* to adversary $\mathcal{A}$.*
- *Phase 2: $\mathcal{A}$ continues to query the oracles as in Phase 1, but with the restriction that (S, W_0^*) and (S, W_1^*) cannot be the input to $\mathcal{O}_{Trapdoor}$ if the attribute set S satisfies the access policy $\mathcal{T}$.*
- ***Guess***: *$\mathcal{A}$ outputs a guess bit b', and wins the CKA game if $b' = b$; otherwise, it fails.*

Let $|Pr[b' = b] - \frac{1}{2}|$ be the advantage of $\mathcal{A}$ winning the above CKA game.

Definition 3: Our scheme is secure against chosen-keyword attack if the advantage of any $\mathcal{A}$ winning the CKA game is negligible.

The another security goal of our scheme is that the adversary cannot obtain the keyword information from the query trapdoor generated by data user, we give formal security definition of it based on a game between a adversary $\mathcal{A}$ and a challenger $\mathcal{C}$ as in the state-of-the-art work [41].

Definition 4: The query trapdoor unrecoverable security against eavesdropper attack model.

First, $\mathcal{A}$ submits challenge query keywords set for times to the challenger $\mathcal{C}$ and in return $\mathcal{C}$ sends the corresponding ciphertext to $\mathcal{A}$. Second, $\mathcal{A}$ sends two query keywords set W_0^* and W_1^* to $\mathcal{C}$, where the restrict is that W_0^* and W_1^* are not challenged before. Then, $\mathcal{C}$ chooses a bit $b \in_R \{0, 1\}$ and generates the trapdoor $T_{W_b^*}$ for the W_b^*, and sent the trapdoor $T_{W_b^*}$ to $\mathcal{A}$. $\mathcal{A}$ is allowed to continue to query $\mathcal{C}$ for the trapdoor of any keywords set W^*, but the only restriction is that W^* is not W_0^* or W_1^*. Finally, $\mathcal{A}$ outputs a guess b'. We define the advantage that $\mathcal{A}$ wins the game to be $|Pr[b' = b] - \frac{1}{2}|$. If $|Pr[b' = b] - \frac{1}{2}|$ is negligible, we say that our proposed query keyword encryption achieves query trapdoor unrecoverable security against eavesdropper attack model.

Definition 5: Our scheme is secure if the advantage of any $\mathcal{A}$ winning the game in Definition 4 is negligible.

V. OUR CONSTRUCTION

We present some notations used in our construction in Table 1, and introduce our technical construction details below.

TABLE 1. Notation used in ABMKS-WM construction.

Notations	Description
S	DU's attribute set
$F = \{\widehat{f}_1 ..., \widehat{f}_N\}$	The file set of DO
W	The keywords set in a system
W_D	The keywords set extracted from one file $\widehat{f}$
W_Q	The query keywords set chosen by DU
$\mathcal{T}$	The access policy tree for f
T_{W_Q}	The trapdoor for W_Q
C	The symmetric ciphertext of file $\widehat{f}$
K	The encryption and decryption key for C
n	The number of keyword in W
m	The number of keyword in W_D
m'	The number of keyword in W_Q
t	The number of nodes in the cover(W_D) in [27]
Y	The leaf nodes set in access tree $\mathcal{T}$

- **Setup**$(\lambda) \rightarrow (PK, MSK)$: The authority executes this algorithm. Given a security parameter p, the authority chooses a bilinear group $\mathbb{G}$ of prime order p with generator g, and chooses two random numbers $\alpha_1, \alpha_2 \in \mathbb{Z}_p^*$ and three hash functions $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p^*, H_1 : \{0, 1\}^* \rightarrow \mathbb{G}, H_2 : \mathbb{G}_T \rightarrow \{0, 1\}^l$, where H_2 is a pseudo-random generator. At last, the authority chooses a pair of encode and decode functions (*Encode*, *Decode*), where Encode denotes a function which encodes a character string to binary string, Decode denotes a function decode a binary string to character string. For example, Encode([1, 2]a) $\rightarrow$ 1011011110001101100110010101110111100001, Decode(1011011110001101100110010101110111100001) $\rightarrow$ [1, 2]a. The authority generates the public key PK and master key MSK as follows:
$$PK = \{\mathbb{G}, g, h = g^{\alpha_2}, e(g, g)^{\alpha_1}, Encode, Decode\}$$
$$MSK = \{\alpha_2, g^{\alpha_1}\}$$
- **KeyGen**$(PK, S, MSK) \rightarrow SK$: Authority executes this algorithm. Given a DU's attribute set S, the authority chooses a random number $r \in \mathbb{Z}_p^*$, and chooses r_i for each attribute $i \in S$. Finally, the authority generates the private key SK for the DU as follows:
$$SK = \{D = g^{(\alpha_1 + r)/\alpha_2}, \quad \forall i \in S : D_i = g^r H_1(i)^{r_i}, D_i' = g^{r_i}\}$$
- **Encrypt**$(PK, \mathcal{T}, F) \rightarrow CT$: DO executes this algorithm. Let $F = \{\widehat{f_1}, \ldots, \widehat{f_N}\}$ be the file set, to easy understand the encrypt algorithm, we give an example of encrypting one file $\widehat{f} \in F$. Let $W = [w_1, \ldots, w_n]$ be the keyword dictionary, W_D be the keywords set that appears in a file $\widehat{f}$.
DO will encrypt the file $\widehat{f}$ using the corresponding symmetric key K to generate the ciphertext C, e.g., using AES to encrypt the file $\widehat{f}$, but the exact algorithm is out of the scope of our discusses.
After that, DO will generate the permutation σ and the binary vector $\vec{D}$ for the keywords set W_D as follows:

1) By randomizing the order of elements in the keyword dictionary W to get the new keyword dictionary $\widetilde{W}$ and then obtaining permutation σ described in Section III-D.
2) Generate the binary vector $\vec{D} = [d_1, \ldots, d_n]$ for the keywords set W_D described in Section III-D.

For example, when

$$\widetilde{W} = [w_6, w_4, w_7, w_2, w_1, w_5, w_8, w_3]$$
$$W_D = [w_1, w_3, w_5, w_6, w_7]$$

then

$$\sigma = [1, 5, 6][2, 4][3, 8, 7]$$
$$\vec{D} = [1, 0, 1, 0, 1, 1, 0, 1]$$

DO computes the ciphertext CT as follows:

1) Choose a polynomial q_v for each node v in the access policy tree $\mathcal{T}$ in a top-down manner, and the degree d_v of q_v is $k_v - 1$, where k_v is the threshold value of the node v. Starting with the root node R of $\mathcal{T}$, DO chooses a random $s \in \mathbb{Z}_p^*$ and sets $q_R(0) = s$, it then randomly chooses d_R other points to define the polynomial q_R. For the non-root node v, it sets $q_v(0) = q_{parent(v)}(index(v))$ and randomly chooses d_v other points to define the polynomial q_v.
   Let Y be the set of leaf nodes in the access tree $\mathcal{T}$, then DO computes
   $$\begin{aligned}\theta_1 &= e(g,g)^{H(\theta_0)} \cdot e(g,g)^{\alpha_1 s},\\ \theta_2 &= \theta_0 \oplus H_2(e(g,g)^{H(\theta_0)}),\\ \theta_3 &= h^s, \theta_y = g^{q_y(0)},\\ \theta_y' &= H_1(att(y))^{q_y(0)}, \quad \forall y \in Y.\end{aligned}$$
   where $\theta_0 = Encode(\sigma||K)$.
2) With the permutation σ and the $\vec{D}$ corresponding file $\widehat{f_i}$, compute $f(x, \vec{D})$ and the encrypted keyword index I for W_D as follows:
   $$f(x, \vec{D}) = \prod_{j=1}^{n} (x - H(\sigma||j))^{d_j} \quad (5)$$
   $$I = f(x, \vec{D})_{|x=H(\theta_0)} \quad (6)$$

where $f(x, \vec{D})$ is a polynomial function and the degree of it is at most $(n-1)$. Finally, DO generates the specific ciphertexts CT for the file $\widehat{f}$ as follows:

$$CT = (C, I, \{\mathcal{T}, \theta_1, \theta_2, \theta_3, \theta_y, \theta_y'\})$$

- **GenTK**$(PK, SK) \to TK$: DU executes this algorithm. Given the public key PK and the private key $SK = \{D = g^{(\alpha_1+r)/\alpha_2}, D_i = g^r H_1(i)^{r_i}, D_i' = g^{r_i}\}$. DU chooses a random value $u \in \mathbb{Z}_p^*$, and computes transformation key TK and the corresponding retrieving key RK as follows:
  $$TK = \{S, D^* = D^u, D_i^* = D_i^u, D_i'^* = D_i'^u\} \quad RK = u$$
- **Transform**$(TK, CT) \to CT'$: CSP executes this algorithm. On receiving the TK from DU, CSP then checks whether DU's attributes set S satisfies the access tree $\mathcal{T}$. If S does not satisfy $\mathcal{T}$, the algorithm outputs $\perp$; otherwise, CSP continues to run the algorithm as follows:
  1) If the node x is a leaf node in $\mathcal{T}$. We let i=$att(x)$. If $i \in S$, then compute ϕ_x as
  $$\begin{aligned}\phi_x &= \frac{e(D_i^*, \theta_x)}{e(D_i'^*, \theta_{x'})}\\ &= \frac{e(g^{ru} \cdot H_1(i)^{r_i u}, g^{q_x(0)})}{e(g^{r_i u}, H_1(i)^{q_x(0)})}\\ &= e(g,g)^{ru \cdot q_x(0)}\end{aligned}$$
  2) If node x is not a leaf node in $\mathcal{T}$, we get ϕ_x by computing $\phi_{x'}$ using a recursive algorithm, where x' is child node of x. Let S_x be an arbitrary k_x set of children nodes x; if no such set exists, set $\phi_{x'} = \perp$; otherwise, compute $\phi_{x'}$ as follows, where $i =$index(x'), $S_x' = \{\text{index}(x') : x' \in S_x\}$
  $$\begin{aligned}\phi_x &= \prod_{x' \in S_x} \phi_{x'}^{\Delta_{i,S_x'}(0)}\\ &= \prod_{x' \in S_x} (e(g,g)^{ru \cdot q_{x'}(0)})^{\Delta_{i,S_x'}(0)}\\ &= \prod_{x' \in S_x} (e(g,g)^{ru \cdot q_{parent(z)}(index(z))})^{\Delta_{i,S_x'}(0)}\\ &= \prod_{x' \in S_x} (e(g,g)^{ru \cdot q_x(i)})^{\Delta_{i,S_x'}(0)}\\ &= e(g,g)^{ru \cdot q_x(0)}\end{aligned}$$

  If the tree is satisfied by S, we set $A = \phi_{root} = e(g,g)^{ru \cdot q_R(0)} = e(g,g)^{rus}$, and compute the partially-decrypted ciphertext pct as follow.
  $$\begin{aligned}pct &= e(\theta_3, D^*)/A\\ &= e(h^s, g^{(\alpha_1+r)\cdot u/\alpha_2})/e(g,g)^{rsu}\\ &= e(g,g)^{\alpha_1 su}\end{aligned}$$

  Return out $CT' = (\theta_1, \theta_2, pct)$.
- **Decrypt**$(RK, CT') \to \sigma||K$: DU executes this algorithm. On receiving the CT' from CSP, DU obtains σ and K as follows.
  $$\begin{aligned}e(g,g)^{H(\theta_0)} &= \theta_1/pct^{\frac{1}{RK}},\\ &= \frac{e(g,g)^{H(\theta_0)} \cdot e(g,g)^{\alpha_1 s}}{(e(g,g)^{\alpha_1 su})^{\frac{1}{u}}}. \quad (7)\\ Encode(\sigma||K) &= \theta_2 \oplus H_2(e(g,g)^{H(\theta_0)}). \quad (8)\\ \sigma||K &= Decode(Encode(\sigma||K)). \quad (9)\end{aligned}$$

  Return the permutation σ and the symmetric key K corresponding to the file $\widehat{f}$.
- **Trapdoor**$(PK, SK, RK, W_Q, \sigma, K) \to T$: DU executes this algorithm. Let W_Q be the query keywords set of DU. After obtaining the permutation σ, DU generates

the new keyword dictionary $\widetilde{W} = [\widetilde{w}_1, \ldots, \widetilde{w}_n]$ by using σ.

After that, DU will generate the binary vector $\overrightarrow{D}$ for the keywords set W_Q as follows:

1) Generate the binary vector $\overrightarrow{Q} = [q_1, \ldots, q_n]$ for the keywords set W_Q described in Section III-D.

For example, when

$$\sigma = [1, 6, 5][2, 4][3, 7, 8]$$
$$W_Q = [w_1, w_3, w_5, w_7]$$

then

$$\widetilde{W} = [w_6, w_4, w_7, w_2, w_1, w_5, w_8, w_3]$$
$$\overrightarrow{Q} = [0, 0, 1, 0, 1, 1, 0, 1]$$

Finally, compute $f(x, \overrightarrow{Q})$ and T with $\sigma, \overrightarrow{Q}$ as

$$f(x, \overrightarrow{Q}) = \prod_{j=1}^{n}(x - H(\sigma||j))^{q_j} \tag{10}$$

$$T = f(x, \overrightarrow{Q})_{|x=H(\theta_0)} \tag{11}$$

where $\theta_0 = Encode(\sigma||K)$. and then return the trapdoor T.

- **Retrieve**$(PK, CT, T) \rightarrow (C\&\perp)$: CSP executes this algorithm. According to the definition of $f(x, \overrightarrow{D})$ and $f(x, \overrightarrow{Q})$ in Encryption and Trapdoor algorithm. We can infer that $\frac{f(x,\overrightarrow{D})}{f(x,\overrightarrow{Q})} = \prod_{j=1}^{n}(x - H(\sigma||j))^{d_j - q_j}$ is a polynomial function in x, only if only the $\overrightarrow{Q} \subseteq \overrightarrow{D}$. Otherwise, it is not a polynomial function. Thus if $\frac{f(x,\overrightarrow{D})}{f(x,\overrightarrow{Q})}$ is a polynomial function in x, when set $x = H(Encode(\sigma||K))$ which is a integer, then $\frac{f(x,\overrightarrow{D})}{f(x,\overrightarrow{Q})} = \frac{I}{T}$ is also a integer. This means that I can be exactly divisible by T. Thus, if Eq.(9) holds, thereby $\overrightarrow{Q} \subseteq \overrightarrow{D}$ $(W_Q \subseteq W_D)$, CSP send the associated search result C to the user; otherwise, return $\perp$.

$$I \ mod \ T \overset{?}{=} 0 \tag{12}$$

where *mod* represents integer modular operation. On receiving all the search results from CSP, the user can decrypt them with the corresponding symmetric key K.

Correctness Analysis: Assume that the submitted attributes set S satisfies the access policy tree $\mathcal{T}$ and $\overrightarrow{Q} \subseteq \overrightarrow{D}$ $(W_Q \subseteq W_D)$, we have that

$$\frac{f(x, \overrightarrow{D})}{f(x, \overrightarrow{Q})} = \prod_{j=1}^{n}(x - H(\sigma||j))^{d_j - q_j} \tag{13}$$

$$\frac{I}{T} = \frac{f(x, \overrightarrow{D})}{f(x, \overrightarrow{Q})}_{|x=H(Encode(\sigma||K))}$$

$$= \prod_{j=1}^{n}(H(\sigma||K), \overrightarrow{Q}) - H(\sigma||j)^{d_j - q_j}) \tag{14}$$

$$I \ mod \ T = 0 \tag{15}$$

Then, we state that Eq.(12) holds if $\overrightarrow{Q} \subseteq \overrightarrow{D}$ $(W_Q \subseteq W_D)$.

VI. SECURITY ANALYSIS

In this section, we present the security analysis of our scheme which is proved to be secure by using the following theorem.

Theorem 2: Given the oracle H_1, H_2 and the one-way hash function H, and support that the CP-ABE scheme [18] is selectively CPA-secure, then the ABMKS-WM scheme is SCKA secure.

Proof: To prove this theorem, we present the two games as follows:

- **Game 0**: The Selectively Chosen-Keyword Attack Game of ABMKS-WM scheme.
- **Game 1**: Same as Game 0 except for the way that the challenger generates the challenge ciphertext $CT^* = (I, \{\mathcal{T}, \theta_1, \theta_2, \theta_3, \theta_y, \theta'_y\})$, where the item I is a random integer and the bit length of I is about $|\mathbb{Z}_p^*|m$-bit, the rest items of CT^* are generated as in **Game 0**.

We prove this theorem by the following two lemmas. **Lemma 1** proves that **Game 0** and **Game 1** are indistinguishable; **Lemma 2** proves that the advantage of the adversary in **Game 1** is negligible. Therefore, we state that the advantage of the adversary in **Game 0** is negligible and the **Theorem 2** is completed. □

Lemma 1: Support that the CP-ABE scheme [18] is selectively CPA-secure, then the **Game 0** and **Game 1** are computationally indistinguishable.

Proof: We state that if there exists an adversary $\mathcal{A}$ who can distinguish the **Game 0** and **Game 1** with a non-negligible advantage ϵ, we then can build an algorithm $\mathcal{B}$ that can break of the CP-ABE scheme [18] with a non-negligible advantage at least ϵ.

Let $\mathcal{C}$ be the challenger corresponding to $\mathcal{B}$ in the secure game of CP-ABE scheme [18]. The $\mathcal{B}$ runs $\mathcal{A}$ by executing the following steps.

- **Init**: $\mathcal{A}$ gives $\mathcal{B}$ a challenge access policy $\mathcal{T}^*$. $\mathcal{B}$ sends the $\mathcal{T}^*$ to $\mathcal{C}$ as its challenge access policy and is given the public key PK' of the CP-ABE scheme [18]. $PK' = \{\mathbb{G}, g, h = g^{\alpha_2}, e(g, g)^{\alpha_1}, H_1\}$
- **Setup**: $\mathcal{B}$ chooses two hash functions $H : \{0, 1\}^* \rightarrow \mathbb{Z}_p^*$, $H_2 : \mathbb{G}_T \rightarrow \{0, 1\}^*$, then chooses a pair of encode and decode functions $(Encode, Decode)$ and a keyword dictionary W, then sends the public key PK=$\{\mathbb{G}, g, h = g^{\alpha_2}, e(g, g)^{\alpha_1}, H_1, H_2, H, Encode, Decode\}$ to $\mathcal{A}$.
- **Phase 1**: The adversary $\mathcal{A}$ issues private key, transformation key and trapdoor generations generations to the following oracles, respectively.

  1) When $\mathcal{A}$ adaptively issues a private key query for a set of attributes S, $\mathcal{B}$ calls the key generation oracle of $\mathcal{C}$ on S to obtain the private key SK. Then, returns the SK to $\mathcal{A}$.

2) When $\mathcal{A}$ adaptively issues a transformation key query for a set of attributes S, chooses a random $u \in \mathbb{Z}_p^*$ and computes $TK = SK^U$. Then, returns the TK to $\mathcal{A}$.
3) When $\mathcal{A}$ adaptively issues a trapdoor query for S and keywords set W_Q, chooses a random permutation σ of the keyword dictionary W, computes $T = f(x, \overrightarrow{Q_{W_Q}})_{|x=H(Encode(\sigma||K))}$, where K is a random symmetric encryption key. If S satisfies $\mathcal{T}$, $\mathcal{B}$ stores the keywords set W_Q in the keyword list L_{W_Q}.

- **Challenge**: $\mathcal{A}$ summits two keyword sets W_{D0} and W_{D1} to be challenged on, with the restriction that W_{D0} and W_{D1} have not been queried in the list L_{W_Q}. $\mathcal{B}$ chooses a random bit $\delta \in \{0, 1\}$, two random symmetric keys K_0 and K_1, and two random permutations σ_0 and σ_1, and sends the σ_0, σ_1, K_0, K_1 and $\mathcal{T}^*$ to $\mathcal{C}$. $\mathcal{C}$ chooses a random bit $\beta \in \{0, 1\}$, and encrypts σ_β and K_β under the public key PK' and $\mathcal{T}^*$ by using the encryption algorithm of CP-ABE scheme [18], and sends the resulting ciphertext $CT^{*\prime} = (\{\mathcal{T}, \theta_1, \theta_2, \theta_3, \theta_y, \theta'_y\})$ to $\mathcal{B}$. Then computation $I = f(x, \overrightarrow{D_{W_{D\delta}}})_{|x=H(Encode(\sigma_\delta||K_\delta))}$. Finally, $\mathcal{B}$ sets $CT^* = (I, \{\mathcal{T}, \theta_1, \theta_2, \theta_3, \theta_y, \theta'_y\})$, and sends CT^* to $\mathcal{A}$ as its challenge ciphertext.
- **Phase 2**: This phase is similar to Phase 1, with the restriction that W_{D0} and W_{D1} have not been issued in Phase 1.
- **Guess**: The adversary outputs a guess δ' for δ. $\mathcal{B}$ also outputs δ' for β.

Note that, if $\delta = \beta$, then $\mathcal{B}$ has properly simulated **Game 0**; otherwise, has properly simulated **Game 1**. Thus, if $\mathcal{A}$ can distinguish **Game 0** and **Game 1** with non-negligible advantage ϵ, we can build $\mathcal{B}$ algorithm to break the CP-ABE scheme [18] with non-negligible advantage ϵ. □

Lemma 2: Assume that the CP-ABE scheme [18] is selectively CPA-secure, then the advantage of the adversary in **Game 1** is negligible. *Proof:* We state that if there exists an adversary $\mathcal{A}$ who can win the Game 1 with a non-negligible advantage ϵ, we then can build an algorithm $\mathcal{B}$ that can break the CP-ABE scheme [18] with a non-negligible advantage at least ϵ.

Let $\mathcal{C}$ be the challenger corresponding to $\mathcal{B}$ in the secure game of CP-ABE scheme [18]. $\mathcal{B}$ runs $\mathcal{A}$ by executing the following steps.

- **Init**: The $\mathcal{A}$ gives $\mathcal{B}$ a challenge access policy $\mathcal{T}^*$. $\mathcal{B}$ sends the $\mathcal{T}^*$ to $\mathcal{C}$ as its challenge access policy and is given the public key PK' of the CP-ABE scheme [18]. $PK' = \{\mathbb{G}, g, h = g^{\alpha_2}, e(g, g)^{\alpha_1}, H_1\}$
- Setup: $\mathcal{B}$ chooses two hash functions $H : \{0, 1\}^* \to \mathbb{Z}_p^*$, $H_2 : \mathbb{G}_T \to \{0, 1\}^*$, then chooses a pair of encode and decode functions ($Encode$, $Decode$) and a keyword dictionary W, then sends the public key $PK = \{\mathbb{G}, g, h = g^{\alpha_2}, e(g, g)^{\alpha_1}, H_1, H_2, H, Encode, Decode\}$ to $\mathcal{A}$.
- **Phase 1**: The adversary $\mathcal{T}$ issues private key, transformation key and trapdoor generations generations to the following oracles, respectively.

1) When the adversary $\mathcal{A}$ adaptively issues a private key query for a set of attributes S (chose by $\mathcal{A}$), $\mathcal{B}$ calls the key generation oracle of $\mathcal{C}$ on S to obtain the private key SK. Then, returns the SK to $\mathcal{A}$.
2) When the adversary $\mathcal{A}$ adaptively issues a transformation key query for a set of attributes S, chooses a random $u \in \mathbb{Z}_p^*$ and computes $TK = SK^U$. Then, returns the TK to $\mathcal{A}$.
3) When the adversary $\mathcal{A}$ adaptively issues a trapdoor query for S and keywords set W_Q (chose by $\mathcal{A}$), chooses a random permutation σ of the keyword dictionary W, computes $T = f(x, \overrightarrow{Q_{W_Q}})_{|x=H(Encode(\sigma||K))}$, where K is a random symmetric encryption key. If S satisfies the $\mathcal{T}$, $\mathcal{B}$ stores the keywords set W_Q in the keyword list L_{W_Q}.

- **Challenge**: $\mathcal{A}$ summits two keyword sets W_{D0} and W_{D1} to be challenged on, with the restriction that W_{D0} and W_{D1} have not been queried in L_{W_Q}. $\mathcal{B}$ chooses a random bit $\delta \in \{0, 1\}$, two random symmetric keys K_0 and K_1, and two random permutations σ_0 and σ_1, and sends σ_0, σ_1, K_0, K_1 and $\mathcal{T}^*$ to $\mathcal{C}$. $\mathcal{C}$ chooses a random bit $\beta \in \{0, 1\}$, and encrypts σ_β and K_β under the public key PK' and $\mathcal{T}^*$ by using the encryption algorithm of CP-ABE scheme [18], and sends the resulting ciphertext $CT^{*\prime} = \{\mathcal{T}, \theta_1, \theta_2, \theta_3, \theta_y, \theta'_y\}$ to $\mathcal{B}$. Then randomly generate I, where I is a random integer and the big long of I is about $|\mathbb{Z}_p^*|m$. Finally, $\mathcal{B}$ sets $CT^* = (I, \{\mathcal{T}, \theta_1, \theta_2, \theta_3, \theta_y, \theta'_y\})$, and sends CT^* to $\mathcal{A}$ as its challenge ciphertext.
- **Phase 2**: This phase is similar to Phase 1, with the restriction that W_{Q0} and W_{Q1} have not been issued in Phase 1.
- **Guess**: The adversary outputs a guess δ' for δ. $\mathcal{B}$ also outputs δ' for β.

We can see that $\mathcal{B}$ has properly simulated **Game 1**. Thus, if $\mathcal{A}$ win **Game 1** with non-negligible advantage ϵ, we can build an algorithm $\mathcal{B}$ to break the CP-ABE scheme [18] with non-negligible advantage at least ϵ. □

Theorem 3: Given the one-way hash function H, and support that the CP-ABE scheme [18] is selectively CPA-secure, then achieves query trapdoor unrecoverable security against eavesdropper attack model.

Proof: Our proof is similar to the proof in the scheme [16], [41], we prove the Theorem 3 by using the game as follows:

Setup: Let CT_{W_D} denote the ciphertext corresponding to a keyword set W_D, which are generated based on the W_D,σ and K. And then sent the CT to adversary $\mathcal{A}$.

(1) The adversary $\mathcal{A}$ queries the following oracles for polynomially-many times.
$\mathcal{O}_{Trapdoor}(SK, W_Q)$: On input a set of keyword W_Q and the SK, the challenger $\mathcal{C}$ runs the *Trapdoor* algorithm to get T_{W_Q} and sets $L_{kw} = L_{kw} \bigcup W_Q$, if the attributes set adversary's S satisfies the policy tree $\mathcal{T}$ in CT. It then returns T_{W_Q} to adversary $\mathcal{A}$, where $T_{W_Q} = f(x, \overrightarrow{Q})_{|x=H(Encode(\sigma||K))}$, $f(x, \overrightarrow{Q}) = \prod_{j=1}^{n}(x - H(\sigma||j))^{q_j}$ and $\overrightarrow{Q}$ are generated based on the W_Q and σ.

TABLE 2. Notation used in performance analysis.

Notation	Description
P	The bilinear pairing operation
M	The multiplication operation in group $\mathbb{Z}$
$E_{\mathbb{G}}$	The modular exponentiation operation in group $\mathbb{G}$
$E_{\mathbb{G}_T}$	The modular exponentiation operation in group $\mathbb{G}_T$
H_1	A hash function which maps a bit-string to an element of the group $\mathbb{G}$
H	A hash function which maps a bit-string to an element of the group $\mathbb{Z}_p$
$\lvert\mathbb{G}\rvert$	The bit length of a element in $\mathbb{G}$
$\lvert\mathbb{G}_T\rvert$	The bit length of a element in $\mathbb{G}_T$
$\lvert\mathbb{Z}_p^*\rvert$	The bit length of a element in $\mathbb{Z}_p^*$
s	The number of a DU's attributes
n	The number of keyword in the keyword dictionary W
m	The number of keyword in W_D
m'	The number of keyword in a query keywords set W_Q
t	The number of nodes in the cover(W') in [27]
$\lvert Y\rvert$	The number of leaf nodes in access tree $\mathcal{T}$

TABLE 3. Computation cost comparison.

Scheme	PAB-MKS scheme [26]	ABMKS-FAIC scheme [27]	Ours
KeyGen	$(2s{+}2)\cdot E_{\mathbb{G}}$	$(2s{+}2)\cdot E_{\mathbb{G}}$	$(2s{+}2)\cdot E_{\mathbb{G}}$
Encrypt	$(2\lvert Y\rvert{+}3{+}m)\cdot E_{\mathbb{G}}$	$(2\lvert Y\rvert{+}1)\cdot E_{\mathbb{G}}{+}(t{+}1)\cdot E_{\mathbb{G}_T}$	$(2\lvert Y\rvert{+}1)\cdot E_{\mathbb{G}}{+}m\cdot M{+}E_{\mathbb{G}_T}$
Trapdoor	$(2s+3)\cdot E_{\mathbb{G}}$	$(2s+1)\cdot E_{\mathbb{G}}{+}m'\cdot\log n\cdot M$	$(2s+1)\cdot E_{\mathbb{G}}{+}m'\cdot M$
Retrieve	$(2s{+}4)\cdot P{+}s\cdot E_{\mathbb{G}_T}$	$(2s{+}1)\cdot P{+}(s{+}2m'{-}1)\cdot E_{\mathbb{G}_T}$	$(2s{+}1)\cdot P{+}s\cdot E_{\mathbb{G}_T}$+mod

TABLE 4. Storage cost comparison.

Scheme	PAB-MKS scheme [26]	ABMKS-FAIC scheme [27]	Ours
KeyGen	$(2s{+}1)\cdot\lvert\mathbb{G}\rvert$	$(2s{+}1)\cdot\lvert\mathbb{G}\rvert$	$(2s{+}1)\cdot\lvert\mathbb{G}\rvert$
Encrypt	$(2\lvert Y\rvert{+}3{+}m)\cdot\lvert\mathbb{G}\rvert$	$(2\lvert Y\rvert{+}1)\cdot\lvert\mathbb{G}\rvert{+}(t{+}1)\cdot\lvert\mathbb{G}_T\rvert$	$(2\lvert Y\rvert{+}1)\cdot\lvert\mathbb{G}\rvert{+}m\cdot\lvert\mathbb{Z}_p^*\rvert$
Trapdoor	$(2s{+}3)\cdot\lvert\mathbb{G}\rvert$	$(2s{+}1)\cdot\lvert\mathbb{G}\rvert{+}(m'\cdot\log n)\cdot\lvert\mathbb{Z}_p^*\rvert$	$(2s{+}1)\cdot\lvert\mathbb{G}\rvert{+}m'\cdot\lvert\mathbb{Z}_p^*\rvert$

(2) The adversary chooses two query keywords set W_{Q0} and W_{Q1} and sent them to challenger $\mathcal{C}$, the restriction is that the W_{Q0} and W_{Q1} is in L_{kw}.

(3) Challenger $\mathcal{C}$ chooses $\delta \in_R \{0, 1\}$, and generate the trapdoor T_{W_δ} as in the Trapdoor algorithm. Then set the T_{W_δ} to adversary.

(4) The adversary can query the oracles as in step (1) with the restriction that the query keywords set other than W_{Q0} and W_{Q1}.

(5) The adversary outputs the guess b' of b. Because the CP-ABE scheme [18] is selectively CPA-secure, then the adversary cannot recovery the $\sigma||K$ from CT_{W_D}, thus the adversary cannot effectively compute T_{W_0} and T_{W_1} without σ,K.

It means that the adversary only has $1/2 + 1/n!$ advantage to guess $b' = b$. If the CP-ABE scheme [18] is selectively, as Theorem 2, the adversary cannot decrypt out the $\sigma||K$ from CT_{W_D}, in addition, as description in Section III-D, the adversary has $1/n!$ to guess out the σ, then the adversary has $1/2+1/n!$ advantage to compute the valid trapdoors T_{W_0} and T_{W_1}. □

VII. PERFORMANCE ANALYSIS

In this section, we compare the performance of our scheme with the related work [26], [27]. Table 2 presents the notations used in this performance analysis.

A. COMPUTATION COMPLEXITY

We compare the schemes [26], [27] with our scheme in terms of computation complexity in Table 3. The time-consuming operations mainly consist of P, $E_{\mathbb{G}}$, $E_{\mathbb{G}_T}$ and M, we ignore the hash functions H_1, H and the XOR ($\bigoplus$) for that these operations are more efficient than the above operations.

As shown in Table 3, the computation cost in KeyGen algorithm of our scheme is as efficient as [26], [27]. Compared with [26], [27], the computation cost in the Encryption, Trapdoor and Retrieve algorithms of our scheme are more efficient than [26] and [27], respectively. For example, in the Encryption algorithm, the computation cost of [26] and [27] are $(2|Y|+3+m)E_{\mathbb{G}}$ and $(2|Y|+1)E_{\mathbb{G}}+(t+1)E_{\mathbb{G}_T}$, respectively, while the computation cost is $(2|Y|+1)E_{\mathbb{G}}+mM+E_{\mathbb{G}_T}$ in our scheme. Due to the multiplication operation M is more efficient than the modular exponentiations $E_{\mathbb{G}}$ and $E_{\mathbb{G}}$, we can

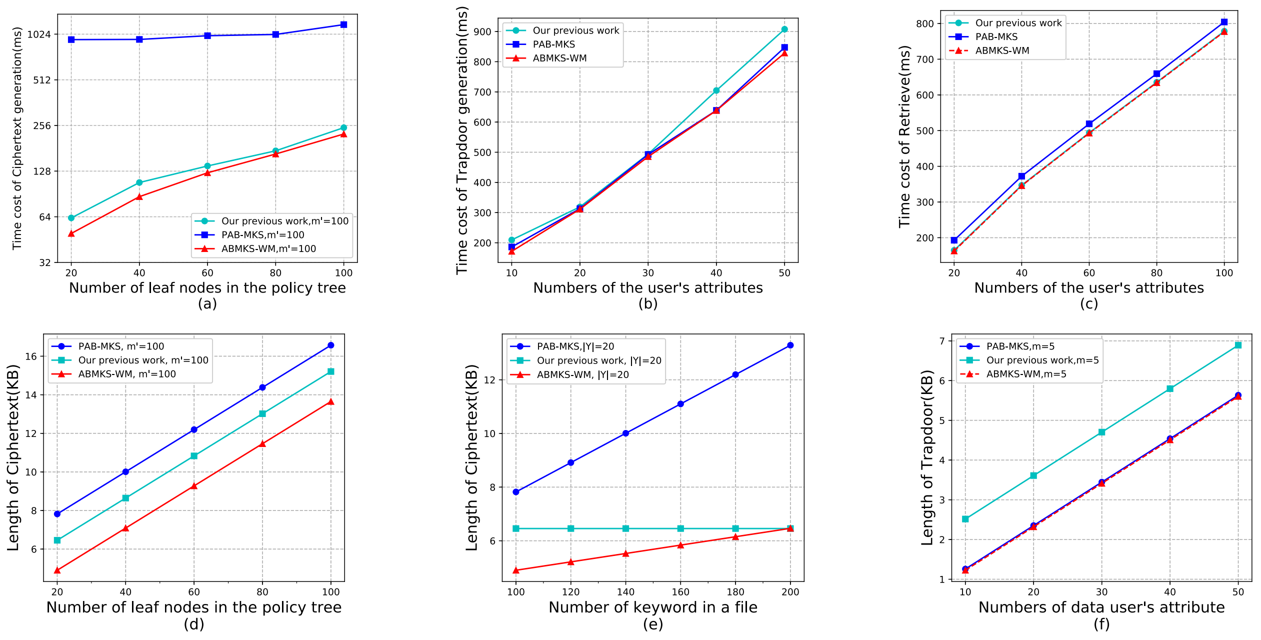


FIGURE 5. The Performance Comparison between related works and our scheme.

note that the computation cost in encryption algorithm of our scheme is more efficient than [26] and [27].

B. STORAGE COMPLEXITY

We compare the schemes [26], [27] with our scheme in terms of storage complexity in Table 4.

As shown in Table 4, the storage cost in the KeyGen algorithm of our scheme is the same as [26], [27]. Compared with [26], [27], the storage cost in Encryption and Trapdoor algorithms of our scheme are less than [26], [27], respectively. For example, in the Encryption algorithm, the storage cost of [26], [27] and our schemes are $(2|Y| + 3 + m)|\mathbb{G}|$, $(2|Y| + 1)|\mathbb{G}| + (t + 1)|\mathbb{G}_T|$, $(2|Y| + 1)|\mathbb{G}| + m|\mathbb{Z}_p^*|$, respectively. Due to $|\mathbb{Z}_p^*| = 160$-bit, $|\mathbb{G}| = |\mathbb{G}_T| = 224$-bit, when setting $n = 1000$ (the $t = 18$ when $n = 1000$ in scheme [27]), $m = 100$, we note that $m|\mathbb{Z}_p^*| < m|\mathbb{G}|$ and $m|\mathbb{Z}_p^*| < (t + 1)|\mathbb{G}_T|$. Thus the storage cost in the encryption algorithm of our scheme is efficient than [26], [27].

C. EXPERIMENTAL PERFORMANCE

In this section, we implement the schemes [26], [27] and our scheme by using Python language on the Ubuntu 16.04 LTS with Intel Core i3 Processor 4170 CPU @3.70GHZ with 10.0 GB of RAM. Because these three schemes are highly dependent on the basic cryptographic operations in the pairing computation, we implement these three schemes in software based on the libfenc library [45] and choose a 224-bit MNT cure ($|\mathbb{G}| = \mathbb{G}_T| = 224$-bit) from the Stanford Pairing-Based Crypto library.

For the comparison, we assume that these three schemes have the same policy tree $\mathcal{T}$: ((A_1 or A_2) and (A_3 or A_4) and …and ($A_{|Y|-1}$ or $A_{|Y|}$)). We set the number of DU's attributes $s \in [10, 20, 30, 40, 50]$, the number of leaf nodes in policy tree $|Y| \in [20, 40, 60, 80, 100]$, the number of keyword in keyword dictionary $n = 500$, the number of keyword appeared in a file $m = 100$, the $|\mathbb{Z}_p^*| = 160$ and the number of query keywords $m' \in [3, 5, 7, 9, 11]$.

As shown in Fig. 5(a), the computation cost in ciphertext generation of our scheme is efficient than [26], [27]. As described in Table 3, the computation cost ciphertext generation of the scheme [26], [27] and our scheme are $(2|Y|+3+m)E_{\mathbb{G}}$, $(2|Y|+1)E_{\mathbb{G}}+(t+1)E_{\mathbb{G}_T}$ and $(2|Y|+1)E_{\mathbb{G}}+mM + E_{\mathbb{G}_T}$, respectively. Due to the multiplication M is efficient than exponentiation $E_{\mathbb{G}}$ and $E_{\mathbb{G}_T}$. Thus our scheme is more efficient than [26], [27] in the ciphertext generation. For example, when $|Y| = 40$, $m = 100$, $n = 1000$, the schemes [26] and [27] need 945.48 ms and 107.68 ms, respectively, while our scheme needs 86.87 ms.

As shown in Fig. 5(b), the computation cost in trapdoor algorithm of our scheme is slightly efficient than [26], [27]. As described in Table 3, the computation cost in the trapdoor algorithm of our scheme are $(2s+3)E_{\mathbb{G}}$, $(2s+1)E_{\mathbb{G}}+m'\log nM$ and $(2s+1)E_{\mathbb{G}}+m'M$, respectively. Due to the multiplication M is more efficient than exponentiation $E_{\mathbb{G}}$ and $E_{\mathbb{G}_T}$. Thus our scheme is efficient than [26], [27]. For example, when $s = 50$, $m = 100$, $n = 1000$, as for the trapdoor algorithm, the schemes [26], [27] need 907.67 ms and 847.41 ms, respectively, while our scheme needs 828.78 ms.

As shown in Fig. 5(c), the computation cost of retrieve algorithm in our scheme is less than [26], [27]. As described in Table 3, the time cost of [26], [27] and our schemes are $(2s + 4)P + sE_{\mathbb{G}_T}$, $(2s + 1)P + (s + 2m - 1)E_{\mathbb{G}_T}$

and $(2s+1)P + \text{mod}$, respectively. Due to once time cost of mod operation is efficient than $3 \cdot P$ or $(2m-1)E_{\mathbb{G}_T}$. Thus, our scheme is efficient than [26], [27] in the retrieve algorithm. For example, when $s = 50$, $m' = 2$, $n = 1000$, as for the trapdoor algorithm, the schemes [27] and [26] need 164.72 ms and 192.97 ms, respectively, while our scheme needs 162.82 ms.

As shown in Fig. 5(d) and Fig. 5(e), the storage cost of encryption algorithm in our scheme is less than [26], [27]. As described in Table 4, the storage cost of [26], [27] and our schemes are $(m+2|Y|+3)|\mathbb{G}|$, $(2|Y|+1)|\mathbb{G}|+(t+1)|\mathbb{G}_T|$ and $(2|Y|+1)|\mathbb{G}|+m|\mathbb{Z}_p^*|$, respectively. Due to $|\mathbb{Z}_p^*| = 160$, $|\mathbb{G}| = |\mathbb{G}_T| = 224$, when $n = 1000$, $m = 100$. As shown in Fig. 5(d), the storage cost in the encryption algorithm of our scheme is less than [26], [27] along with increase of the number $|Y|$. As shown in Fig. 5(e), when $n = 1000$, $|Y| = 20$, the storage cost in the encryption algorithm of our scheme is less than [26], [27] along with the increase of of m. For example, when $n = 500$, $|Y| = 40$, and $m = 100$, the schemes [26] and [27] need 10.0 kb and 8.64 kb, respectively, while our scheme needs 7.09 kb.

As shown in Fig. 5(d) and Fig. 5(e), the storage cost in the trapdoor algorithm of our scheme is slightly less than [26], [27]. For example, when setting $m' = 5$, $|Y| = 50$, the schemes [26], [27] need 5.63 kb, 6.89 kb, respectively, while our scheme needs 5.60 kb.

VIII. CONCLUSION

In this paper, we have designed an ABMKS with only multiplication operations in encrypted keyword index generation which provides secure multi-keyword search service with fine-grained access control. The computation operations in the index generation are only multiplication, which is more efficient than modular exponentiation and pairing. In addition, the encrypted keyword indexes are aggregated into one item, being independent on the number of underlying keyword in a file. The formal security analysis shows that our scheme is secure. Moreover, the performance evaluation demonstrates that the ABMKS-WM scheme is better than the current works in terms of both the computation and communication overhead.

CONFLICTS OF INTEREST

The authors declare that there are no conflicts of interest regarding the publication of this article.

REFERENCES

[1] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Proc. Int. Conf. Financial Cryptogr. Data Secur.* Berlin, Germany: Springer, Jan. 2010, pp. 136–149.

[2] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, Feb. 2013.

[3] A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, and I. Stoica, "Above the clouds: A Berkeley view of cloud computing," Dept. Elect. Eng. Comput. Sci., Univ. California, Berkeley, Berkeley, CA, USA, Tech. Rep. UCB/EECS-2009-28, 2009, vol. 28, no. 13.

[4] A. D. Josep, R. Katz, A. Konwinski, L. E. E. Gunho, D. Patterson, and A. Rabkin, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[5] C. Gentry and D. Boneh, *A Fully Homomorphic Encryption Scheme*, vol. 20, no. 9. Stanford, CA, USA: Stanford Univ., 2009.

[6] O. Goldreich and R. Ostrovsky, "Software protection and simulation on oblivious RAMs," *J. ACM*, vol. 43, no. 3, pp. 431–473, May 1996.

[7] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy. (S&P)*, May 2000, pp. 44–55.

[8] Y. C. Chang and M. Mitzenmacher, "Privacy preserving keyword searches on remote encrypted data," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.* Berlin, Germany: Springer, 2004.

[9] M. Bellare, A. Boldyreva, and A. O'Neill, "Deterministic and efficiently searchable encryption," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, Aug. 2007, pp. 535–552.

[10] F. Bao, R. H. Deng, X. Ding, and Y. Yang, "Private query on encrypted data in multi-user settings," in *Proc. Int. Conf. Inf. Secur. Pract. Exper.* Berlin, Germany: Springer, Apr. 2008, pp. 71–85.

[11] Q. Chai and G. Gong, "Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2012, pp. 917–922.

[12] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," *J. Comput. Secur.*, vol. 19, no. 5, pp. 895–934, Nov. 2011.

[13] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. ACM Conf. Comput. Commun. Secur. (CCS)*, 2012, pp. 965–976.

[14] 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.

[15] D. Boneh, C. G. Di, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Int. Conf. Theory Appl. Cryptograph. Techn.* Berlin, Germany: Springer, May 2004, pp. 506–522.

[16] Q. Zheng, S. Xu, and G. Ateniese, "VABKS: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proc. IEEE INFOCOM-IEEE Conf. Comput. Commun.*, Apr. 2014, pp. 522–530.

[17] 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. (CCS)*, 2006, pp. 89–98.

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

[19] J. Li, X. Lin, Y. Zhang, and J. Han, "KSF-OABE: Outsourced attribute-based encryption with keyword search function for cloud storage," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 715–725, Sep. 2017.

[20] J. Li, Y. Zhang, J. Ning, X. Huang, G. S. Poh, and D. Wang, "Attribute based encryption with privacy protection and accountability for CloudIoT," *IEEE Trans. Cloud Comput.*, early access, Feb. 19, 2020, doi: 10.1109/TCC.2020.2975184.

[21] J. Li, Y. Wang, Y. Zhang, and J. Han, "Full verifiability for outsourced decryption in attribute based encryption," *IEEE Trans. Services Comput.*, early access, May 31, 2017, doi: 10.1109/TSC.2017.2710190.

[22] J. Li, Q. Yu, and Y. Zhang, "Hierarchical attribute based encryption with continuous leakage-resilience," *Inf. Sci.*, vol. 484, pp. 113–134, May 2019.

[23] J. Li, W. Yao, J. Han, Y. Zhang, and J. Shen, "User collusion avoidance CP-ABE with efficient attribute revocation for cloud storage," *IEEE Syst. J.*, vol. 12, no. 2, pp. 1767–1777, Jun. 2018.

[24] 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. 2017.

[25] Y. Miao, J. Ma, X. Liu, X. Li, Q. Jiang, and J. Zhang, "Attribute-based keyword search over hierarchical data in cloud computing," *IEEE Trans. Services Comput.*, early access, Sep. 28, 2017, doi: 10.1109/TSC.2017.2757467.

[26] Y. Miao, J. Ma, X. Liu, X. Li, Z. Liu, and H. Li, "Practical attribute-based multi-keyword search scheme in mobile crowdsourcing," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3008–3018, Aug. 2018.

[27] Z. Li, W. Li, F. Gao, W. Yin, H. Zhang, Q. Wen, and K. Liang, "Towards privacy-preserving and efficient attribute-based multi-keyword search," Cryptol. ePrint Arch., Tech. Rep. 2019/1314, Nov. 2019. [Online]. Available: https://eprint.iacr.org/2019/1314

[28] E.-J. Goh, "Secure indexes," IACR Cryptol. ePrint Arch., Tech. Rep. 2003/216, Mar. 2004. [Online]. Available: https://eprint.iacr.org/2003/216

[29] P. Golle, J. Staddon, and B. Waters, "Secure conjunctive keyword search over encrypted data," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.* Berlin, Germany: Springer, Jun. 2004, pp. 31–45.

[30] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, and H. Shi, "Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions," in *Proc. Annu. Int. Cryptol. Conf.* Berlin, Germany: Springer, Aug. 2005, pp. 205–222.

[31] Y. H. Hwang and P. J. Lee, "Public key encryption with conjunctive keyword search and its extension to a multi-user system," in *Proc. Int. Conf. Pairing-Based Cryptogr.* Berlin, Germany: Springer, Jul. 2007, pp. 2–22.

[32] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren, and W. Lou, "Fuzzy keyword search over encrypted data in cloud computing," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–5.

[33] J. Baek, R. Safavi-Naini, and W. Susilo, "Public key encryption with keyword search revisited," in *Proc. Int. Conf. Comput. Sci. Appl.* Berlin, Germany: Springer, Jun. 2008, pp. 1249–1259.

[34] B. Cui, Z. Liu, and L. Wang, "Key-aggregate searchable encryption (KASE) for group data sharing via cloud storage," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2374–2385, Aug. 2016.

[35] Z. Liu, T. Li, P. Li, C. Jia, and J. Li, "Verifiable searchable encryption with aggregate keys for data sharing system," *Future Gener. Comput. Syst.*, vol. 78, pp. 778–788, Jan. 2018.

[36] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "Dual-server public-key encryption with keyword search for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 4, pp. 789–798, Apr. 2016.

[37] J. Ning, J. Xu, K. Liang, F. Zhang, and E.-C. Chang, "Passive attacks against searchable encryption," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 3, pp. 789–802, Mar. 2019.

[38] J. Li, Y. Shi, and Y. Zhang, "Searchable ciphertext-policy attribute-based encryption with revocation in cloud storage," *Int. J. Commun. Syst.*, vol. 30, no. 1, p. e2942, Jan. 2017.

[39] H. Yin, Z. Qin, J. Zhang, H. Deng, F. Li, and K. Li, "A fine-grained authorized keyword secure search scheme with efficient search permission update in cloud computing," *J. Parallel Distrib. Comput.*, vol. 135, pp. 56–69, Jan. 2020.

[40] K. Liang and W. Susilo, "Searchable attribute-based mechanism with efficient data sharing for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 9, pp. 1981–1992, Sep. 2015.

[41] H. Yin, J. Zhang, Y. Xiong, L. Ou, F. Li, S. Liao, and K. Li, "CP-ABSE: A ciphertext-policy attribute-based searchable encryption scheme," *IEEE Access*, vol. 7, pp. 5682–5694, 2019.

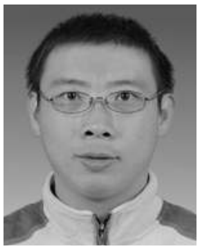
[42] H. Yin, Z. Qin, J. Zhang, L. Ou, F. Li, and K. Li, "Secure conjunctive multi-keyword ranked search over encrypted cloud data for multiple data owners," *Future Gener. Comput. Syst.*, vol. 100, pp. 689–700, Nov. 2019.

[43] Q. Huang and H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," *Inf. Sci.*, vols. 403–404, pp. 1–14, Sep. 2017.

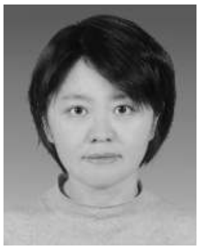
[44] D. J. Park, K. Kim, and P. J. Lee, "Public key encryption with conjunctive field keyword search," in *Proc. Int. Workshop Inf. Secur. Appl.* Berlin, Germany: Springer, Aug. 2004, pp. 73–86.

[45] M. Green, A. Akinyele, and M. Rushanan. (2004). *Libfenc: The Functional Encryption Library*. [Online]. Available: http://code.google.com/p/libfenc

YUANBO CUI received the B.S. degree in information and computing science from Weinan Teachers' University, in 2009, and the M.S. degree in applied math from the North China University of Technology, in 2013. He is currently pursuing the Ph.D. degree with the State Key Laboratory of Networking and Switching Technology, Network Security Research Center, Beijing University of Posts and Telecommunications (BUPT). His research interests include cryptography, location-based services, and security and privacy.



FEI GAO received the B.S. and Ph.D. degrees in cryptology from the Beijing University of Posts and Telecommunications, Beijing, China, in 2002 and 2007, respectively. He is currently a Professor and the Ph.D. Supervisor with the Beijing University of Posts and Telecommunications. His research interests include quantum cryptography protocol and its security analysis, quantum private query, and quantum key distribution.



YIJIE SHI received the Ph.D. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2016. She is currently a Lecturer with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications. Her research interests include industrial control system security and data privacy protection.

WEI YIN received the B.S. degree in mathematics and applied mathematics from Huaibei Normal University, Huaibei, Anhui, China, in 2012, and the Ph.D. degree in cryptography from the Beijing University of Posts and Telecommunications, Beijing, China, in 2019. He is currently with the National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing. His research interests include public key cryptography, lattice cryptography, and provable security.

EMMANOUIL PANAOUSIS (Member, IEEE) received the B.Sc. degree in informatics and telecommunications from the University of Athens, Greece, in 2006, and the M.Sc. degree in computer science from the Athens University of Economics and Business, Greece, in 2008, and the Ph.D. degree in mobile communications security from Kingston University London, U.K., in 2012. He was a Senior Lecturer of cyber security and privacy with the University of Brighton, an Invited Researcher with Imperial College London, a Postdoctoral Researcher with the Queen Mary University of London, and a Research and Development Consultant with Ubitech Technologies Ltd., and Surrey Research Park. He is currently an Associate Professor with the University of Surrey. His main research interests include cybersecurity and privacy engineering.

KAITAI LIANG (Member, IEEE) received the Ph.D. degree from the Department of Computer Science, City University of Hong Kong, in 2014. He is currently an Assistant Professor with the Department of Computer Science, University of Surrey, U.K. His research interests include applied cryptography and information security, in particular, encryption, network security, big data security, privacy enhancing technology, blockchain, lattice-based crypto, and security in cloud computing.

• • •