Towards Privacy-Preserving and Efficient Attribute-Based Multi-Keyword Search

Zhidan Li, Wenmin Li, Fei Gao, Wei Yin, Hua Zhang, Qiaoyan Wen, Kaitai Liang

Abstract—Searchable encryption can provide secure search over encrypted cloud-based data without infringing data confidentiality and data searcher privacy. In this work, we focus on a secure search service providing fine-grained and expressive search functionality, which can be seen as a general extension of searchable encryption and called attribute-based multi-keyword search (ABMKS). In most of the existing ABMKS schemes, the ciphertext size of keyword index (encrypted index) grows linearly with the number of the keyword associated with a file, so that the computation and communication complexity of keyword index is limited to $O(m)$, where m is the number of the keyword. To address this shortage, we propose the first ABMKS scheme through utilizing keyword dictionary tree and the subset cover, in such a way that the ciphertext size of keyword index is not dependent on the number of underlying keyword in a file. In our design, the complexity of computation and the complexity of the keyword index are at most $O(2\cdot \log(n/2))$ for the worst case, but $O(1)$ for the best case, where n is the number of keyword in a keyword dictionary. We also present the security and the performance analysis to demonstrate that our scheme is both secure and efficient in practice.

Index Terms—Searchable encryption, keyword dictionary tree, subset cover, attribute-based multi-keyword search, encrypted index.

1 Introduction

Being offered advanced cloud-based data storage [1], [2] and the cloud computing services [3], Internet data owners (DO) prefer to outsource the massive amount of data and expensive data computation to remote cloud. To rapidly occupy a piece of data outsourcing market, cloud service providers (CSP), such as Alibaba Cloud, Google AppEngine, and Dropbox, have prepared themselves to provide flexible and multi-functional data services for Internet data users (DU) to meet various needs, e.g., data sharing, retrieving, and cooperative computing [4], [5], [6]. However, data privacy and access control over the outsourced data must be guaranteed in practice. To this end, DO may be recommended to encrypt their data via some encryption techniques (such as AES, RSA) before outsourcing the data to CSP. This approach can protect the data privacy, security and somewhat data integrity. Nevertheless, it also brings side effects, at the same time, to DU and CSP. For example, if the outsourced data is encrypted, standard plaintext-based search mechanism may be applicable no more. This paper deals with the case where a privacy-preserving search over encrypted data is needed.

In order to address the search problem over encrypted data, searchable encryption (SE) has been proposed in the literature. Combining encryption with secure search, it can be used to protect privacy of search query but also the confidentiality of outsourced data. Specifically, it enables a valid data user to generate a trapdoor based on some specified query "description" for cloud server, so that the server can use the trapdoor to search over the encrypted data without violating data confidentiality and search privacy.

Song $et\ al.$ [7] seminally proposed the notion of keyword search over encrypted data, called Searchable Symmetric Encryption (SSE), in which users have to securely share key for data encryption. Later, SE was designed in the public-key context [13], [14], [15], which can resolve the problem of secret key distribution yielded by SSE. The follow-up research works [19], [20], [21], [22], [23], [25], [26], [27] were proposed to achieve various functionalities under the umbrella of public key based SE, for example, these works [19], [20], [21] focus on single keyword query, while the papers [25], [27] consider to enable users to make multi-keyword search.

In practice, DOs may prefer to maintain secure access control over their outsourced data, so that the unauthorized data user can not gain access to the data. One of the promising technologies called ciphertext-policy attribute-based encryption (CP-ABE) has been used to inject access control policy into data encryption. Zheng $et\ al.$ [19] proposed a new primitive called attribute-based keyword search (CP-ABKS) by integrating SE and ABE. In their CP-ABKS schemes, an access policy tree is associated with ciphertext and a private key is associated with user attributes. If the attributes satisfy the policy tree and the trapdoor matches the keywords ciphertext (encrypted indexes) simultaneously, then the ciphertext can be located and further decrypted. Some follow-up research works [26], [27] that are proposed to achieve multi-keyword search based on [19]. However, there is a limitation in most of the existing attribute-based multiple keyword search schemes (ABMKS), e.g., [26], [27]. Due to the number of keyword increases in the multi-keyword search, the size of encrypted index and the computational complexity are bounded by $O(m)$, where m is the number of keyword embedded in a file.

In this paper, inspired by the CP-ABKS schemes, we de-

- *Z. Li, W. Li, F. Gao, H. Zhang and Q. Wen are with State key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China.*
- *K. Liang is with Department of Computer Science, University of Surrey, Guildford GU2 7XH, UK*
- *W. Yin is with the National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100020, China*
- *Corresponding author: wqy@bupt.edu.cn*

sign a Towards Privacy-Preserving and Efficient Attribute-Based Multi-Keyword Search (TPPE-ABMKS) through utilizing keyword dictionary tree and the subset cover. Our TPPE-ABMKS can achieve multi-keyword search with fine-grained access control and the number of the encrypted index is relatively small, which is not dependent on the number of underlying keywords in a file. The contribution of this paper is summarized as follows.

1) We, for the first time, provide a secure multi-keyword search service with short-size ciphertext of keyword index. Our number of the encrypted index is not linear with the number of keyword embedded in a file. To the best of our knowledge, our design is the first of its type that achieves short-size ciphertext of keyword index in the model of MKS. More specifically, by designing the tools of keyword dictionary tree and the subset cover, in our scheme, the complexity of computation and the complexity of the encrypted index are at most $O(2\cdot \log (n/2))$ for the worst case but $O(1)$ for the best case, where n is the number of keyword in a keyword dictionary in system.
2) We show that our design is selectively secure against the chosen-keyword attack 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 retrieve are more efficient than that of the existing ABMKS schemes.

The rest of this paper is organized as follows. In Section 2, we present an overview of related work. In Section 3, we briefly review some basic primitives which will be used in this paper and we also define the main building blocks for our construction. We then present the system and threat models of our scheme, the construction, and the security model in Section 4. In Section 5, the proposed scheme is presented. Section 6 presents the security analysis of the prosed scheme. Section 7 shows the experimental analysis of the proposed scheme and the comparison with some related works. We conclude the paper in Section 8.

2 Related Work

SE was first proposed by Song $et\ al.$ [7], which allows DOs or DUs to generate a trapdoor that can be used by cloud server to search over the outsourced encrypted data. The existing SE schemes can be categorized into two branches: SEs in the symmetric-key context [7], [9], [10], [11], [12] and the ones in public-key scenario [13], [14], [15], [16], [17], [18]. In order to quickly obtain the encrypted record, researchers have been working on the keyword-based information retrieval to achieve various functionalities and practical needes. The research works in [9], [11] have been proposed SE in the multi-users setting, where DOs can enforce an access control policy by distributing some secret keys to authorized system users. Some SE schemes focus on improving the level of keyword security and privacy. Chen $et\ al.$ [23] proposed a public key encryption with keyword search in dual-server model, which can resist the keyword guessing attack. Later on, Liu $et\ al.$ [6] and Miao $et\ al.$ [27] presented the verifiable SE schemes by using public-auditing technique, respectively. Their schemes guarantee that DUs can prevent the semi-trusted CSP from tampering the sharing data and returning false search results.

Attribute-based encryption (ABE) [30], [31] is a useful data encryption tool for enforcing access control policy via cryptographic technique. It makes sure that if a user has a legitimate credential, then he/she can decrypt a given ciphertext which is encrypted according to an access control policy [30], [31]. With the special features of ABE, Zheng $et\ al.$ [19] proposed a new secure search service called attribute-based keyword search (ABKS). After that, [26], [27] were proposed to achieve extended functionalities (e.g., multi-keyword search) based on [19]. In ABKS, keywords are encrypted according to an access control policy, and a legitimate DU can generate trapdoors that can be used by server to search over the encrypted data. ABKS can maintain access control but also privacy-preserving keyword search in practice.

In order to enable DUs to make queries over multiple keywords at the same time, Golle $et\ al.$ [28] proposed the concept of conjunctive keyword search. Later on, Park $et\ al.$ [29] extended the notion into public key system. [26], [27] were put forth to achieve advanced functionalities in the multi-keyword search in public-key context. These schemes are denoted as attribute-based multiple keyword search (ABMKS) in this paper. However, the computation cost of keyword index generation is relatively high in [26], [27], which is growing linearly with the number of keyword embedded in a file. Recall that the complexity of keyword index ciphertext is $O(m)$, where m is the number of keyword embedded in a file.

3 Preliminaries

We give a brief review of some basic primitives which will be used in this paper in Sections 3.1 and 3.2. We also define the main building blocks of our scheme in Sections 3.3-3.6.

3.1 Bilinear Map

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p, g be a generator of the group $\mathbb{G}$, a bilinear mapping e: $\mathbb{G} \times \mathbb{G} \longrightarrow \mathbb{G}_T$ satisfies following properties [30]:

- 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: there is an efficient algorithm to compute $e(g^a, g^b)$, $\forall g \in \mathbb{G}$, $\forall a, b \in \mathbb{Z}_p^*$

3.2 Access Tree

Let $\mathcal{T}$ be an access tree [30] representing an access control policy. In $\mathcal{T}$, each non-leaf node denotes a threshold gate and is described by its a threshold value and children notes. Let num_v represent the number of children notes for a node v, and the children notes from left to right be labeled as $1, ..., num_v$. The k_v $(k_v \leq num_v)$ represents the threshold value for the node v, when k_v=1, the note v is an OR gate; otherwise 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.

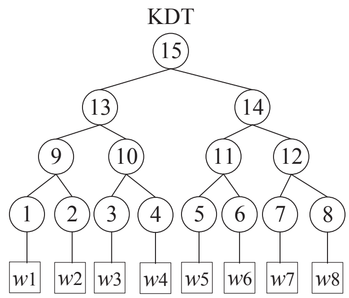


Fig. 1: Example of a Keyword Dictionary Tree

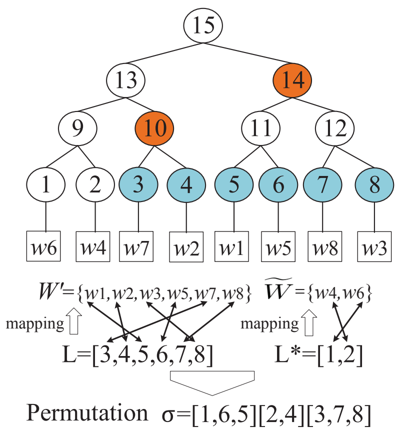


Fig. 2: Example of a Changed Dictionary Tree

In order to easy understand the access tree, we give some few functions. Let parent(v) denote 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) represent the label of the node v, and $\mathcal{T}_v$ represent the subtree of $\mathcal{T}$ rooted at node v.

$\mathcal{T}_v(\gamma)$=1 denotes that the attribute set γ meets 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.

3.3 Keyword Dictionary Tree

In this section, we first give the definition of the Keyword Dictionary Tree (KDT). Let $W = [w_1, ..., w_n]$ be the keyword dictionary in the system, KDT denotes a full binary tree with n leaves, and the leaf node L_i be associated with a keyword w_i as in Fig. 1 (hereafter, we set n=8 in all the examples to easy understand). In a KDT tree, from left to right, from bottom to top, the nodes of the tree is labeled with $1, ..., 2n-1$. We give an example as in Fig. 1. The bottom left node of the KDT holds number 1, and the root node of KDT holds number 15.

3.4 Changed Dictionary Tree and the Path note

We call the KDT as an original dictionary tree (ODT), if the leaf L_i of it is labeled with the keyword w_i. The changed dictionary tree (CDT) denotes that the positions of all the keywords in the ODT are changed. The CDT is defined as follows.

Let $W' = [w'_1, ..., w'_m]$ be the keyword set for one file f_i, and $\widetilde{W}$=W/W' denote the difference of sets W and W', e.g., as shown in Fig. 2, W=$[w_1, ..., w_8]$, W'=$[w_1, w_2, w_3, w_5, w_7, w_8]$, then $\widetilde{W}$=$[w_4, w_6]$. DO will choose $|W'|$ continuous leaf nodes of ODT, in which these leaf nodes are denoted as L, and L* is the remain of leaf nodes in ODT, e.g., as shown in Fig. 2, L=[3,4,5,6,7,8], and L*=[1,2].

For each leaf node in L and L*, it will be randomly labeled with one keyword in W' and $\widetilde{W}$, respectively, as shown in Fig. 2, $1 \to w_6$, $2 \to w_4$, $3 \to w_7$, $4 \to w_2$, $5 \to w_1$, $6 \to w_5$, $7 \to w_8$, and $8 \to w_3$. Thus, all of the keyword labeled with the leaf node of ODT is changed. We state that there exists a permutation σ=$[1, 6, 5][2, 4][3, 7, 8]$, where [1,6,5] indicates that leaf node 1 is labeled with keyword w_6, leaf node 6 is labeled with keyword w_5, and leaf node 5 is labeled with keyword w_1. The $[2, 4]$ and $[3, 7, 8]$ follow the same rule as $[1, 6, 5]$.

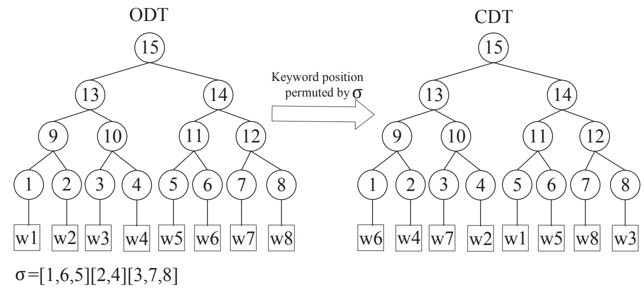


Fig. 3: The Generation of the Changed Dictionary Tree

We note that CDT is generated by changing the keyword order of the ODT with the permutation σ, and the σ has at least $2 \cdot m! \cdot (n-m)! = 2 \times [m \times (m-1) \times ... \times 2 \times 1] \times [(n-m) \times (n-m-1) \times ... \times 2 \times 1]$ different possibilities in total. Since n is a relatively large number, the σ will be very hard to be guessed (considering $(n-m)!$ or $m!$ is a very large number, e.g., $n = 1000$, $m = 100$). As shown in Fig. 3, we present an example of generation of CDT by using σ=$[1, 6, 5][2, 4][3, 7, 8]$.

Let $W^* = [w^*_1, ..., w^*_{m^*}]$ be a query keyword set, for each $w^*_i \in W^*$, let path$(w^*_i) = [\Omega_{w_i,1}, ..., \Omega_{w_i,l}]$ (where l =1+log n) be the path nodes of the keyword w^*_i, which starts from the leaf node labeled with w_i to the root node in CDT, e.g., in Fig.2, path$(w_4) = [2, 9, 13, 15]$.

3.5 Subset Cover

In this section, we give the definition of the subset cover for a keyword set W'. For the keyword set W' in the CDT, with the subset cover technique [32], DO selects root nodes of the minimum cover sets in the CDT tree that can cover all of the leaf nodes in W', denoted as cover$(W') = [\Omega_1, ..., \Omega_t]$, where t is the number of node in cover(W') and $t \leq 2\cdot$ log $(n/2)$, e.g., as in Fig 2. When $W' = [w_1, w_2, w_3, w_5, w_7, w_8]$, we have the cover$(W') = [10, 14]$. For each keywords W', we will prove that $t \leq 2\cdot$log $(n/2)$ as follows.

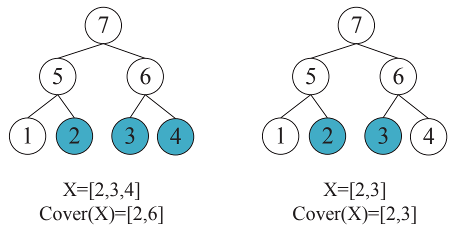
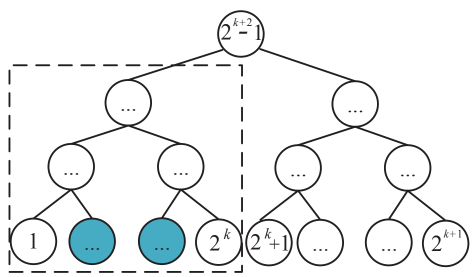
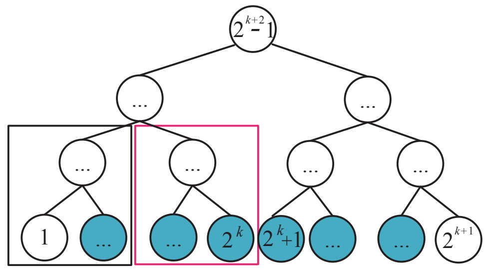


Fig. 4: An Example when n=2^2=4



Case 1:keyword W' labeled with right(or left) half nodes of CDT



Case 2:keyword W' labeled with right and left half nodes of CDT

Fig. 5: An Example when n=2^{k+1}

Theorem 1. *In the CDT, for each keyword set W' for a file, we have that $t \leq 2\cdot$ log $(n/2)$, where t is the number of node in cover(W'), and when $t = 2\cdot$ log $(n/2)$, the half of nodes in the subset cover(W') are labeled with the right half nodes of the CDT, and the remaining half of nodes in cover(W') are labeled with the left half nodes of the CDT.*

Proof. We will prove it by using mathematical induction.

First, when n=4, as shown in Fig. 4, we know that the maximum value of t is 2≤2=2·log (4/2), Theorem 1 holds.

Second, assuming that the theorem 1 holds when $n = 2^k$, then

$$t \leq 2\cdot \text{ log } (n/2) = 2\cdot \text{ log } (2^k/2) = 2(k-1)$$

It can be deduced that the Theorem 1 holds when $n = 2^{k+1}$.

Now we prove that the Theorem 1 holds when $n = 2^{k+1}$.

As shown in Fig. 5, in case 1, the keyword W' are continuously labeled with only right (or left) half nodes of CDT. We find that it is same as the situation that $n' = 2^k$ (The CDT with $n' = 2^k$ leaf nodes in dotted rectangle), then

$$t \leq 2\cdot\text{log } (n'/2) = 2\cdot\text{log } (2^k/2) = 2(k-1) < 2k = 2\cdot\text{log } (2^{k+1}/2)=2\cdot\text{log } (2^n/2).$$

As shown in Fig. 5, in case 2, the keyword W' are continuously labeled with both right and left half of CDT, the keywords in W' that are labeled with the left half leaf nodes of CDT denoted as W_{left}, and the keywords in W' that are labeled with the right half leaf nodes of CDT denoted as W_{right}.

Consider the keyword W_{left} and cover(W_{left}) (where t_{left} denotes the number of nodes in cover(W_{left})). Because W' are continuously labeled with the leaf nodes in CDT, we note that t_{left}≤1+log$(2^k/2)$=k (As shown in Fig. 5, the number of cover note in the black rectangle is log $(2^k/2)$, and the number of cover note in the red rectangle is 1). Similarly, we have t_{right}≤1+log$(2^k/2)$=k. Therefore, we have t=t_{left}+t_{right}≤k+k=2k=2·log $(2^{k+1}/2)$=2·log $(n/2)$.

Therefore the Theorem 1 is proved. □

Theorem 2. *In the CDT, for each keyword set W', the number of node in the subset cover(W')=$t \leq m$, where t is the number of node in cover(W'), m is the number of keyword in keyword set W'.*

Proof. As the example shown in Fig. 2, recall that the keywords in W' are continuously labeled with the leaf nodes in CDT, then there must exist two keywords that have the same cover node in CDT. Thus, it is easy to prove that $t \leq m$. □

Theorem 3. *From Theorem 1 and Theorem 2, we have that t≤ $Min\{2\cdot log(n/2), m\}$*

3.6 Key Idea of Keyword Index Ciphertext Generation and Match

In this section, we present the idea of keyword index ciphertext generation, and the process of whether a query keyword set W^* matches the keyword set $W' = [w_1', ..., w_m']$ embedded in a file.

When a DO wants to share a file, and the keywords set W' is the keyword set embedded in it. First, DO generates a permutation σ with W' as in the Section 3.4, and obtains the cover(W') as in the Section 3.5. Then DO chooses an access policy tree $\mathcal{T}$, and encrypts the permutation σ based on the access policy tree by using ABE scheme.

$$\text{cover}(W')=[\Omega_1, ..., \Omega_t]$$

Note that DO will use all the elements of cover(W') to build the keyword index ciphertext instead of W'. We will give more details of it in the Encrypt algorithm in Section 5. Thus the complexity of computation and size is reduced from m to t. As in the Theorem 3, we have proved that $t \leq Min\{2\cdot \text{ log}(n/2), m\}$.

Let $W^* = [w_1^*, ..., w_{m^*}^*]$ be the query keyword set chosen by a data user. If the data user's attributes set satisfies the access policy tree $\mathcal{T}$, then he/she can get the permutation σ, which means that he/she is allowed to generate the CDT as in the Section 3.4. Thus, for each $w_i^* \in W^*$, the data user can get the path(w_i^*) from the CDT.

$$\text{path}(w_i^*) = [\Omega_{w_i^*,1}, ..., \Omega_{w_i^*,l}], \text{ where } l = 1 + \text{log } n$$

To verify whether the query keyword set $W^* \subseteq W'$, we need to check that whether there is an element in both path(w_i^*) and cover(W'), for each $w_i^* \in W^*$. We present this theorem by the following formula.

$$W^* \subseteq W' \Longleftrightarrow \text{path}(w_i^*) \bigcap \text{cover}(W') \neq \emptyset, \forall w_i^* \in W^*.$$

Otherwise, if $W^* \not\subseteq W'$, then there exists one keyword w_i satisfies path$(w_i^*) \bigcap$ cover$(W') = \emptyset$. For example, as in Fig. 2, if W'=$[w_1, w_2, w_3, w_5, w_7, w_8]$, W^*=$[w_1, w_2, w_5, w_8]$, we see that

path$(w_1) = [5, 11, \mathbf{14}, 15]$, path$(w_2) = [4, \mathbf{10}, 13, 15]$,
path$(w_5) = [6, 11, \mathbf{14}, 15]$, path$(w_8) = [7, 12, \mathbf{14}, 15]$,
cover$(W') = [\mathbf{10}, \mathbf{14}]$.

Note that

path$(w_1) \bigcap$ cover(W')=14,
path$(w_2) \bigcap$ cover(W')=10,
path$(w_5) \bigcap$ cover(W')=14,
path$(w_8) \bigcap$ cover(W')=14.

Remark. Using the above method to generate keyword index ciphertext, the complexity of keyword index ciphertext and the computation complexity is at most $O(2{\cdot}\log(n/2))$ for the worst case, but we state that the complex is $O(1)$ for the best case (In this case, the number of node in the cover(W') is only 1), where n is number of the keywords in keyword dictionary W. It is to say the number of the keyword index ciphertext can be reduced to only one.

4 Problem Formulations

In this section, we present the system and threat models, the scheme construction, and the security model.

4.1 System and Threat Models

In Fig. 6, we present the system model of our scheme, which involves four entities: DO, DU, CSP and Authority.

The DO encrypts the data files F as well as corresponding keyword index 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 will generate a trapdoor by his specified query keywords and submit it to the CSP. The CSP will retrieve the DO's encrypted data according to the specified keywords by the corresponding trapdoor, if the user's attributes satisfy the access policy and the trapdoor matches the keyword index. The role of Authority is to issue credentials (PK/SK) to data owners/users, and the credentials are sent over secure communication channel.

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

4.2 The Construction of TPPE-ABMKS Scheme

In this section, we present the overview of our scheme which consists of eight algorithms.

- $\mathbf{Setup}(\lambda)$. The Setup algorithm takes as input the security parameter λ. It initializes the global system parameter, and outputs the master key MSK and the public key PK.

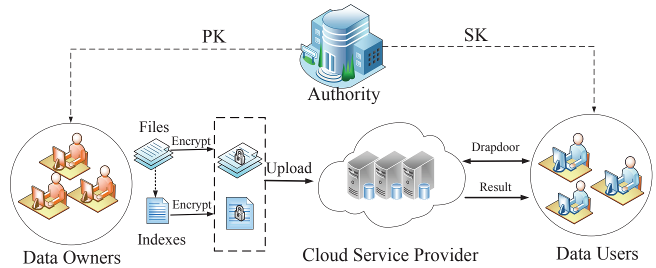


Fig. 6: Basic Framework of Our Scheme

- $\mathbf{KeyGen}(PK, MSK, S)$. The KeyGen algorithm takes as input data user's attribute set S, the authority uses the MSK to generate the private key SK for the data user.
- $\mathbf{Encrypt}(PK, \mathcal{T}, \text{ODT}, F, W')$. The Encrypt algorithm takes as input files set F and the keyword dictionary W. For each file $f \in F$, DO generates the permutation σ by using the keyword set W' corresponding to the file f, then encrypts the file f by using the symmetric key K to get the ciphertext C, encrypts the σ and K by using ABE to get the ciphertext I_1, generates the encrypted index I_2 according to W'. At last sends the ciphertext $CT = \{C, I_1, I_2\}$ to the CSP.
- $\mathbf{GenTK}(PK, SK)$. The GenTK algorithm takes as input public key PK and the private key SK for user's attribute set S, user generates the transformation key TK and the corresponding retrieving key RK.
- $\mathbf{Transform}(CT, TK)$. The Transform algorithm takes as input the ciphertext CT and the transformation key TK, it outputs a partially decrypted ciphertext $CT' = \{CT_1', CT_2'\}$.
- $\mathbf{Decrypt}(CT_1', RK)$. The Decrypt algorithm takes as input the transformed ciphertext CT_1' and the retrieving key RK, it outputs the permutation σ and the symmetric key K.
- $\mathbf{Trapdoor}(PK, SK, RK, W^*, \text{ODT}, \sigma)$. The trapdoor algorithm takes as input the public key PK, private key SK and corresponding key RK, queried keyword set W^*, original dictionary tree ODT, the permutation σ, it outputs the trapdoor T_{W^*} according to the query keyword W^* and the data user's attribute set S.
- $\mathbf{Retrieve}(PK, CT', T_{W^*})$. The Retrieve algorithm takes as input the ciphertext CT' and the trapdoor T_{W^*}. CSP checks whether T_{W^*} satisfies the ciphertext I_1 and the encrypted index I_2. If it holds, then returns the corresponding C to user, otherwise, outputs $\perp$.

4.3 Security Model

In this section, we present the security model of our scheme. As described in the threat model, only the CSP is honest-but-curious. Intuitively, indistinguishability against chosen-keyword attack (IND-CKA) means that the CSP (an adversary $\mathcal{A}$) can learn nothing information about keyword

TABLE 1: Notation in TPPE-ABMKS

Notation	Description
S	DU's attribute set
SK	DU's private key corresponding to S
F	The file set of DO
W	The keyword dictionary in a system
W'	The keyword set embedded in one file f
W^*	The query keyword set chosen by DU
$\mathcal{T}$	The access policy tree for one file f
T_{W^*}	The trapdoor corresponding to W^*
K	The symmetric key for one file f
C	The ciphertext of one file f encrypted by K
I_1	The ciphertext of σ and K
I_2	The ciphertext of keyword index W'
n	The number of keyword in W
m	The number of keyword in W'
m^*	The number of keyword in W^*
t	The number of nodes in the cover(W')
Y	The leaf nodes set in access tree $\mathcal{T}$

set plaintext of the keywords set ciphertext except for the search tokens and the result. We present the security model by utilizing the indistinguishability against chosen-keyword attack (CKA) game as follows.

Definition 1. <u>*IND-CKA Game:*</u>

- *Setup: The challenger $\mathcal{C}$ runs the setup algorithm to generate the public parameters PK and the master key MSK, and gives the public parameters PK to the adversary $\mathcal{A}$. Note that the master key MSK is kept by the challenger $\mathcal{C}$. 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 polynomial time, 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) *$\mathcal{O}_{Trapdoor}(SK, W^*)$: On input a set of keyword W^* and the SK, the challenger $\mathcal{C}$ runs the Trapdoor algorithm to get T_{W^*} and sets L_{kw}=$L_{kw} \bigcup W^*$, 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 keyword set W_1^* and W_2^*, where W_1^*, $W_2^* \notin L_{kw}$, it means that W_1^*, W_2^* cannot be queried in $\mathcal{O}_{Trapdoor}$. Then, the challenge $\mathcal{T}$ picks a random $b \in \{0, 1\}$ and encrypts W_b^* as CT^* by using Encrypt algorithm. Finally, $\mathcal{C}$ returns CT^* to the adversary $\mathcal{A}$.*
- *Phase 2: $\mathcal{A}$ continues to query the oracles as in Phase 1, but the restriction is that (S, W_1^*) and (S, W_2^*) 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 IND-CKA game if $b' = b$; otherwise, it fails.*

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

Definition 2. *Our scheme is IND-CKA secure if the advantage of any PPT $\mathcal{A}$ winning the IND-CKA game is negligible.*

Besides, our scheme is selectively IND-CKA secure if an Int stage is added before the Setup algorithm where the adversary $\mathcal{A}$ claims the two keyword sets W_1^*, W_2^* which to be attacked.

5 Our Design

We present some notations used in our construction in the TABLE 1 and our design below.

- **Setup**(λ)→(PK, MSK): This algorithm is executed by authority, given a security parameter λ, the authority first chooses a bilinear group $\mathbb{G}$ of prime order p with generator g. It then chooses two random numbers α_1, $\alpha_2 \in \mathbb{Z}_p^*$ and two hash functions H_1:$\{0, 1\}^*$→$\mathbb{G}$, H_2:$\{0, 1\}^*$→$\mathbb{Z}_p^*$. Finally, it generates the public key PK and master key MSK as follows:
$$PK = \{\mathbb{G}, g, h = g^{\alpha_2}, e(g, g)^{\alpha_1}\}$$
$$MSK = \{\alpha_2, g^{\alpha_1}\}$$
- **KeyGen**(PK, S, MSK)→SK: This algorithm is executed by authority, given a DU's attribute set S, the authority first chooses a random number $r \in \mathbb{Z}_p^*$, and randomly chooses $r_i \in \mathbb{Z}_p^*$ for each attribute $i \in S$. Finally, it outputs DU's private key SK as follows:
$$SK = \{D = g^{(\alpha_1 + r)/\alpha_2},$$
$$\forall i \in S : D_i = g^r H_1(i)^{r_i}, D_i' = g^{r_i}\}$$
- **Encrypt**$(PK, \mathcal{T}$,ODT$, F, W)$→CT: This algorithm is executed by DO. Let $F = \{f_1, ..., f_N\}$ be the file set to be shared, in order to easy understand, we encrypt one file $f \in F$ to explain the Encrypt algorithm, let the $W = [w_1, ..., w_n]$ be the keyword dictionary, and the $W' = [w_1', ..., w_m']$ be the keyword set for f.

  DO will encrypt the f under the corresponding symmetric key K to generate the ciphertext C (e.g., Using AES to encrypt the f, this is out of the scope of our discusses).

  After that, DO will generate the σ and cover(W') for file f as follows.

  1) As in Section 3.4, DO will randomly generate a permutation σ by using W'.
  2) As in Section 3.4, DO will generate the CDT by using σ and ODT.
  $$\text{CDT} \xleftarrow{\sigma} \text{ODT}$$
  3) As in Section 3.5, DO will get the cover(W') corresponding to W'.
  $$\text{cover}(W')\text{=}[\Omega_1, ..., \Omega_t]$$

  As shown in the Theorem 3, we state that $t \leq Min\{2 \cdot \log\,(n/2), m\}$. For example, as shown in the Fig. 2, when $W' = [w_1, w_2, w_3, w_5, w_7, w_8]$, and $\sigma =$

$[1, 6, 5][2, 4][3, 7, 8]$, then the cover$(W') = [10, 14]$.

Then, DO computes the ciphertext I_1 and I_2 for the file f as follows:

1) For each node v in the in the access policy tree $\mathcal{T}$, choose a polynomial q_v in a top-down manner, the degree d_v of q_v is $k_v - 1$, where k_v is the threshold value of the node v. For the root node R of $\mathcal{T}$, randomly choose $s \in \mathbb{Z}_p^*$ and set $q_R(0) = s$, then randomly choose d_R other points to build the polynomial q_R. For the non-root node v, set $q_v(0) = q_{parent(v)}(index(v))$ and randomly choose d_v other points to build the polynomial q_v.
Let Y be the set of leaf nodes in the access tree $\mathcal{T}$ and compute $I_1 = \{I_{\sigma||K}, \{\theta, \theta_y, \theta'_y\}\}$
$$I_{\sigma||K} = (\sigma||K) \cdot e(g, g)^{\alpha_1 s}.$$
$$\theta = h^s, \theta_y = g^{q_y(0)}, \theta'_y = H_1(att(y))^{q_y(0)}, \forall y \in Y.$$
2) Compute $I_2 = \{\text{cover}(W'), \{I_{\Omega_i}\}_{i=1}^{i=t}\}$ with the permutation σ and the cover$(W') = [\Omega_1, ..., \Omega_t]$
$$I_{\Omega_i} = e(g, g)^{\alpha_1 s H_2(\Omega_i||\sigma||K)}, i \in [1, t].$$

Note that in order to encrypt the $\sigma||K$, the item $\sigma||K$ will be encoded into the forma of the element in group $\mathbb{G}_T$, it is out of the scope of our discusses.

Finally, outputs $CT = \{C, I_1, I_2\}$ for the file f.

- **GenTK**$(PK, SK) \rightarrow TK$: This algorithm is executed by DU, it takes into the public key PK and the user's private key $SK = \{D = g^{(\alpha_1+r)/\alpha_2}, D_i = g^r H_1(i)^{r_i}, D'_i = g^{r_i}\}$ for a set of attributes S. It chooses a random value $u \in \mathbb{Z}_p^*$, and computes transformation key TK and the corresponding retrieving key RK:
$$TK = \{S, D^* = D^u, D_i^* = D_i^u, {D'_i}^* = {D'_i}^u\}$$
$$RK = u$$
- **Transform**$(TK, CT) \rightarrow (CT' \& \perp)$: This algorithm is executed by cloud service provider (CSP), after getting user's attributes set S from TK, then CSP checks whether the S can meet $\mathcal{T}$. If S can not meet $\mathcal{T}$, it outputs $\perp$; otherwise, it runs the algorithm as follows:

1) If the node x is a leaf node in the $\mathcal{T}$, we let i=$att(x)$ and define as follows: if $i \in S$, then compute θ_x as follows;
$$\begin{aligned}\theta_x &= \frac{e(D_i^*, \theta_x)}{e({D'_i}^*, \theta_{x'})} \\ &= \frac{e(g^{ru} \cdot H_1(i)^{r_i u}, g^{q_x(0)})}{e(g^{r_i u}, H_1(i)^{q_x(0)})} \\ &= e(g, g)^{ru \cdot q_x(0)}\end{aligned} \tag{1}$$
2) If the node x is a no-leaf node in the $\mathcal{T}$, we get the θ_x by computing $\theta_{x'}$ in a recursive manner, where x' is the children nodes of x. The S_x is an arbitrary k_x set of children nodes x, if there exists no such a set, set $\theta_{x'}$=$\perp$; otherwise, compute $\theta_{x'}$ as
$$\begin{aligned}\theta_x &= \prod_{x' \in S_x} \theta_{x'}^{\Delta_{i,S'_x}(0)} \\ &= \prod_{x' \in S_x} \left(e(g, g)^{ru \cdot q_{x'}(0)}\right)^{\Delta_{i,S'_x}(0)} \\ &= \prod_{x' \in S_x} \left(e(g, g)^{ru \cdot q_{parent(z)}(index(z))}\right)^{\Delta_{i,S'_x}(0)} \\ &= \prod_{x' \in S_x} \left(e(g, g)^{ru \cdot q_x(i)}\right)^{\Delta_{i,S'_x}(0)} \\ &= e(g, g)^{ru \cdot q_x(0)}\end{aligned} \tag{2}$$
where i=index(x'), S'_x=$\{$index$(x') : x' \in S_x\}$
If the tree is satisfied by S, we can get that A=θ_{root}=$e(g, g)^{ru \cdot q_R(0)}$=$e(g, g)^{rus}$, and compute the partially-decrypted ciphertext pdc as
$$\begin{aligned}pdc &= e(\theta, D^*)/A \\ &= e(h^s, g^{(\alpha_1+r) \cdot u/\alpha_2})/e(g, g)^{rsu} \\ &= e(g, g)^{\alpha_1 su}\end{aligned} \tag{3}$$

Then output $CT' = \{CT'_1, CT'_2\}$, and return CT'_1 to DU.
$$CT'_1 = \{I_{\sigma||K}, pdc\}$$
$$CT'_2 = \{C, I_2\}$$

- **Decrypt**$(RK, CT'_1) \rightarrow (\sigma||K)$: This algorithm is executed by DU, on receiving the CW' from CSP, then obtain the σ and K by using retrieve key RK as
$$\begin{aligned}I_{\sigma||K}/pdc^{\frac{1}{RK}} &= (\sigma||K) \cdot e(g, g)^{\alpha_1 s}/(e(g, g)^{\alpha_1 su})^{\frac{1}{u}} \\ &= \sigma||K\end{aligned}$$
Thus, the DU gets the permutation σ and the symmetric key K corresponding to the file F.

- **Trapdoor**$(PK, SK, RK, W^*, \text{ODT}, \sigma) \rightarrow T_{W^*}$: When a user wants to issue a search query according to keyword set W^*=$[w_1^*, ..., w_{m^*}^*]$, where m is the number of the keywords included in W^*.
After getting the permutation σ and K, DU will generate the CDT and $\{\text{path}(w_i^*)\}_{i=1}^{i=m^*}$ with ODT and σ as follows.

1) CDT$\xleftarrow{\sigma}$ODT as in Section 3.4.
2) For each $w_i^* \in W^*$, obtain the path nodes path(w_i^*) as in Section 3.4.
path(w_i^*)=$[\Omega_{w_i^*,1}, ..., \Omega_{w_i^*,l}]$, l=1+log n.

Then, compute T_1 and T_2 as follows:
$$\begin{aligned}T_1 &= \{S, D^* = D^{RK}, D_i^* = D_i^{RK}, {D'_i}^* = {D'_i}^{RK}\} \\ &= \{S, D^* = D^u, D_i^* = D_i^u, {D'_i}^* = {D'_i}^u\}\end{aligned}$$
$$\begin{aligned}T_2 &= \{path(w_i^*), \widehat{T}_{\Omega_{w_i^*,j}} = \frac{H_2(\Omega_{w_i^*,j}||\sigma||K)}{RK}\} \\ &= \{path(w_i^*), \widehat{T}_{\Omega_{w_i^*,j}} = \frac{H_2(\Omega_{w_i^*,j}||\sigma||K)}{u}\}\end{aligned}$$
for $i \in [1, m^*]$, $j \in [1, 1+\log n]$.

Note that $T_1 = TK$ is computed by DU as in the GenTK phase.

Finally, return $T_{W^*} = \{T_1, T_2\}$.

- **Retrieve**$(PK, CT', T_{W^*}) \rightarrow (C\&\perp)$: This algorithm is executed by CSP, and it works as follow. If S does satisfy $\mathcal{T}$, the CSP will terminate the search process; otherwise, the CSP works as follows.

  For each path(w_i^*) for the keyword $w_i^* \in W^*$, if there exists one case that cover$(W') \bigcap$ path$(w_i^*) = \emptyset$, CSP breaks the retrieve and outputs $\perp$. It is because that $W^* \subseteq W' \iff$ path$(w_i^*) \bigcap$ cover$(W') \neq \emptyset$, $\forall w_i \in W^*$ as described in Section 3.6.

  Otherwise, we let Ω_i=cover$(W') \bigcap$ path(w_i^*), $i \in [1, m^*]$. Check

$$\prod_{i=1}^{i=m^*} I_{\Omega_i} \stackrel{?}{=} \prod_{i=1}^{i=m^*} pdc^{\widehat{T}_{\Omega_i}} \tag{4}$$

  If it does not hold, return $\perp$; otherwise, it means that Trapdoor T_{W^*} matches the keyword index ciphertext I, then send the corresponding C to the user. Once gaining all the search results from CSP, the user can decrypt them by using the corresponding symmetric key K.

Correctness analysis. If the user's attribute set S satisfies the access policy tree $\mathcal{T}$ and the queried keyword set satisfies $W^* \subseteq W$, we have that

$$\prod_{i=1}^{i=m^*} I_{\Omega_i} = \prod_{i=1}^{i=m^*} e(g,g)^{\alpha_1 s H_2(\Omega_i||\sigma||K)} \tag{5}$$

$$\begin{aligned}\prod_{i=1}^{i=m^*} pdc^{\widehat{T}_{\Omega_i}} &= \prod_{i=1}^{i=m^*} (e(g,g)^{\alpha_1 s u})^{\widehat{T}_{\Omega_i}} \\ &= \prod_{i=1}^{i=m^*} (e(g,g)^{\alpha_1 s u})^{\frac{H_2(\Omega_i||\sigma||K)}{u}} \\ &= \prod_{i=1}^{i=m^*} (e(g,g)^{\alpha_1 s H_2(\Omega_i||\sigma||K)}\end{aligned} \tag{6}$$

Therefore, we can state that the Eq. (4) holds if $W^* \subseteq W$.

6 Security Analysis

In this section, we present the security analysis of our scheme proposed in Section 5. Our scheme is secure based on the following theorems.

Theorem 4. *Given the random oracle H_1 and the one-way hash function H_2, the TPPE-ABMKS scheme is secure against IND-CKA in the generic bilinear group model.*

Proof. In the IND-CKA game, $\mathcal{A}$ wants to distinguish $e(g,g)^{\alpha_1 s H_2(\Omega_i||\sigma_0)}$ from $e(g,g)^{\alpha_1 s H_2(\Omega_i||\sigma_1)}$. Given $\eta \in \mathbb{Z}_p^*$, the advantage of distinguishing $e(g,g)^{\alpha_1 s H_2(\Omega_i||\sigma_0)}$ from $e(g,g)^{\eta}$ is same as the advantage of distinguishing $e(g,g)^{\eta}$ from $e(g,g)^{\alpha_1 s H_2(\Omega_i||\sigma_1)}$. Then, if $\mathcal{A}$ has a advantage ϵ to break the IND-CKA game, then it has the advantage $\epsilon/2$ in distinguishing $e(g,g)^{\alpha_1 s H_2(\Omega_i||\sigma_0)}$ from $e(g,g)^{\eta}$. We present the following IND-CKA game as follows.

Setup : The challenger $\mathcal{C}$ randomly chooses α_1, $\alpha_2 \leftarrow \mathbb{Z}_p^*$ and outputs public key $(e, g, H_1, H_2, h = g^{\alpha_2}, e(g,g)^{\alpha_1})$. $\mathcal{A}$ chooses an access policy tree $\mathcal{T}^*$ and sends it to $\mathcal{C}$.

$H_1(att_j)$ is simulated as follows: If att_j has not been queried before, the challenger $\mathcal{C}$ chooses r_j^* from $\mathcal{Z}_p$, adds (att_j, r_j^*) to $\mathcal{O}_{H_1}$ and outputs $g^{r_j^*}$; otherwise the challenger $\mathcal{C}$ returns $g^{r_j^*}$ by only retrieving r_j^* from $\mathcal{O}_{H_1}$.

Phase1 : $\mathcal{A}$ can query the following oracles:

1) $\mathcal{O}_{KeyGen}(S)$: The challenger $\mathcal{C}$ randomly chooses r^* from $\mathbb{Z}_p^*$ and computes $D = g^{(\alpha_1 + r^*)/\alpha_2}$. For each attribute $att_j \in S$, the challenger $\mathcal{C}$ randomly chooses $r_j^* \in \mathbb{Z}_p$, then computes $D_j = g^{r^*} \cdot H_1(att_j)^{r_j^*}$ and $D_j' = g^{r_j^*}$. Finally, it returns the secret key $SK_S = (D, \{D_j, D_j'\})$ to the adversary $\mathcal{A}$.
2) $\mathcal{O}_{GenTK}(SK)$: The challenger $\mathcal{C}$ searches the S in the set D. If there exists the tuple, it randomly chooses $u \in \mathbb{Z}_p$ and generates the TK_S and RK,
$$TK_S = SK_S^u$$
$$RK = u$$
otherwise, the challenger $\mathcal{C}$ runs the KeyGen algorithm to get SK_S, and runs GenTK to get TK_S.
3) $\mathcal{O}_{Trapdoor}(SK, W^*)$: The challenger $\mathcal{C}$ generates the Trapdoor $T_{W^*} = (T_1, T_2)$, where $T_1 = TK_S = SK_S^{RK}$, $RK = u$.
The T_2 is generated as follows: The challenger chooses a permutation σ and generates the CDT by σ as in Section 3.4, for each keyword $w_i \in W^*$, generates the path$(w_i^*) = [\Omega_{w_i^*,1}, ..., \Omega_{w_i^*,l}]$, where path$(w_i^*)$ represents the path nodes set, which consists of the nodes originate from the leaf node associated with the keyword w_i^* up to the root node of the tree CDT, then computes $\widehat{T}_{W_{\Omega_{w_i^*,j}}} = \frac{H_2(\Omega_{w_i^*,j}||\sigma)}{u}$, $i \in [1, m]$, $j \in [1, \log\ n]$, and sets $T_2 = \{\Omega_{w_i^*}, \widehat{T}_{W_{\Omega_{w_i^*,j}}}\}$. If the attributes set S satisfies the policy tree $\mathcal{T}^*$, the challenger adds the keyword set W^* to the keyword list L_{kw}.

Challenge phase : The $\mathcal{A}$ give two keyword sets W_0^*, W_1^* to be challenged on, where W_0^*, $W_1^* \notin L_{kw}$ and the length of W_0^* is equal to W_1^*, the challenger randomly chooses $s \in \mathbb{Z}_p$, a symmetric key K and computes secret shares of s for each leaves in $\mathcal{T}^*$. The challenger chooses $\lambda \leftarrow \{0, 1\}$ and generates a permutation σ_λ corresponding to W_λ^*. If $\lambda = 0$, it runs the Encrypt algorithm to generate the ciphertext corresponding to keyword set W_λ^* and outputs

$I_{\sigma_\lambda} = (\sigma_\lambda||K)e(g,g)^{\alpha_1\eta}$, $\theta = h^{\eta}$,
$\{I_i = e(g,g)^{\alpha_1 \eta H_2(\Omega_i||\sigma_\lambda||K)}\}_{i\in[1,t]}$,
$\{\theta_y = g^{q_y(0)}, \theta_y' = H_1(att(y))^{q_y(0)}\}_{\forall y \in Y}$ by selecting $\eta \in \mathbb{Z}_p^*$;

otherwise, it outputs

$I_{\sigma_\lambda} = (\sigma\_\lambda||K)e(g,g)^{\alpha_1 s}$,$\theta = h^s$,
$\{I_i = e(g,g)^{\alpha_1 s H_2(\Omega_i||\sigma_\lambda||K)}\}_{i\in[1,t]}$,
$\{\theta_y = g^{q_y(0)}, \theta_y' = H_1(att(y))^{q_y(0)}\}_{\forall y \in Y}$

Phase2 : This phase is like Phase 1, but the restriction is that W_0^*, W_1^* have not been issued in Phase 1.

Guess: The adversary $\mathcal{A}$ outputs a guess for $\lambda' \in \{0, 1\}$. If $\lambda' = \lambda$, $\mathcal{A}$ wins the IND-CKA game; otherwise, it fails. We can note that if $\mathcal{A}$ can construct $e(g,g)^{\chi\alpha_1 s H_2(\Omega_i||\sigma_\lambda||K_i)}$ by using the term $e(g,g)^{\chi}$ contained in the aforementioned oracles, then $\mathcal{A}$ can use it to distinguish $e(g,g)^{\eta}$ from

TABLE 3: Computational Cost Comparison

Scheme	PAB-MKS	Ours
KeyGen	$(2s{+}2)\cdot E_{\mathbb{G}}{+}s\cdot H_1$	$(2s{+}2)\cdot E_{\mathbb{G}}{+}sH_1$
Encrypt	$(m{+}2\|Y\|{+}3)\cdot E_{\mathbb{G}}{+}\|Y\|\cdot H_1{+}m\cdot H_2$	$(t{+}1)\cdot E_{\mathbb{G}_T}{+}(2\|Y\|{+}1)\cdot E_{\mathbb{G}}{+}\|Y\|\cdot H_1{+}t\cdot H_2$
Trapdoor	$(2s{+}3)\cdot E_{\mathbb{G}}{+}m^*\cdot H_2$	$(2s{+}1)\cdot E_{\mathbb{G}}{+}(m^*\cdot\log n)\cdot H_2$
Retrieve	$(2s{+}4)\cdot P{+}s\cdot E_{\mathbb{G}_T}$	$(2s{+}1)\cdot P{+}(s{+}2m^*\text{-}1)\cdot E_{\mathbb{G}_T}$

TABLE 2: Notation used in performance evaluation

Notation	Description
P	The bilinear pairing operation
$E_{\mathbb{G}}$	The exponentiation operation in group $\mathbb{G}$
$E_{\mathbb{G}_T}$	The exponentiation operation in group $\mathbb{G}_T$
H_1	Map a bit-string to an element of $\mathbb{G}$
H_2	Map a bit-string to an element of $\mathbb{Z}_p$
$\|\mathbb{G}\|$	The element length in $\mathbb{G}$
$\|\mathbb{G}_T\|$	The element length in $\mathbb{G}_T$
$\|\mathbb{Z}_p\|$	The element length in $\mathbb{Z}_p$
s	The number of a DU's attributes
n	The number of keyword in W
m	The number of keyword in W'
m^*	The number of keyword in W^*
t	The number of nodes in the cover(W')
$\|Y\|$	The number of leaf nodes in access tree $\mathcal{T}$

TABLE 4: Storage Cost Comparison

Scheme	PAB-MKS	Ours
KeyGen	$(2s{+}1)\cdot\|\mathbb{G}\|$	$(2s{+}1)\cdot\|\mathbb{G}\|$
Encrypt	$(m{+}2\|Y\|{+}3)\cdot\|\mathbb{G}\|$	$(t{+}1)\cdot\|\mathbb{G}_T\|{+}(2\|Y\|{+}1)\cdot\|\mathbb{G}\|$
Trapdoor	$(2s{+}3)\cdot\|\mathbb{G}\|$	$(2s{+}1)\cdot\|\mathbb{G}\|{+}(m^*\cdot\log n)\cdot\|\mathbb{Z}_p\|$

$e(g,g)^{\alpha_1 s H_2(\Omega_i||\sigma_0)}$. Therefore, we need to prove that the adversary $\mathcal{A}$ can rebuilt $e(g,g)^{\delta\alpha_1 s H_2(\Omega_i||\sigma_\lambda)}$ by using the term $e(g,g)^{\delta}$ with a negligible probability. It means that $\mathcal{A}$ cannot gain non-negligible advantage in the IND-CKA game. As $\alpha_1 s$ can be rebuilt by using $(\alpha_1+r^*)/\alpha_2$, $q_y(0)$, $\alpha_2 s$ due to $((\alpha_1+r^*)/\alpha_2)(\alpha_2 s)=\alpha_1 s+r^* s$, $\mathcal{A}$ needs to cancel $r^* s$, which needs to use the terms r_j^*, $r^*+r_j^*$, $q_y(0)$, $r_j^* q_y(0)$. Because of $q_y(0)$ is the secret share of s according to the access tree $\mathcal{T}^*$. But the adversary $\mathcal{A}$ cannot rebuilt $r^* s$ for that the terms outlined above can only be rebuilt only if the attribute set S can meet $\mathcal{T}^*$.

Therefore, we prove that $\mathcal{A}$ gains a negligible advantage in the IND-CKA game. It is to say that our scheme is secure against IND-CKA. This completes the proof. □

7 Performance Evaluation

In this section, we present the efficiency analysis for our scheme in terms of both complexity and actual execution time, and further compare our scheme with the state-of-the-art PAB-MKS [27]. TABLE 2 defines the notation used in comparison.

In the theoretical analysis, we mainly show the computational and storage cost complexity in TABLE 3 and TABLE 4, respectively. We mainly consider the time-consuming operations, namely, bilinear pairing operation P, exponentiation operation $E_{\mathbb{G}}$ in group $\mathbb{G}$, exponentiation operation $E_{\mathbb{G}_T}$ in group $\mathbb{G}_T$, hash operation H_1 and H_2. We do not consider the multiplication operations because they are much more lightweight than the above operations.

7.1 Theoretical Comparison

In TABLE 3, we present the comparison of computation cost under the same access control policy tree $\mathcal{T}$. We observe that the complexity of KeyGen in our scheme is the same as that of the PAB-MKS [27].

In the PAB-MKS scheme, the encryption cost is more expensive than that of our scheme. Specifically, the former scheme computation cost is $(m+2|Y|+3)\cdot E_{\mathbb{G}}+|Y|\cdot H_1+m\cdot H_2$, while in our scheme the computation cost is $(t+1)\cdot E_{\mathbb{G}_T}+(2|Y|+1)\cdot E_{\mathbb{G}}+|Y|\cdot H_1+t\cdot H_2$. We note that $t\le 2\cdot\log(n/2)$, where n is the number of keyword in the keyword dictionary W, m is the number of keyword in W' embedded in one file, then $t\ll m$, e.g., when $n=1000$, $t\le 2\cdot\log 1000/2\le 18$, and the minimum keyword number in a file is about $m=100$.

We state that the computation cost of Trapdoor and Retrieve in our scheme is also less than that of PAB-MKS.

The computation cost of Trapdoor generation in PAB-MKS and our scheme are $(2s+3)\cdot E_{\mathbb{G}}+m^*\cdot H_2$ and $(2s+1)\cdot E_{\mathbb{G}}+(m^*\cdot\log n)\cdot H_2$, respectively.

The computation cost of retrieve in PAB-MKS and our scheme are $(2s+4)\cdot P+s\cdot E_{\mathbb{G}_T}$ and $(2s+1)\cdot P+(s+2m^*-1)\cdot E_{\mathbb{G}_T}$, respectively.

In TABLE 4, we present the comparison of storage cost. With the same reason shown as in TABLE 3, the storage cost of keyword index ciphertext (the output of Encrypt algorithm) in our scheme is less than that in PAB-MKS scheme, while the storage cost of KeyGen in our scheme is same as that of PAB-MKS. Our scheme has higher storage in Trapdoor, but it is acceptable and it is only one-time operation.

7.2 Experimental Performance

To present the practicability of our scheme in practice, we implement the scheme and PAB-MKS [27] with real-world dataset using Python language, and further run the experiment tests for 100 times, in which the dataset includes 1000 distinct keywords extracted from 500 PDF files (academic papers) provided by the Google Scholar. The maximum number of the keywords in a file is 200 while the minimum is 100. Our experimental platform is on Ubuntu 16.04 LTS with Intel Core i3 Processor 4170 CPU @3.70GHZ with 10.0 GB of RAM. Since these two schemes are highly dependent on the basic cryptographic operations in the pairing computation, we implement PBA-MKS and our scheme in software based on the libfenc library [33], using a 224-bit ($|\mathbb{G}|$=$|\mathbb{G}_T|$=224 bits, $\mathbb{Z}_p$=224 bits) MNT elliptic curve from the Stanford Pairing-Based Crypto library.

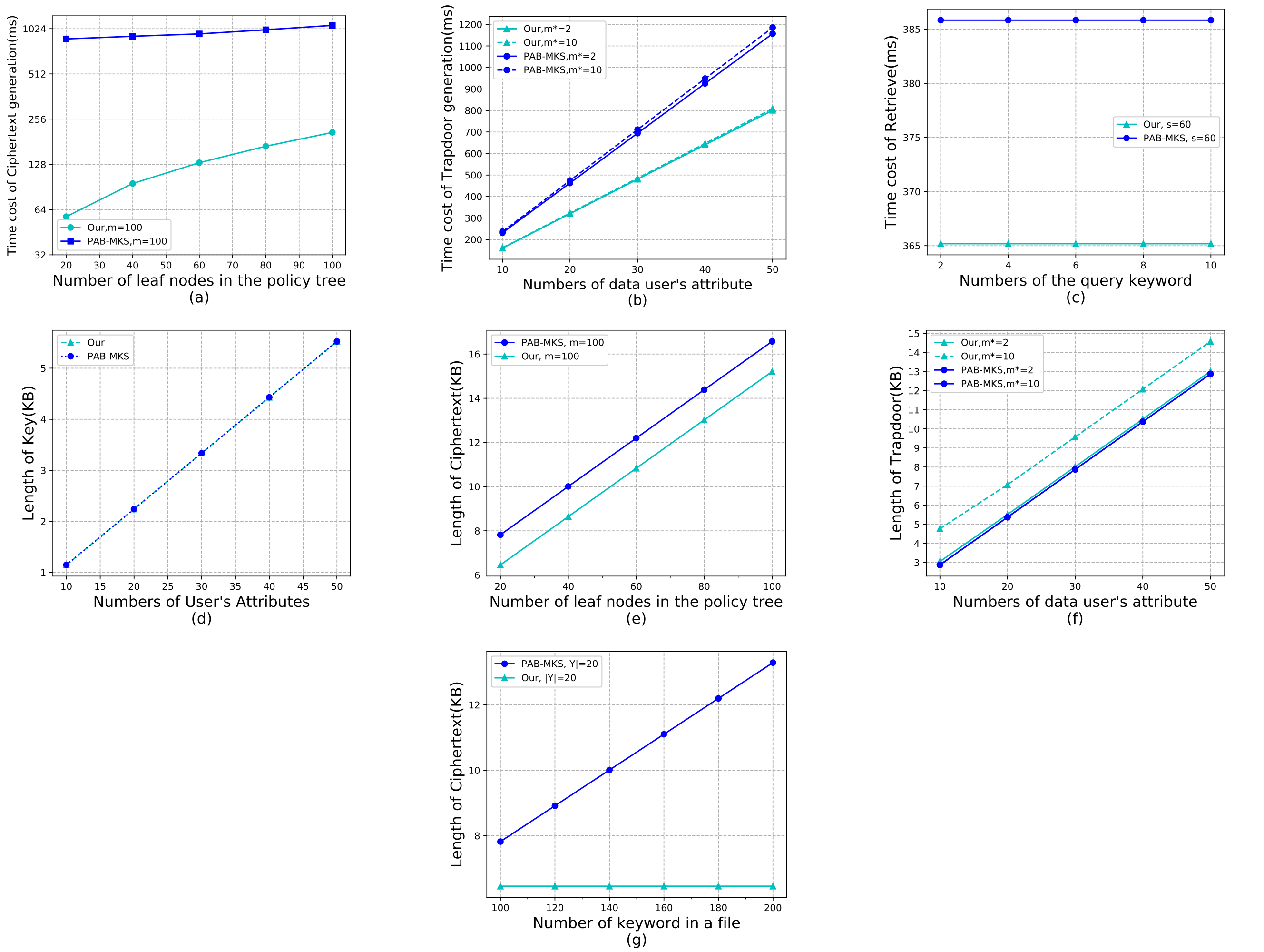


Fig. 7: The performance of the PABE-MKS and our scheme

To bring convenience in comparison, in PAB-MKS and our scheme, we encrypt the keyword sets of each file by using the same access control policy $\mathcal{T}$. For example, the policy tree $\mathcal{T}$ is "AND" access tree: (A_1 AND A_2 AND,...,AND $A_{|Y|}$), where A_i is an attribute. We set the number of DU's attribute $s \in [10, 20, 30, 40, 50]$, the number of leaf node in the access policy tree $|Y| \in [20, 40, 60, 80, 100]$, the number of keyword in each files m=100, and the number of keyword in the query $m^* \in [2, 4, 6, 8, 10]$.

Fig. 7a presents the computation time cost of ciphertext generation which is executed by DO. As described in TABLE 3, the ciphertext generation time cost of the scheme PAB-MKS is affected by two factors, the number of keyword embedded in a file (m) and the number of leaf nodes ($|Y|$) in policy tree $\mathcal{T}$, while the ciphertext generation time is affected by t and $|Y|$ in our scheme. We further set $m = 100$ and $n = 1000$ in the Encrypt phase. Because $t \leq 2 \cdot \log (n/2) \leq 18$, and then $t \ll m$, thus, we have that the cost time of ciphertext generation in our scheme is efficient than PAB-MKS. When $|Y| = 60$, the computation cost time is 131ms for us, while the PAB-MKS scheme needs 946ms to generate a ciphertext.

Fig. 7b presents the computation time cost of generating a trapdoor which is executed by DU. As described in TABLE 3, the trapdoor generation time cost of the scheme PAB-MKS and our scheme is affected by two factors: m^* and s. The PAB-MKS scheme and our scheme need $(2s{+}3) \cdot E_{\mathbb{G}} {+} m^* \cdot H_2$, $(2s{+}1) \cdot E_{\mathbb{G}} {+} (m^* \cdot \log n) \cdot H_2$ in trapdoor generation phase, respectively. Because the hash operation H_1 is much more efficient than the exponentiation operations [19], then the hash operation almost can be ignored. Thus, the trapdoor generation cost in our scheme is efficient than that of PAB-MKS. As Fig. 7(b), we can see that when $m^* = 2$ and $m^* = 10$, the our cost of generating trapdoor is efficient than that in PAB-MKS scheme, respectively. For example, when $s = 50$, $m^* = 10$, our scheme (requiring 809 ms) outperforms the PAB-MKS scheme (1186 ms) by around 300 ms.

Fig. 7c presents the computation time cost of retrieving the ciphertext. As described in TABLE 3, the time cost of retrieve algorithm in PAB-MKS and our scheme are $(2s + 4) \cdot P + s \cdot E_{\mathbb{G}_T}$, $(2s + 1) \cdot P + (s + 1) \cdot E_{\mathbb{G}_T}$, respectively. Since one time exponentiation computation is efficient than one time pairing operation under the same security condition, we can state that our scheme is efficient than PAB-MKS in retrieving process. For example, when $s = 50$, $m^* = 6$, our scheme only takes 74.636748 ms.

As described in TABLE 4, the storage cost of KeyGen algorithm in PAB-MKS and our scheme are $(2s+1)\cdot|\mathbb{G}|$, $(2s+1)\cdot|\mathbb{G}|$, respectively. As shown in Fig. 7d, the storage cost of KeyGen algorithm in our scheme is the same as that in PAB-MKS.

Fig. 7e and Fig. 7g present the ciphertext size of PAB-MKS and our scheme. As the analysis in TABLE 4, the number of keyword index ciphertext of PAB-MKS is affected by m and $|Y|$, while ours is affected by t and $|Y|$. Due to $t \leq 2\cdot \log{(n/2)}$, as shown in Fig. 7e, when setting $m = 100$, as the number $|Y|$ decreases, our scheme outperforms the PAB-MKS in terms of storage cost in Encrypt algorithm. For instance, when setting $n = 1000$, $m = 100$, $|Y| = 60$, the ciphertext length of PAB-MKS is 12.19 KB, while ours is 8.42 KB. As shown in Fig. 7g, when setting $|Y| = 20$, with the increase of keyword number m, the number of keyword index ciphertext in our scheme is constant, while that of PAB-MKS is linearly growing with m.

Fig. 7f presents the trapdoor size of the two schemes. As the analysis in TABLE 4, compared with PAB-MKS, the trapdoor size in our scheme is slightly more than PAB-MKS due to the extra item $(m^*\cdot \log{n})|\mathbb{Z}_p|$. For example, when setting $s = 50$, $m^* = 10$, the trapdoor size of PAB-MKS and ours are 12.875 KB and 14.578 KB, respectively. Then our scheme is still acceptable in practice since the Trapdoor algorithm is a one-time cost.

8 Conclusion

In this paper, we've proposed an efficient ABMKS with short-size ciphertext of keyword index which provides secure multi-keyword search service with fine-grained access control. Its number of search index is very small, 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 scheme is efficient in terms of both the computation and communication overhead in practice.

Acknowledgments

This work is supported by NSFC (Grant Nos. 61672110, 61671082, 61976024, 61972048)

References

[1] Kamara, S., & Lauter, K. (2010, January). Cryptographic cloud storage. In International Conference on Financial Cryptography and Data Security (pp. 136-149). Springer, Berlin, Heidelberg.

[2] Yu, Y., Li, Y., Au, M. H., Susilo, W., Choo, K. K. R., & Zhang, X. (2016, July). Public cloud data auditing with practical key update and zero knowledge privacy. In Australasian Conference on Information Security and Privacy (pp. 389-405). Springer, Cham.

[3] JoSEP, A. D., KAtz, R., KonWinSKi, A., Gunho, L. E. E., PAttERSon, D., & RABKin, A. (2010). A view of cloud computing. Communications of the ACM, 53(4).

[4] Quick, D., & Choo, K. K. R. (2013). Dropbox analysis: Data remnants on user machines. Digital Investigation, 10(1), 3-18.

[5] Quick, D., & Choo, K. K. R. (2016). Big forensic data reduction: digital forensic images and electronic evidence. Cluster Computing, 19(2), 723-740.

[6] Liu, Z., Li, T., Li, P., Jia, C., & Li, J. (2018). Verifiable searchable encryption with aggregate keys for data sharing system. Future Generation Computer Systems, 78, 778-788.

[7] Song, D. X., Wagner, D., & Perrig, A. (2000, May). Practical techniques for searches on encrypted data. In Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000 (pp. 44-55). IEEE.

[8] Bellare, M., Boldyreva, A., & O'Neill, A. (2007, August). Deterministic and efficiently searchable encryption. In Annual International Cryptology Conference (pp. 535-552). Springer, Berlin, Heidelberg

[9] Bao, F., Deng, R. H., Ding, X., & Yang, Y. (2008, April). Private query on encrypted data in multi-user settings. In International Conference on Information Security Practice and Experience (pp. 71-85). Springer, Berlin, Heidelberg.

[10] Chai, Q., & Gong, G. (2012, June). Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers. In 2012 IEEE International Conference on Communications (ICC) (pp. 917-922). IEEE.

[11] Curtmola, R., Garay, J., Kamara, S., & Ostrovsky, R. (2011). Searchable symmetric encryption: improved definitions and efficient constructions. Journal of Computer Security, 19(5), 895-934.

[12] Kamara, S., Papamanthou, C., & Roeder, T. (2012, October). Dynamic searchable symmetric encryption. In Proceedings of the 2012 ACM conference on Computer and communications security (pp. 965-976). ACM.

[13] Boneh, D., Di Crescenzo, G., Ostrovsky, R., & Persiano, G. (2004, May). Public key encryption with keyword search. In International conference on the theory and applications of cryptographic techniques (pp. 506-522). Springer, Berlin, Heidelberg.

[14] Hwang, Y. H., & Lee, P. J. (2007, July). Public key encryption with conjunctive keyword search and its extension to a multi-user system. In International conference on pairing-based cryptography (pp. 2-22). Springer, Berlin, Heidelberg.

[15] Li, J., Wang, Q., Wang, C., Cao, N., Ren, K., & Lou, W. (2010, March). Fuzzy keyword search over encrypted data in cloud computing. In 2010 Proceedings IEEE INFOCOM (pp. 1-5). IEEE.

[16] Bellare, M., Boldyreva, A., & O'Neill, A. (2007, August). Deterministic and efficiently searchable encryption. In Annual International Cryptology Conference (pp. 535-552). Springer, Berlin, Heidelberg.

[17] Baek, J., Safavi-Naini, R., & Susilo, W. (2008, June). Public key encryption with keyword search revisited. In International conference on Computational Science and Its Applications (pp. 1249-1259). Springer, Berlin, Heidelberg.

[18] Cui, B., Liu, Z., & Wang, L. (2015). Key-aggregate searchable encryption (KASE) for group data sharing via cloud storage. IEEE Transactions on computers, 65(8), 2374-2385.

[19] Zheng, Q., Xu, S., & Ateniese, G. (2014, April). VABKS: verifiable attribute-based keyword search over outsourced encrypted data. In IEEE INFOCOM 2014-IEEE Conference on Computer Communications (pp. 522-530). IEEE.

[20] Zhang, R., Xue, R., Yu, T., & Liu, L. (2016, June). PVSAE: A public verifiable searchable encryption service framework for outsourced encrypted data. In 2016 IEEE International Conference on Web Services (ICWS) (pp. 428-435). IEEE.

[21] Sun, W., Yu, S., Lou, W., Hou, Y. T., & Li, H. (2014). Protecting your right: Verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. IEEE Transactions on Parallel and Distributed Systems, 27(4), 1187-1198.

[22] Huang, Q., & Li, H. (2017). An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks. Information Sciences, 403, 1-14.

[23] Chen, R., Mu, Y., Yang, G., Guo, F., & Wang, X. (2015). Dual-server public-key encryption with keyword search for secure cloud storage. IEEE transactions on information forensics and security, 11(4), 789-798.

[24] Miao, Y., Ma, J., & Liu, Z. (2016). Revocable and anonymous searchable encryption in multi-user setting. Concurrency and Computation: Practice and Experience, 28(4), 1204-1218.

[25] Miao, Y., Ma, J., Liu, X., Jiang, Q., Zhang, J., Shen, L., & Liu, Z. (2017). VCKSM: Verifiable conjunctive keyword search over mobile e-health cloud in shared multi-owner settings. Pervasive and Mobile Computing, 40, 205-219.

[26] Miao, Y., Ma, J., Liu, X., Li, X., Jiang, Q., & Zhang, J. (2017). Attribute-based keyword search over hierarchical data in cloud computing. IEEE Transactions on Services Computing.

[27] Miao, Y., Ma, J., Liu, X., Li, X., Liu, Z., & Li, H. (2017). Practical attribute-based multi-keyword search scheme in mobile crowdsourcing. IEEE Internet of Things Journal, 5(4), 3008-3018.

[28] Golle, P., Staddon, J., & Waters, B. (2004, June). Secure conjunctive keyword search over encrypted data. In International Conference on Applied Cryptography and Network Security (pp. 31-45). Springer, Berlin, Heidelberg.

[29] Park, D. J., Kim, K., & Lee, P. J. (2004, August). Public key encryption with conjunctive field keyword search. In International Workshop on Information Security Applications (pp. 73-86). Springer, Berlin, Heidelberg.

[30] Goyal, V., Pandey, O., Sahai, A., & Waters, B. (2006, October). Attribute-based encryption for fine-grained access control of encrypted data. In Proceedings of the 13th ACM conference on Computer and communications security (pp. 89-98). Acm.

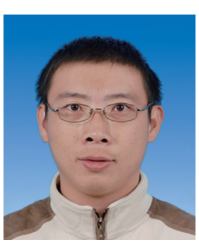
[31] Bethencourt, J., Sahai, A., & Waters, B. (2007, May). Ciphertext-policy attribute-based encryption. In 2007 IEEE symposium on security and privacy (SP'07) (pp. 321-334). IEEE.

[32] Naor, D., Naor, M., & Lotspiech, J. (2001, August). Revocation and tracing schemes for stateless receivers. In Annual International Cryptology Conference (pp. 41-62). Springer, Berlin, Heidelberg.

[33] Green, M., Akinyele, A., & Rushanan, M. (2004). libfenc: The functional encryption library. Avaiable from http://code. google. com/p/libfenc. Baodong Qin received the B. Sc. degree.

Zhidan Li received his bachelor's degree at ZhengZhou university in 2013. Now he is doing research in the Institute of Network and Technology at BUPT and his interests are focused on the network security and cryptography protocols. E-mail: zhidanli@bupt.edu.cn

Wenmin Li received the B.S. and M.S. degrees in Mathematics and Applied Mathematics from Shaanxi Normal University, Xi'an, Shaanxi, China, in 2004 and 2007, respectively, and the Ph.D. degree in Cryptology from Beijing University of Posts and Telecommunications, Beijing, China, in 2012. Her research interests include cryptography and information security. E-mail: liwenmin02@outlook.com



Fei Gao received the B.S. degrees and the Ph.D. degree in Cryptology from Beijing University of Posts and Telecommunications, Beijing, China, in 2002 and in 2007, respectively. Now he is a Professor, doctoral supervisor of Beijing University of Posts and Telecommunications. Her research interests include Quantum cryptography protocol and its security analysis, Quantum Private Query, Quantum key distribution. E-mail: gaof@bupt.edu.cn

Wei Yin 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 Beijing University of Posts and Telecommunications, Beijing, China, in 2019. His research interests include public key cryptography, lattice cryptography, and provable security.

Hua Zhang received the BS degree in telecommunications engineering from the Xidian University in 1998, the MS degree in cryptology from Xidian University in 2005, and the Ph.D degree in cryptology from Beijing University of Posts and Telecommunications in 2008. Now she is a lecturer of Beijing University of Posts and Telecommunications. Her research interests include cryptography, information security and network security. E-mail: zhanghua_288@bupt.edu.cn

Qiaoyan Wen received the B.S. and M.S. degrees in Mathematics from Shaanxi normal University, Xi'an, China, in 1981 and 1984, respectively, and the Ph.D degree in cryptography from Xidian University, Xi'an, China, in 1997. She is a professor of Beijing University of Posts and Telecommunications. Her present research interests include coding theory, cryptography, information security, internet security and applied mathematics. E-mail: wqy@bupt.edu.cn

Kaitai Liang 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 are 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. E-mail: k.liang@surrey.ac.uk