# Privacy Protection Data Retrieval Scheme With Inverted Index for IoT Based on Blockchain

Wenmin Li, Yang Chen, Fei Gao, Shuo Zhang, Hua Zhang, and Qiaoyan Wen

***Abstract*—In the 6G era, Internet of Things (IoT) devices can form a blockchain network, which also faces the problems of data sharing. The data transmitted and stored through the network have the risk of privacy leaking. Encrypting the shared data can satisfy the need of the privacy, and retrieving the encrypted data can make the data used efficiently. However, to enable users to retrieve encrypted data and perform fine-grained authorization on their encrypted files is still a great challenge. Although attribute-based keyword search (ABKS) is a well-received solution to the challenge, there are still privacy and efficiency issues if the traditional ABKS schemes are directly used in blockchain data sharing. In order to solve the problems, this article proposes privacy protection data retrieval scheme with an inverted index, which is an application of attribute-based encryption. First, our scheme is proved secure against the outside keyword guessing attack (KGA) and chosen keyword attack (CKA) under the semitrusted model. Second, the scheme returns a multikeywords ranked result. Third, we analyze the efficiency of our scheme and verify it by simulation. The results show that our scheme has improvement in efficiency and can meet the data sharing needs of the blockchain network composed of IoT devices.**

***Index Terms*—Blockchain, inverted index, Internet of Things (IoT), multikeyword ranked search, privacy protection.**

## I. Introduction

IN THE 6G era, the great progress of communication technology will continue to bring earth shaking changes to mankind. In the real world, more information about people and objects will be extracted, processed, and preserved in digital format. We need strong computing and communication capabilities as support. 6G has stronger carrying capacity, which is more conducive to the communication of short-range equipment. This will meet the needs of more devices accessing the network at the same time, and these devices do not need long-distance access to the network. The development of network technology has also brought many new technologies to meet more functions, which promote the emergence of a large number of novel and interesting devices. Therefore, more and more designs are used to meet people's needs, and more and more Internet of Things (IoT) devices came into being.

These devices with certain computing and communication capabilities naturally form a distributed network environment. This distributed network is composed of devices with limited computing power, but it has a very large number of devices, and the overall computing power cannot be underestimated. However, the problem of managing so many devices also makes the traditional network method difficult. Many new network models have been widely discussed [1], among those, the blockchain network stands out among many theories [2]–[4]. It ensures that in a decentralized network, the data will not be tampered at will, and any behavior will be recorded and documented, so as to realize a network mode of mutual trust [5]. The development and popularization of blockchain will change people's traditional interaction mode and create a new application mode. Many imaginative and practical scenarios will be put forward.

With the popularity of IoT devices for blockchain network management, it is easy to trust IoT devices. With human trust in IoT devices, a large amount of data generated by users' IoT devices will also be obtained, processed, and displayed by the corresponding terminals. In this process, different devices will generate a large amount of data, and better processing and use of these data will also be an important aspect of IoT application promotion. New technology application, such as inductive coloring [6] and dynamic carrier technology [7], brings new user experience. The IoT device needs to deal with massive of shared private data, which may contain a large number of sensitive information of individuals or organizations. Meanwhile, the data provider (DP) may not fully trust the server, which is a prominent problem in data sharing of the blockchain. In order to protect privacy, data will be outsourced to blockchain after encrypted. However, the mechanism of outsourcing after encrypting affects data retrieval. Moreover, given the actual needs of DP, there is an urgent need for refined permission to access encrypted data. Accordingly, exploring a way for a data user (DU) to quickly retrieve over encrypted data without leaking privacy, and for a DP to perform refined permission is of prime importance in the scenarios of data sharing in the blockchain. Data sharing will lead to serious privacy and security problems. Even the best online services are facing security vulnerabilities and the possibility of data theft. The sharing platform needs an effective method to meet the user's safe access to data and better manage data.

Manuscript received 5 July 2021; revised 24 September 2021; accepted 8 November 2021. Date of publication 16 November 2021; date of current version 7 July 2023. This work was supported in part by the National Key Research and Development Program of China under Grant 2020YFB1005900, and in part by the National Natural Science Foundation of China under Grant 62072051, Grant 61672110, Grant 61671082, Grant 61976024, and Grant 61972048. *(Corresponding authors: Fei Gao; Yang Chen.)*

Wenmin Li, Fei Gao, Shuo Zhang, Hua Zhang, and Qiaoyan Wen are with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China (e-mail: gaof@bupt.edu.cn).

Yang Chen is with the Department of Emergency Responding, National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100029, China (e-mail: cylovehehe@163.com).

Digital Object Identifier 10.1109/JIOT.2021.3128528

A technique called searchable encryption (SE) [8], [9] is used by many researchers to solve the data retrieval problem on the blockchain network composed of IoT devices. SE enables users to retrieve encrypted data for keywords as they want. There are many study results in last decade. In order to support fine-grained search authorization in SE schemes, attribute-based keyword search (ABKS) gives different schemes and sustains attention for the application in cloud. In an ABKS scheme, a DP can embed his/her access policy into the encrypted index for each file. The server returns the right encrypted one to a DU whose attributes of secret keys and keywords of trapdoor fit the access policy and index, respectively. By this means, ABKS can finish keyword retrieval and authorization simultaneously with fine grain. However, direct adoption of the traditional ABKS suffers two problems, so that it do not suit for blockchain made up of devices connected to the IoT.

The first problem is that most of the ABKS schemes are less efficient when retrieving massive data. The reason is that the forward index is deployed in the ABKS schemes which means each file corresponds to an encrypted index. To finish the search, the server has to search all these indices; therefore, the search complexity is linearly about the number of files stored by a server. When the amount of files is too large, the computational cost of retrieving will be very high. The IoT devices cannot afford the computing cost. The second problem is that most of the ABKS schemes cannot resist outside keyword guessing attack (KGA). The root of the problem is that any entity in these schemes can generate index and search keywords of the trapdoor in the index by itself. By intercepting a mobile user's trapdoor via an public channel, the entity can construct the index with its guessed keyword. The entity can also verify whether the guessed keyword is in objective trapdoor. It just need checking if the target trapdoor matches the generated index. The correct keyword will be leaked if the entity repeats checking. Consequently, outside KGA breaks the privacy of IoT devices and blockchain.

Low efficiency and possible attacks hinder the popularity of retrieval under ciphertext. Then, the functionality of building a blockchain network through IoT devices will also be limited accordingly. In order to realize privacy protect data retrieval, this article proposes an inverted index-based ABKS (IABKS) scheme to improve the privacy and practicability of ABKS to make it better applied to IoT devices.

### A. Related Work

*1) ABKS:* Goyal *et al.* [10] put forward the first attribute-based encryption (ABE) scheme. ABE can realize access control with a fine-grained result; hence, it is considered to be a promising cryptography technology. It can be divided into: key policy ABE (KP-ABE) [10] and ciphertext policy ABE (CP-ABE) [11], [12]. In KP-ABE, the ciphertext is related to attributes, in CP-ABE, the ciphertext is related to the access policy designed by the data owner, and CP-ABE has gradually become a preferred choice when designing access control mechanisms in the cloud because of its scalability and flexibility, and it has been enriched with kinds of functions [13]–[15].

In order to provide fine-grained search authorization in SE schemes, Zheng *et al.* [16] and Sun *et al.* [17] combined CP-ABE with ABKS, and proposed the concept of ABKS successively. The access policy in [16] and [17] is, respectively, access tree and AND-gates. In an ABKS scheme, a data owner constructs the encrypted index based on keywords of the file and the designed the access policy that is used to define what kind of users can access the file. The user can independently generate the trapdoor with his/her attributes and interested keywords. Only when the attributes contained in the trapdoor satisfy the access policy can the server returns the matching results to the user.

*2) ABKS Against Outside KGA:* Byun *et al.* [18] first proposed KGA. The guessing-then-verifying attack pattern is mentioned above. KGA is subdivided into inside KGA and outside one depending on whether the adversary can be a server or not. Although ABKS has been studied for several years, only a few ABKS schemes have been proved to resist outside KGA so far.

Research work [16] and [19] pointed out that the inside KGA makes it impossible to fully protect the keyword information encrypted in the trapdoor in the public-key system. In view of this, a weaker notion called keyword secrecy is introduced in [16], which ensures that the probability of the adversary successfully guessing a keyword from the trapdoor and index is not bigger than the random result. Although the schemes in [16] and [20] can achieve keyword secrecy, they still suffer from outside KGA.

Lately, Chen *et al.* [21] utilized a dual-server model to propose an attribute-based search scheme with a multikeyword ranked result. There are two servers in the scheme, and the search operation is completed by the two server cooperatively and restrictively to resist the chosen keyword attack (CKA) and inside KGA. Although the security and search flexibility of the scheme have been significantly improved, the scheme also has shortcomings. The adopted dual-server model assumes that the servers are not collusive. The assumption is relatively strong to a certain extent, which will affect the application range of the scheme in IoT devices.

*3) Inverted Index versus Forward Index in ABKS:* Inverted index and forward index are two commonly used data structures in the information retrieval system [22], [23]. In an inverted index, each keyword maps the identifiers of the files that contain that keyword. Inversely, in a forward index, each identifier of the file maps the keywords contained in that file. Compared with forward index-based ABKS schemes, inverted index-based ABKS schemes have mainly two advantages. First, these schemes are much more efficient when facing massive data. In these schemes, search is directly pointed to the related files rather than search file by file. The complexity of searching is reduced to constant level about keywords number in the files. Second, the inverted index has become the first choice in large data set search schemes. Consequently, IABKS schemes can incrementally be used in the exited inverted index.

Zheng *et al.* [16] first proposed IABKS and [16] provides single keyword search and its search efficiency is relatively low. It is likely to lead to a large amount of irrelevant

TABLE I
FEATURE COMPARISON

| Scheme | Retrieval Mode | Access Policy | Resist Outside KGA | Forward/Inverted Index |
|---|---|---|---|---|
| [17] | Single Keyword | AND-gates on +/- | × | Forward index |
| [6-8] | Conjunctive Keyword | Access tree | × | Forward index |
| [21] | Multi Keyword Ranked | Access tree | ✓ | Forward index |
| [16] | Single Keyword | Access tree | × | Inverted index |
| IABKS | Multi Keyword Ranked | Access tree | ✓ | Inverted index |

files in the search results, especially when there is a large number of files, thus wasting computing resources and bandwidth resources. There is an improved scheme in [16] (named VABKS) to avoid such shortcoming, the DO has to separately generate $n$ encrypted indices to cover the $n$ keywords. But such an approach will also reduce the efficiency of the scheme, because $n-1$ extra parts related to the same access policy are computed. As far as we know, there is no follow-up study on inverted index-based ABKS.

Sun *et al.* [17] put forward the first forward index-based ABKS scheme, the scheme provides single keyword search. Then, a series of researches on forward index-based ABKS followed. Miao *et al.* [24]–[26] presented several outstanding ABKS schemes, the retrieval mode of them is conjunctive keyword search. These schemes provide a more flexible retrieval mode by publishing an ordered keyword set. In 2018, He *et al.* [27] proposed a boolean keyword search ABKS scheme based on exposing keyword names.

### B. Contribution

To solve the problem of data sharing in a blockchain network of IoT devices, this article proposes privacy protection data retrieval scheme with an inverted index which is an application of ABE. It is called as IABKS scheme. Comparison of relevant schemes and IABKS schemes is shown in the Table I. Analyzing their performance, we conclude that the IABKS is more suitable for blockchain data sharing, particularly which is made up by IoT devices.

The main contributions of this article are as follows.

1) *Privacy:* To improve the security of the ABKS deployed in blockchain, we provide the formal security model and the IABKS scheme is proved to be IND-CKA secure and IND-KGA secure. This means any adversary cannot guess the keywords of trapdoor and index hence the privacy of IoT devices and data is protected.
2) *Multikeyword Ranked Search:* To make the search more accurate and to improve searching experience of users, the IABKS scheme supports search with multikeyword ranked result.
3) *Efficiency and Practicality:* To improve the search efficiency when facing massive data in blockchain, the IABKS scheme deploys inverted index. An all-round simulations of the schemes is carried out using actual data set, and the simulation results show that the proposed scheme has a huge advantage in efficiency compared with similar schemes, especially in a IoT environment.

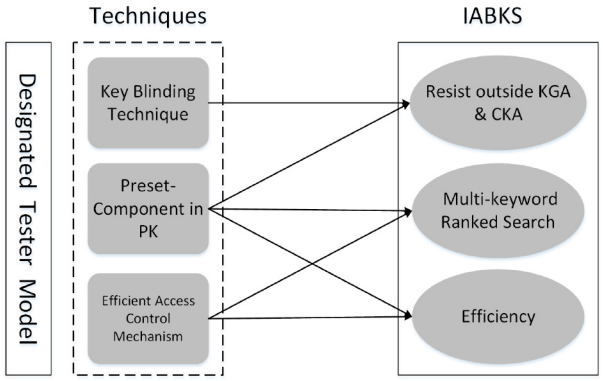


Fig. 1. Technical roadmap of IABKS.

### C. Our Techniques

Fig. 1 shows the proposed scheme's technical roadmap. Building a designated tester model is the crucial point and then use techniques shown in the dotted frame to take full advantages of the model. The right part shows the effects that the techniques can achieve.

*To resist outside KGA*, we put forward a designated tester model which is inspired from the idea of [28]. In our model, only the server can use its secret key to perform the search operation. To achieve this, we preset in public key a series of components $B_{1,i} = g_1^{b_i}$ and $B_{2,i} = g_2^{a\alpha b_i}$, they are used in the generation of trapdoor and encrypted index, respectively. The secret key of the server, $a$, is indispensable for the search operation, which prevents outside adversaries from verifying their guesses. In addition, the key blinding technique [12] is commonly used in the generation of trapdoor in ABKS, in the scheme, we utilize that technique and the preset-components $H_i$ in the public key *to resist CKA*.

*To provide search with multikeyword ranked result*, we design innovative structures of the trapdoor and encrypted index. Benefitting from the structures and the preset components $H_i$ in the public key, every keyword of the encrypted index and trapdoor is separate under the premise of ensuring the security of the scheme. In this way, the *inverted index* can be established and the trapdoor and the index with the same keywords can be matched with each other freely. The server can calculate the relevance scores according to the interest by users' query set, and then sorts the search results.

*To manage IoT devices on blockchain networks better,* we use attributes to do it. Different from the distributed network

composed of traditional PC, there are many kinds of devices in the network. These strange devices have more attributes at the same time, some of which are natural or artificially marked. This classification of IoT devices enables users to accurately find the lamp and it is more convenient to manage according to the attributes.

The ABKS schemes mainly constructed based on [11] realize access control, and their efficiency can be further improved. *To improve efficiency*, we construct a more efficient access control mechanism. Specifically, the efficiency of generating secret keys and trapdoors in the IABKS scheme is almost 50% higher than these ABKS schemes. In addition, with the preset components in the public key, the structure of the keyword-related parts in the encrypted index and trapdoor is also concise. Therefore, our IABKS is more efficient, especially in a massive data environment.

## II. Preliminaries

### A. Asymmetric Bilinear Maps

Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be three multiplicative cyclic groups of prime order $p$, and $g_1$ and $g_2$ are the generator of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Let $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be an asymmetric bilinear map that satisfies three properties [29].

1) *Nondegeneracy:* $e(g_1, g_2) \neq 1$.
2) *Bilinearity:* For all $\eta_1 \in \mathbb{G}_1$, $\eta_2 \in \mathbb{G}_2$, $\delta_1, \delta_2 \in \mathbb{Z}_p^*$, we have $e(\eta_1^{\delta_1}, \eta_2^{\delta_2}) = e(\eta_1, \eta_2)^{\delta_1 \delta_2}$.
3) *Computability:* For all $\eta_1 \in \mathbb{G}_1$, $\eta_2 \in \mathbb{G}_2$, $e(\eta_1, \eta_2)$ and the group operations in $\mathbb{G}_1$ and $\mathbb{G}_2$ can be efficiently computed.

### B. Generic Bilinear Group Model

We prove IABKS's security in the generic bilinear group model [30]. $\xi_1$, $\xi_2$, and $\xi_T$ are additive group $\mathbb{F}_p$'s three random encodings, they are injective maps: $\xi_1, \xi_2, \xi_T : \mathbb{F}_p \to \{0, 1\}^m$, $m > 3\log(p)$. Let $\mathbb{G}_i = \{\xi_i(x) : x \in \mathbb{F}_p\}$, $i = 1, 2, T$. The adversary can use the oracles that simulate the hash function, group operations in $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ and the asymmetric bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ or interact with the challenger in the security game. Here, $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ denote generic bilinear groups.

### C. Access Tree

Let $\mathcal{T}$ denote an access tree, it is a tree structure and each nonleaf node $y$ of $T$ corresponds to a threshold gate characterized by its children nodes and threshold value. Let $num_y$ denote the number of children of node $y$ and $k_y$ denote its threshold value, where $0 < k_y \leq num_y$.

To facilitate the description of the access tree, we define some functions.

1) att($y$) represents the attribute associated with leaf node $y$ in $\mathcal{T}$.
2) *parent*($y$) represents the parent of node $y$ in $\mathcal{T}$.
3) *index*($y$) returns a unique number associated with node $y$.
4) $\mathcal{T}_y$ stands for the subtree of $\mathcal{T}$ with node $y$ as the root node. Let $f(S, \mathcal{T}_y) = 1$ show that the attribute set $S$ satisfies access tree $\mathcal{T}_y$. The recursive computation of the access tree is: if $y$ is a nonleaf node, $f(S, \mathcal{T}_y)$ returns 1 when at least $k_y$ children return 1. If $y$ is a leaf node, $f(S, \mathcal{T}_y)$ returns 1 when att($y$) $\in S$.

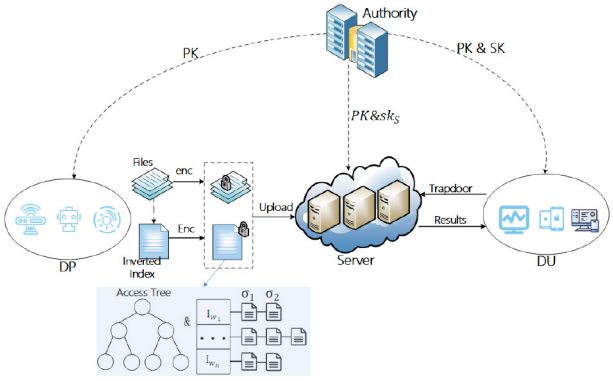


Fig. 2. System architecture of IABKS.

## III. System Architecture and Security Definition

### A. Entities in System Architecture

In our scheme, IoT devices will collect information, images, and recordings in the user's environment, work information of IoT devices themselves, or other information worthy of collection and storage, and these information will be classified and sorted by the uploader, and attribute tags will be made according to the uploader devices and the details of the uploaded information. The information is saved through the server or blockchain, and the server will also undertake most of the calculation. Terminal equipment or data processing equipment using the method of ABKS can quickly obtain the correct information, and show it to the human.

As shown in Fig. 2, there are four generic entities: 1) authority; 2) DP; 3) DU; and 4) server involved in the architecture of the IABKS scheme. A detailed description of them is as follows.

1) *Authority:* Authority is trusted in the system and responsible for generating public key of the system, DUs' secret keys, and server's secret key.
2) *Data Provider:* The DP, which maybe a smart device, an IoT device, an environmental information collector, or another sensor, has data to be stored and shared. A DP needs to generate the encrypted index and uploads the encrypted index and encrypted files to the server.
3) *Data User:* DU is usually a terminal or a data processor, which presents or analyzes the data for people. In order to search for data from different data owners, a DU needs to submit the server a trapdoor generated by its secret key and the keywords of interest.
4) *Server:* The server stores the encrypted indices and encrypted files uploaded by the DPs. It could be a cloud server or blockchain. After receiving the trapdoor (search request) sent by the DU, the server searches files and sends the ranked search results to DU.

### B. Symbol and Definition

The following symbols are used in the rest of the article.

1) $M = (m_1, m_2, \ldots, m_l)$ is a set of $l$ files.
2) $F = (F_1, F_2, \ldots, F_l)$ is a series of $l$ files encrypted by the same access policy, where $F_i$ is generated by symmetric encryption of $m_i$.
3) $\Sigma = (\sigma_1, \sigma_2, \ldots, \sigma_l)$ is a set of identifiers, where $\sigma_i$ is the ID of the $i$th encrypted file $F_i$.
4) $Score = (s_1, s_2, \ldots, s_l)$ is a set of scores for each files, where $s_i$ is the score of $F_i$.
5) $W = \{w_1, w_2, \ldots, w_n\}$ is the keyword set from $M$.
6) $II = \{I_{w_1}, I_{w_2}, \ldots, I_{w_n}\}$ is the inverted index for $M$. Each $I_{w_i}$ is a inverted list contains the set of identifiers of the files associated with keyword $w_i$.

### C. Algorithms in IABKS Scheme

The IABKS scheme includes five algorithms: 1) **Setup**; 2) **KeyGen**; 3) **IndexGen**; 4) **TrapGen**; and 5) **Search**.

1) **Setup**$(1^\lambda)$: Given the security parameter $\lambda$, the authority runs the algorithm and outputs the public key $PK$, master key $MK$, and the secret key of server $sk_S$.
2) **KeyGen**$(MK, S)$: Given $MK$ and an attribute set $S$, the authority runs the algorithm and outputs the secret key of the DU $SK$.
3) **IndexGen**$(PK, \mathcal{T}, W)$: Given $PK$, an access policy $\mathcal{T}$, and a keyword set $W$, the DP runs the algorithm and outputs an encrypted index $I$.
4) **TrapGen**$(PK, W', SK)$: Given $PK$, a keyword set $W'$, and $SK$, the DU runs the algorithm and outputs a trapdoor $TK$.
5) **Search**$(I, TK, QI, II, sk_S)$: Given $I$, $TK$, DU's query interest $QI$, the inverted index $II$, and $sk_S$, the server runs the algorithm and outputs the set of identifiers of the selected encrypted files if the matching process holds, or a terminator $\perp$.

### D. Security Model

*1) Adversarial Model:* We provide the following assumptions in the model: the semitrusted server tries to learn the valuable information while performing the protocol truthfully. In addition, the DP and authority are reliable. Therefore, two types of adversaries are considered.

1) Type-1 adversary ($\mathcal{A}_1$) means that the collusion of corrupted users and server, whose secret keys can be obtained by the adversary. Therefore, the secret keys can be obtained by the adversary.
2) Type-2 adversary ($\mathcal{A}_2$) means that the outside adversary who can obtain corrupted users secret keys.

*2) Security Model:* The security of the IABKS scheme requires that the scheme can resist the following two attacks.

*Chosen-Keyword Attack:* This requires that $\mathcal{A}_1$ or $\mathcal{A}_2$ cannot judge the challenge index coming from which challenge keyword sets in the challenge index without any matched trapdoor.

*Outside Keyword Guessing Attack:* This requires that $\mathcal{A}_2$ cannot judge the challenge trapdoor matched with which of the two challenge keyword sets. In the public-key setting, it is *infeasible* to guarantee that the server cannot get the information encode by search tokens due to inside KGA.

*CKA Security Game for $\mathcal{A}_1$:*

**Setup:** The challenger $\mathcal{C}$ executes **Setup** algorithm and sends $PK$ and $sk_S$ to $\mathcal{A}_1$.

*Phase 1:* $\mathcal{C}$ maintains an empty table $E$, an empty set $D$, an empty keyword list $L_{kw}$ and sets $j = 0$. The following oracles can be queried by $\mathcal{A}_1$ adaptively.

1) $\mathcal{O}_{\textbf{KeyGen}}(S)$: $\mathcal{C}$ executes **KeyGen**$(MK, S)$, let $D \coloneqq D \cup \{S\}$ and sends $SK$ to $\mathcal{A}_1$.
2) $\mathcal{O}_{\textbf{Trap}}(S, W)$: $\mathcal{C}$ let $j \coloneqq j+1$, it executes **KeyGen**$(MK, S)$ to generate $SK$, then it performs **Trap**$(PK, W, SK)$ to generate $TK$. It stores the triple $(j, S, W)$ in table $E$ and sends $TK$ to $\mathcal{A}_1$.

*Challenge:* $\mathcal{A}_1$ sends two keywords sets $W_0$ and $W_1$, and an access policy $\mathcal{T}^*$ to $\mathcal{C}$. The requirements are as follows.

1) $\forall S \in D, f(S, \mathcal{T}^*) \neq 1$.
2) $W_0$ and $W_1$ have the same number of elements.[1]
3) For each triple in $E$, if $f(S, \mathcal{T}^*) = 1$, add $W$ to $L_{kw}$; then $W_0 \cap W = W_1 \cap W$, $W \in L_{kw}$.

$\mathcal{C}$ chooses $b \in_R \{0, 1\}$, runs **IndexGen**$(PK, \mathcal{T}^*, W_b)$ to construct $I^*$ and submits $I^*$ to $\mathcal{A}_1$.

*Phase 2:* $\mathcal{A}_1$ query the oracles continuously as in Phase 1. The condition in this stage is if $f(S, \mathcal{T}^*) = 1$:

1) $\mathcal{A}_1$ stop querying $\mathcal{O}_{\textbf{KeyGen}}$;
2) if $W \cup W_0 \neq W \cup W_1$, $(S, W)$ cannot be the input to $\mathcal{O}_{\textbf{Trap}}$.

*Guess:* $\mathcal{A}_1$ returns a guess $b'$ of $b$. The experiment outputs 1 if and only if $b' = b$.

*CKA Security Game for $\mathcal{A}_2$*: The Challenge, Phase 2, and *Guess* are the same as those in the last game for $\mathcal{A}_1$. Following are the **Setup** and Phase 1 in this game.

**Setup: Setup** can be executed by $\mathcal{C}$ and outputs $PK$ to $\mathcal{A}_2$.

*Phase 1:* This stage is the same as Phase 1 of the CKA security game for $\mathcal{A}_1$, apart from that, $\mathcal{A}_2$ can query the following oracle.

1) $\mathcal{O}_{\textbf{Search}}(I, TK, QI, II)$: $\mathcal{C}$ executes **Search** and the result is sent to $\mathcal{A}_2$.

The IABKS scheme is IND-CKA secure if the advantage of $\mathcal{A}_i$ in the CKA security game for $\mathcal{A}_i$, $i \in \{1, 2\}$

$$\left| \Pr\left[ \mathrm{Exp}^{\text{IND-CKA}}_{\mathcal{A}_i, \Pi_{IABKS}}(1^\lambda, U) = 1 \right] - \frac{1}{2} \right| \leq \mathrm{negl}(\lambda).$$

*IND-KGA Security Game for $\mathcal{A}_2$:*

**Setup:** $\mathcal{C}$ executes the **Setup** and sends $PK$ to $\mathcal{A}_2$.

*Phase 1:* $\mathcal{A}_2$ can query the following oracles for multiple times polynomially.

1) $\mathcal{O}_{\textbf{KeyGen}}(S)$: $\mathcal{C}$ performs **KeyGen** and sends $SK$ to $\mathcal{A}_2$.
2) $\mathcal{O}_{\textbf{Trap}}(S, W)$: $\mathcal{C}$ generates $SK$ by running **KeyGen**, and it uses inputs $W$ and $SK$ to run **Trap**. It returns $TK$ to $\mathcal{A}_2$.
3) $\mathcal{O}_{\textbf{Search}}(I, TK, QI, II)$: $\mathcal{C}$ runs **Search** and returns the outcome to $\mathcal{A}_2$.

*Challenge:* $\mathcal{A}_2$ sends $S^*$, keywords sets $W_0$ and $W_1$ to $\mathcal{C}$, it requires that $W_0$ and $W_1$ have the same elements. $\mathcal{C}$ chooses

[1] It also requires that each common element in the two sets corresponds to the same ids of files, which is beyond the scope of our discussion.

$b \in_R \{0,1\}$, runs **KeyGen**$(MK, S^*)$ to generate $SK$, then runs **Trap**$(PK, W_b, SK)$ to get $TK^*$, and submits $TK^*$ to $\mathcal{A}_2$.

*Phase 2:* $\mathcal{A}_2$ continues to query oracles just like in Phase 1 except that the index $I$ with $\mathcal{T}$ and $W$ cannot be used as the input of $\mathcal{O}_{\textbf{Search}}$ if $f(S^*, \mathcal{T}) = 1$ and $W \cap W_0 \neq W \cap W_1$.

*Guess:* $\mathcal{A}_1$ outputs a guess $b'$ of $b$. The experiment returns 1 if and only if $b' = b$.

The IABKS scheme is IND-KGA secure if the advantage of $\mathcal{A}_2$ in the IND-KGA security game for $\mathcal{A}_2$

$$\left|\Pr\left[\mathrm{Exp}^{\text{IND-KGA}}_{\mathcal{A}_2,\Pi_{IABKS}}(1^\lambda, U) = 1\right] - \frac{1}{2}\right| \leq \mathrm{negl}(\lambda).$$

## IV. Proposed IABKS Scheme

Let $N$ be the maximum of keywords of the files encrypted by the same access policy, and $H : \{0,1\}^* \to \mathbb{G}_1$,[2] $\overline{H} : \{0,1\}^* \to \mathbb{G}_1$ be secure hash functions. We define the Lagrange coefficient $\Delta_{i,S}$ for $i \in \mathbb{Z}_p$ and a set, $S$, of elements in $\mathbb{Z}_p$: $\Delta_{i,S}(x) = \prod_{j \in S, j \neq i}([x-j]/[i-j])$, other notations are defined in Section III-B.

**Setup**$(1^\lambda)$: It executes the group generator algorithm $\mathcal{G}(1^\lambda)$ and obtains bilinear groups $(g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$. It first selects $\alpha, \{\beta_i\}_{i=1}^N, a, h_0, \{b_i\}_{i=1}^M \in_R \mathbb{Z}_p$, then computes $\{B_{1,i} = g_1^{b_i}, B_{2,i} = g_2^{a\alpha b_i}\}_{i=1}^M$, $h = g_1^{h_0}$, and $\{H_i = g_1^{\beta_1}\}_{i=1}^N$. The public key is $PK = (g_1, g_2, g_2^\alpha, h, \{H_i\}_{i=1}^N, \{B_{1,i}, B_{2,i}\}_{i=1}^M)$, the secret key of server is $sk_S = (a)$, and the master key is $MK = (\{g_1^{\alpha\beta_i}\}_{i=1}^N, a)$.

**EncIndex**$(PK, \mathcal{T}, W)$: For encrypted files with the same access policy $\mathcal{T}$, let $W = \{w_1, w_2, \ldots, w_n\}$ be the keyword set, $n \leq N$. It chooses $s, d \in_R \mathbb{Z}_p$, and computes $W_0 = g_2^{-\alpha d}$, $\{W_i = H_i^s H(w_i)^d\}_{i=1}^n$, $\{D_i = B_{2,i}^d\}_{i=1}^M$.

It then selects a polynomial $q_x$ for each node in $\mathcal{T}$, these polynomials $q_x$ are chosen in a top to bottom manner starting with the root node $R$. For every node $x$ in the tree, the degree of $q_x$ is set as $d_x = k_x - 1$, where $k_x$ is the threshold value of the node.

Starting from the root node $R$, it sets $q_R = s$ and arbitrarily chooses $d_R$ other points of $q_R$ to define $q_R$ absolutely. For each nonroot node $x$, it let $q_x(0) = q_{parent(x)}(index(x))$ and chooses $d_x$ other points arbitrarily to completely define $q_x$. Let $Y$ be the leaf nodes in $\mathcal{T}$. It computes $C = g_2^s, \{C_y = h^{q_y(0)}\overline{H}(\mathrm{att}(y))^{-s}\}_{y \in Y}$. The index is set to: $I = (W_0, \{W_i\}_{i=1}^n, \{D_i\}_{i=1}^M, C, \{C_y\}_{y \in Y})$.

**KeyGen**$(MK, S)$: It chooses $r \in_R \mathbb{Z}_p$, then computes $\{K_i = g_1^{\alpha\beta_i} h^r\}_{i=1}^N$, $K' = g_2^r$, $\{K_x = \overline{H}(x)^r\}_{x \in S}$. The key is set to: $SK = (S, \{K_i\}_{i=1}^N, K', \{K_x\}_{x \in S})$.

**Trap**$(PK, SK, W')$: Let $W' = \{w'_1, w'_2, \ldots, w'_m\}$ be the keyword set for IoT user to query, $m \leq M$, it chooses $z, u \in_R \mathbb{Z}_p$ and computes $T = g_2^{\alpha u}, T' = g_1^z, \{T_j = H(w'_j)^u B_{1,j}^z\}_{j=1}^m$, $\{tk_j = K_j^u\}_{j=1}^N, tk' = K'^u, \{tk_x = K_x^u\}_{x \in S}$. The trapdoor is set as $TK = (S, \{tk_i\}_{i=1}^N, tk', \{tk_x\}_{x \in S}, T, T', \{T_j\}_{j=1}^m)$.

**Search**$(TK, I, QI, II, sk_S)$: Let $QI = \{q_1, \ldots, q_m\}$ be the mobile user's query interest with $TK$. As shown in Algorithm 1, the search process includes two phases.

[2]The hash function can be constructed as $H(x) = u^{h(x)} v^{g(x)}$, where $u, v \in \mathbb{G}_1$, $h$, and $g$ are hash functions that map $\{0,1\}^*$ to $\mathbb{Z}_p$.

*First*, the algorithm first checks whether the attributes match. It checks whether $S$ can satisfy $\mathcal{T}$, if not, it terminates the search process; or else, it continues to perform the following two steps. If $x$ is a leaf node and $x \in S$, let $k = \mathrm{att}(x)$, it computes $\psi_x = e(C_x, tk') \cdot e(tk_k, C) = e(h, g_2)^{urq_x(0)}$. If $x \notin S$, $\psi_x = \perp$.

If $x \in \mathcal{T}$ is a nonleaf node, $\psi_x$ can be calculated by recursive algorithm $\psi_z$, where $z$ is the children node of $x$. Let $S_x$ be an arbitrary $k_x$-sized set of children nodes $z$ such that $\psi_z \neq \perp$, if there is no such set, then $\psi_x = \perp$; or else, let $j = index(z)$, $S'_x = \{index(z) | z \in S_x\}$ and calculate $\psi_x$ as

$$\begin{aligned}\psi_x &= \prod_{z \in S_x} \psi_z^{\Delta_{j,S'_x(0)}} \\ &= \prod_{z \in S_x} \left((h, g_2)^{urq_{parent(z)}(index(z))}\right)^{\Delta_{j,S'_x}(0)} \\ &= e(h, g_2)^{urq_x(0)}.\end{aligned}$$

Recursively, it computes $\psi_R = e(g_1, h)^{rsu}$, then it computes

$$\Psi_i = e(tk_i, C)/\psi_R = e(g_1, g_2)^{\alpha\beta_i su}, i \in [1, n].$$

*Second*, the algorithm accelerates the search efficiency by inverted index.It sets an array $Score = (s_1, \ldots, s_l)$, where $s_k = 0, k \in [1, l]$, it denotes the score of $F_k$. The server computes as follows:

$$\begin{aligned}A_i &= e(W_i, T) = e\left(g_1^{\beta_i s} H(w_i)^d, g_2^{\alpha u}\right) \\ B_j &= e\left(T_j, W_0\right) \cdot e\left(T'^{1/a}, D_i\right) = e\left(H\left(w_j\right)^u, g_2^{-\alpha d}\right). \quad (1)\end{aligned}$$

For $i \in [1, n], j \in [1, m]$, it multiplies $A_i$ and $B_j$, if the result equals to $\Psi_i$, it computes $\{s_k = s_k + q_j, \sigma_k \in I_{w_i}\}$ and updates them in *Score*. Finally, the set of identifiers of selected encrypted files $ID = \{\sigma'_1, \ldots, \sigma'_k\}$ is returned, where $\sigma'_i \in \Sigma, i \in [1, k]$, and its score is one of the top-k scores in *Score*.

In our scheme as Fig. 3, DP provides data for the DU query, and the Server helps to complete the main calculation work. Among them, DP runs the **EncIndex** algorithm to generate the Index $I$ and sends $I$ to the Server. The Server runs the **Keygen** algorithm to generate $SK$ and sends it to DU When DU wants to query the corresponding keyword, it executes the **Trap** algorithm and sends the obtained trapdoor $TK$ to the Server. After the Server executes **Search** as Algorithm 1, it returns the results to DU.

*Correctness Analysis:* Assuming $f(S, \mathcal{T}) = 1$ and $w_i = w'_j$, the validity of the search process can be verified as follows:

$$\begin{aligned}\Psi_i &= e(tk_i, C)\psi_R^{-1} \\ &= e\left(g_1^{\alpha\beta_i u} h^{ru}, g_2^s\right) e(h, g_2)^{-rus} \\ &= e(g_1, g_2)^{\alpha\beta_i su} \\ A_i &= e(W_i, T) \\ &= e\left(g_1^{\beta_i s} H(w_i)^d, g_2^{\alpha u}\right) \\ &= e(g_1, g_2)^{\alpha\beta_i us} e(H(w_i), g_2)^{\alpha ud}\end{aligned}$$

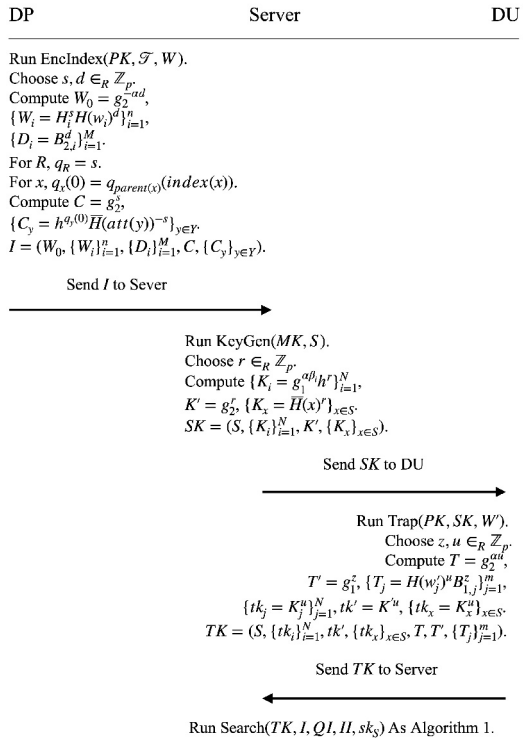


Fig. 3. Flow of IABKS.

**Algorithm 1 Search**$(TK, I, QI, II, sk_S)$

**Input:** Trapdoor $TK$, index $I$, query interest $QI$, inverted index $II$, secret key of server $sk_S$.
**Output:** The set of identifiers of selected encrypted files $ID$.

```
Check whether f(S, T) = 1
if f(S, T) ≠ 1 then
    ⊥
else
    Recursively compute ψ_R
    Set Score = (s_1, ..., s_l) = (0, ..., 0)
    for i ∈ [1, n] do
        Compute Ψ_i and A_i
        for j ∈ [1, m] do
            Compute B_j
            if A_i · B_j = Ψ_i then
                Compute {s_k} and update Score
                break
            end if
        end for
    end for
    return ID
end if
```

$$\begin{aligned} B_j &= e\left(T_j, W_0\right) e\left(T'^{1/a}, D_j\right) \\ &= e\left(H\left(w_j'\right)^u B_{1,j}^z, g_2^{-\alpha d}\right) e\left(g_1^z, B_{2,j}^d\right)^{1/a} \\ &= e\left(H\left(w_j'\right), g_2\right)^{-\alpha u d}. \end{aligned}$$

Hence, $A_i \cdot B_j = e(g_1, g_2)^{\alpha\beta_i s u} = \Psi_i$

## V. Security Analysis

In this section, we use the following three theorems to formally prove the security of the IABKS scheme.

*Theorem 1:* $\xi_1, \xi_2, \xi_T, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T$, and $e$ are defined in Section II-B. Assume that $q$ is the upper limit of the sum of all the queries of $\mathcal{A}_1$, then the advantage of $\mathcal{A}_1$ over $\mathcal{A}_1$ in the CKA game is $O(q^2/p)$.

*Proof:* In the CKA security game, $I^*$ has the components which are randomly either $\{W_i\}_{w_i \in W_0}$ or $\{W_i\}_{w_i \in W_1}$. Let us consider a modified game, assume that $W = W_0 \cup W_1 = \{w_1, \ldots, w_n\}$, $W_\xi$ is either $g_1^{\beta_\xi s} H(w_\xi)^d$ or $g_1^\theta$, where $\xi \in [1, n]$, $\theta \in_R \mathbb{F}_p$. Given the challenge index $I^* = (W_0, \{W_i\}_{i=1}^n, \{D_i\}_{i=1}^M, C, \{C_y\}_{y \in Y})$, $\mathcal{A}_1$ must decide whether $W_\xi$ is $g_1^{\beta_\xi s} H(w_\xi)^d$ or $g_1^\theta$. It is not difficult to draw a conclusion that if $\mathcal{A}_1$ has advantage $\epsilon$ in the CKA security game, then $\mathcal{A}_1$ has at least $\epsilon/2$ in the modified game. The following proof analyzes the advantage of $\mathcal{A}_1$ in the modified game.

**Setup**: The simulator $\mathcal{B}$ selects $g_1 \in_R \mathbb{G}_1$, $g_2 \in_R \mathbb{G}_2$, $\alpha, \{\beta_i\}_{i=1}^N, a, h_0, \{b_i\}_{i=1}^M \in_R \mathbb{Z}_p$, then $PK = (g_1, g_2, h = g_1^{h_0}, \{g_1^{\beta_i}\}_{i=1}^N, g_2^\alpha, \{B_{1,i} = g_1^{b_i}, B_{2,i} = g_2^{a\alpha b_i}\}_{i=1}^M)$ and $sk_S = a$ are sent to $\mathcal{A}_1$.

If attribute $j$ has not been queried before, $\mathcal{B}$ returns $g_2^{\varphi_j}$ and stores $(j, \varphi_j)$ into the list $\overline{H}$; otherwise, it returns $g_1^{\varphi_j}$ by retrieving $\varphi_j$ from $\overline{H}$.

If keyword $w_i$ has not been queried before, $\mathcal{B}$ returns $g_1^{\eta_i}$ and stores $(i, \eta_i)$ into the list $H$; otherwise, it returns $g_1^{\eta_i}$ by retrieving $\eta_i$ from $H$.

*Phase 1:* $\mathcal{B}$ creates $E$, $D$, and $L_{kw}$ according to the definition, and sets $j = 0$. $\mathcal{A}_1$ can access the two oracles adaptively (take the $t$th query as an example).

1) $\mathcal{O}_{\mathbf{KeyGen}}(S)$: $\mathcal{B}$ chooses $r^{(t)} \in_R \mathbb{F}_p$ and computes $SK = (S, \{K_i = g_1^{\alpha\beta_i} h^{r^{(t)}}\}_{i=1}^N, K' = g_2^{r^{(t)}}, \forall j \in S : K_j = g_1^{\varphi_j r^{(t)}})$, it makes $D = D \cup \{S\}$ and sends $SK$ to $\mathcal{A}_1$.
2) $\mathcal{O}_{\mathbf{Trap}}(S, W')$: Assuming $W' = \{w_1', \ldots, w_m'\}$, $\mathcal{B}$ sets $j = j + 1$, it accesses $\mathcal{O}_{\mathbf{KeyGen}}(S)$ to obtain $SK$, and chooses $u^{(t)}, z^{(t)} \in_R \mathbb{F}_q$. It stores $(j, S, W')$ in $E$ and sends $TK = (S, T = g_2^{\alpha u^{(t)}}, T' = g_1^{z^{(t)}}, \{T_j = g_1^{\eta_j u^{(t)}} B_{1,j}^{z^{(t)}}\}_{j=1}^m, \{tk_i = K_i^{u^{(t)}}\}_{j=1}^N, tk' = K'^{u^{(t)}}, \{tk_x = K_x^{u^{(t)}}\}_{x \in S})$ to $\mathcal{A}_1$.

*Challenge:* $\mathcal{A}_1$ submits two keywords sets $W_0$ and $W_1$, a keyword $w_\xi$, and an access policy $\mathcal{T}^*$ to $\mathcal{B}$. The requirements are as follows.

1) $w_\xi \in W = W_0 \cup W_1 = \{w_1, \ldots, w_n\}$ and $|W_0| = |W_1|$.
2) $\forall S \in D, f(S, \mathcal{T}^*) \neq 1$.
3) For each entry in $E$, if $f(S, \mathcal{T}^*) = 1$, adds $W'$ to $L_{kw}$; then $\forall W' \in L_{kw}, w_\xi \notin W'$.

$\mathcal{B}$ chooses $s, d, \theta \in_R \mathbb{F}_p$ and uses a linear sharing technique related to $\mathcal{T}^*$ to generate shares $\lambda_j$ of $s$ for each related attribute $j$. It then computes $C = g_2^s, \{C_j = h^{\lambda_j} g_1^{-\varphi_j s}\}_{j \in \mathcal{T}^*}, W_0 = g_2^{-\alpha d}, \{W_i = g_1^{\beta_i s + \eta_i d}\}_{i=1}^n, \{D_i = B_{2,i}^d\}_{i=1}^M$ except that $W_\xi = g_1^\theta$. $\mathcal{B}$ returns challenge index to $\mathcal{A}_1$.

*Phase 2:* $\mathcal{A}_1$ continues to ask oracles as in Phase 1. The constraint at this stage is that $f(S, \mathcal{T}) = 1$.

1) $\mathcal{A}_1$ cannot access $\mathcal{O}_{\mathbf{KeyGen}}$.

TABLE II
POSSIBLE $\mathbb{G}_1$ AND $\mathbb{G}_2$ QUERIES FOR $\mathcal{A}_1$

| | $\mathbb{G}_1$ | | | | | $\mathbb{G}_2$ | | |
|---|---|---|---|---|---|---|---|---|
| PK | $\beta_i$ | $b_i$ | $\eta_i$ | $\varphi_j$ | $h_0$ | $\alpha$ | $a\alpha b_i$ | |
| Index | $\beta_i s + \eta_i d$ | | | $h_0\lambda_i - \varphi_i s$ | | $s$ | $a\alpha b_i d$ | $-\alpha d$ |
| SK | $\alpha\beta_i + h_0 r^{(t)}$ | | | $\varphi_j r^{(t)}$ | | $r^{(t)}$ | | |
| TK | $\eta_j u^{(t)} + b_j z^{(t)}$ | | | $z^{(t)}$ | $r^{(t)}u^{(t)}$ | $\alpha u^{(t)}$ | | |
| | $(\alpha\beta_j + h_0 r^{(t)})u^{(t)}$ | | | $\varphi_j r^{(t)} u^{(t)}$ | | | | |

TABLE III
SYMBOLS INVOLVED IN EFFICIENCY COMPARISON

| Notation | Definition |
|---|---|
| $\mathbb{G}_i$ | Modular exponential operation in group $\mathbb{G}_i$, $(i = 1, 2, T)$ |
| $P$ | Bilinear pairing operation |
| $n$ | Number of keywords in the files under the same $\mathcal{T}$ |
| $m$ | Number of keywords in an trapdoor |
| $N$ | Maximum number of keywords in the files under the same $\mathcal{T}$ |
| $M$ | Maximum number of keywords in a trapdoor |
| $\|S\|$ | Number of user's attributes |
| $\|\mathcal{T}\|$ | Number of attributes in $\mathcal{T}$ |
| $s$ | Number of used attributes in the search process |
| $L_*$ | Length of element in * |

2) $(S, W)$ cannot be the input to $\mathcal{O}_{\mathbf{Trap}}$ if $w_\xi \in W$.

Next, we analyze the probability of the accidental collisions. Since every oracle query can be treated as a rational function $\vartheta = \kappa/\psi$, where $\kappa$ and $\psi$ are selected from related variables. The conflict occurs when two queries corresponding to two different rational functions are mapped to the same output. Referring to [11], the probability of such an event occurring is at most $\mathcal{O}(q^2/p)$. Table II lists all possible $\mathbb{G}_1$ and $\mathbb{G}_2$ queries for $\mathcal{A}_1$. This proof will consider that $\mathcal{A}_1$ has variable $sk_S = a$ though it is not listed for the sake of simplicity. Next, we use the following two lemmas to prove that $\mathcal{A}_1$ can distinguish $g_1^\theta$ and $g_1^{\beta_\xi s} H(w_\xi)^d$ with a negligible advantage. ■

*Lemma 1:* $\mathcal{A}_1$ cannot construct a $TK$ with $W$ and $S$ such that $f(S, \mathcal{T}^*) = 1$ and $w_\xi \in W$.

*Proof:* This lemma can be proved from two aspects.

1) Given that the attribute set of $TK$ is $S$ and $f(S, \mathcal{T}^*) = 1$, $\mathcal{A}_1$ cannot generated a term $\eta_\xi u^{(t)} + z^{(t)} b_\xi$ in $\mathbb{G}_1$. From Table II, it is easy to find that $\mathcal{A}_1$ cannot construct that term.
2) $\mathcal{A}_1$ cannot construct $SK$ whose $S$ satisfies $f(S, \mathcal{T}^*) = 1$, that is equivalent to prove that $\mathcal{A}_1$ cannot construct $e(g_1, g_2)^{\gamma\alpha\beta_\xi s}$, $\gamma \in \mathbb{F}_p$.

The "TK" part can be ignored in this section, because the variables chosen in $\mathcal{O}_{\mathbf{Trap}}$ are independent of the target. It can be seen from Table II there is only one *possible* type of output that contains $\alpha\beta_\xi s$ in $\mathbb{G}_T$.

It is $(\alpha\beta_\xi + h_0 r^{(t)}) \cdot s = \alpha\beta_\xi s + h_0 r^{(t)} s$, then $\mathcal{A}_1$ needs to cancel out the term $h_0 r^{(t)} s$. From Table II, for a set $T$ and $\gamma_j \neq 0$, we analyze that $\mathcal{A}_1$ can only get $h_0 r^{(t)} s$ from the following formula:

$$\sum_{j \in T} \gamma_j \Big[ \big(h_0\lambda_j - \varphi_j s\big) \cdot r^{(t)} + \varphi_j r^{(t)} \cdot s \Big] = \gamma h_0 r^{(t)} s. \tag{2}$$

Nevertheless, this cannot be achieved because in the security game, it requires $\forall S \in D, f(S, \mathcal{T}^*) \neq 1$; therefore, there is no set $T$ supporting the reconstruction of the secret $s$. Therefore, we can conclude that $\mathcal{A}_1$ cannot construct $e(g_1, g_2)^{\gamma\alpha\beta h_\xi s}$. ■

*Lemma 2:* $\mathcal{A}_1$ can construct $e(g_1, g_2)^{\delta(\beta_\xi s + \eta_\xi d)}$ for some $g_2^\delta$ that can be made up of the oracles outputs with a negligible probability.

*Proof:* Our goal is to query $\Delta = \delta(\beta_\xi s + \eta_\xi d)$ in $\mathbb{G}_T$ for some $\delta$ in $\mathbb{G}_2$.

For the term $\delta\eta_\xi d$ in $\mathbb{G}_T$, we analyze it from four aspects.

1) $\mathcal{A}_1$ pairs $\eta_\xi$ with $-\alpha d$ to get $-\alpha\eta_\xi d$; thus, $\delta = \alpha$. Then, $\mathcal{A}_1$ needs to construct $\delta\beta_\xi s = \alpha\beta_\xi s$ in $\mathbb{G}_T$, however it has been proved in Lemma 1 that it cannot be constructed.
2) $\mathcal{A}_1$ pairs $\eta_\xi$ with $a\alpha b_\xi d$ to get $a\alpha b_i d\eta_\xi$, consider $\mathcal{A}_1$ has the variable $a$, $\delta = \alpha b_\xi$. However, from Table II, we can conclude that the term $\alpha\beta_\xi b_\xi s$ in $\mathbb{G}_T$ cannot be constructed.
3) $\mathcal{A}_1$ pairs $\eta_\xi u^{(t)} + z^{(t)} b_\xi$ with $-\alpha d$ to get $-\alpha\eta_\xi d u^{(t)} - b_\xi \alpha d z^{(t)}$, thus $\delta = \alpha u^{(t)}$. Similar with case 1), the term $\delta\beta_\xi s = \alpha\beta_\xi s u^{(t)}$ cannot be constructed by $\mathcal{A}_1$.
4) $\mathcal{A}_1$ pairs $\eta_\xi u^{(t)} + z^{(t)} b_\xi$ with $a\alpha b_\xi d$, $\delta = \alpha u^{(t)} b_\xi$, from Table II, we can find that the term $\delta\beta_\xi s$ cannot be constructed.

Therefore, Lemma 2 is proved and $\mathcal{A}_1$ can tell the difference between $g_1^{\beta_\xi s} H(w_\xi)^d$ and $g_1^\theta$ with a negligible advantage, further, we can conclude that the advantage of $\mathcal{A}_1$ in the CKA game for $\mathcal{A}_1$ is bounded by $\mathcal{O}(q^2/p)$. ■

*Theorem 2:* Letting $\xi_1, \xi_2, \xi_T, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e$, and $q$ be defined in Theorem 1, then we have that the advantage of $\mathcal{A}_2$ in the CKA security game for $\mathcal{A}_2$ is $O(q^2/p)$.

The proof of Theorem 2 is similar to that of Theorem 1, so it is omitted here.

*Theorem 3:* Let $\xi_1, \xi_2, \xi_T, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e$, and $q$ be defined in Theorem 1, then we have that the advantage of $\mathcal{A}_2$ in the IND-KGA security game for $\mathcal{A}_2$ is $O(q^2/p)$.

*Proof:* Similar with Theorem 1, we also consider a modified game, the challenge keyword set is $W = W_0 \cup W_1 = \{w_1, \ldots, w_m\}$, $T_\xi = H(w_\xi)^u B_{1,\xi}^z$ or $g_1^\theta$, where $\xi \in [1, m]$, $\theta \in_R \mathbb{F}_p$. Given $TK^* = (S^*, \{tk_i\}_{i=1}^N, tk', \{tk_j\}_{j \in S^*}, T, T', \{T_j\}_{j \in [1,m]})$, $\mathcal{A}_2$ must decide whether $T_\xi$ is $H(w_\xi)^u B_{1,\xi}^z$ or $g_1^\theta$.

The **Setup**, **Hash Queries**, and Phase 1 are the same with those in the proof of Theorem 1 except that:

1) in **Setup**, $\mathcal{A}_2$ does not receive $sk_S$;
2) $\mathcal{O}_{\mathbf{Search}}(I, TK, QI, II)$ is included in Phase 1.

*Challenge:* $\mathcal{A}_2$ submits two keyword sets $W_0$ and $W_1$, a keyword $w_\xi$ and $S^*$ to $\mathcal{B}$. It requires that $w_\xi \in W^* = W_1 \cup W_2 = \{w_1, \ldots, w_m\}$, and $|W_1| = |W_2|$. $\mathcal{B}$ chooses $\theta, z, u$, and $r$ and computes $\{tk_i = g_1^{\alpha\beta_i u} h^{ru}\}_{i=1}^N$, $tk_i' = g_2^{ru}$, $\{tk_j = g_1^{\varphi_j ru}\}_{j \in S^*}$, $T = g_2^{\alpha u}$, $T' = g_1^z$, $\{T_j = g_1^{\eta_j u + z b_j}\}_{j \in [1,m]}$ except that $T_\xi$ is computed as $g_1^\theta$. It submits $TK^*$ to $\mathcal{A}_2$.

*Phase 2:* $\mathcal{A}_2$ accesses oracles as in Phase 1 except that index $I$ with $\mathcal{T}$ and $W$ cannot be the input of $\mathcal{O}_{\mathbf{Search}}$ if $f(S^*, \mathcal{T}) = 1$ and $w_\xi \in W$.

As discussed in the proof of Theorem 1, the probability of unexpected collusion is $O(q^2/p)$. The only way for $\mathcal{A}_2$ to tell the difference between $g_1^\theta$ and $H(w_\xi)^u B_\xi^z$ is to construct $e(g_1, g_2)^{\delta(\eta_\xi u + z b_\xi)}$ for some $g_2^\delta$ that can be composed from

TABLE IV
STORAGE COST

| Scheme | SK Size | TK Size | Index Size |
|---|---|---|---|
| [16] | $\|S\|L_{\mathbb{G}_1} + (\|S\|+1)L_{\mathbb{G}_2}$ | $(\|S\|+1)L_{\mathbb{G}_1} + (\|S\|+2)L_{\mathbb{G}_2}$ | $(\|\mathcal{T}\|+2)nL_{\mathbb{G}_1} + (\|\mathcal{T}\|+1)nL_{\mathbb{G}_2}$ |
| IABKS | $(N+\|S\|)L_{\mathbb{G}_1} + L_{\mathbb{G}_2}$ | $(N+\|S\|+m+1)L_{\mathbb{G}_1} + 2L_{\mathbb{G}_2}$ | $(\|\mathcal{T}\|+n)L_{\mathbb{G}_1} + (M+2)L_{\mathbb{G}_2}$ |

TABLE V
COMPUTATIONAL COST

| Scheme | **KeyGen** | **EncIndex** | **Trap** | **Search** |
|---|---|---|---|---|
| [16] | $(\|S\|+1)\mathbb{G}_1 + (\|S\|+1)\mathbb{G}_2$ | $(\|T\|+3)n\mathbb{G}_1 + (\|\mathcal{T}\|+1)n\mathbb{G}_2$ | $(\|S\|+2)\mathbb{G}_1 + (\|S\|+2)\mathbb{G}_2$ | $(2s+3)nP + sn\mathbb{G}_T$ |
| IABKS | $(\|S\|+1)\mathbb{G}_1 + \mathbb{G}_2$ | $(2\|\mathcal{T}\|+3n)\mathbb{G}_1 + (2M+2)\mathbb{G}_2$ | $(N+\|S\|+3m+1)\mathbb{G}_1 + 2\mathbb{G}_2$ | $(2s+2m+2n)P$ $+s\mathbb{G}_T + \mathbb{G}_1$ |

the oracles outputs. Then, we prove that $\mathcal{A}_2$ cannot construct $\Delta = \delta(\eta_\xi u + z b_\xi)$ in $\mathbb{G}_T$ for some $\delta$ in $\mathbb{G}_2$.

For $\delta\eta_\xi u$ in $\mathbb{G}_T$, we analyze it from the following three aspects.

1) $\mathcal{A}_2$ pairs $\eta_\xi$ with $\alpha u$ to get $\eta_\xi \alpha u$, thus $\delta = \alpha$. For $\delta z b_\xi = \alpha z b_\xi$ in $\mathbb{G}_T$, there are two *possible* terms that contain $\alpha z b_\xi$, they are pairing $z$ with $a\alpha b_\xi$ to get $a\alpha z b_\xi$ or pairing $z$ with $a\alpha b_\xi d$ to get $a\alpha z b_\xi d$, however, the variable $a$ cannot be canceled out since $\mathcal{A}_2$ does not have $sk_S$.
2) $\mathcal{A}_2$ pairs $\eta_\xi$ with $\varphi_j u r$ to get $\eta_\xi \varphi_j u r$, thus we have $\delta = \varphi_j r$. However, from the Table II we can find that the term $\delta z b_\xi = \varphi_j r z b_\xi$ in $\mathbb{G}_T$ cannot be further constructed.
3) $\mathcal{A}_2$ pairs $\eta_\xi$ with $(\alpha\beta_j + h_0 r)u$, it is easy to examine this case is impossible to construct $\delta(\eta_\xi u + z b_\xi)$ in $\mathbb{G}_T$.

Therefore, $\mathcal{A}_2$ can *never* tell the difference between $H(w_\xi)^u B_\xi^z$ and $g_1^\theta$. Then, we conclude that the advantage of $\mathcal{A}_2$ in the KGA game for $\mathcal{A}_2$ is $\mathcal{O}(q^2/p)$. ■

## VI. PERFORMANCE ANALYSIS

This chapter analyzes and compares the efficiency of the ABCKS scheme with similar schemes from both theoretical and experimental aspects.

### A. Theoretical Analysis

A comparison between relevant ABKS schemes and the IABKS scheme is showed in Table I. Through the four aspects, we draw the conclusion that the IABKS scheme is friendly to blockchain data sharing. In terms of retrieve mode, [16] and [17] give the single keyword retrieval method, [24]–[26] give conjunctive keyword retrieval methods, and [27] provides the Boolean keyword search, [21] and the IABKS scheme provide multikeyword ranked search, which is more flexible and leads to better search accuracy and search efficiency. In terms of access policy, the access tree is used in schemes except [17], which means more flexible user access pattern. In addition, except for [21] and the IABKS scheme, other schemes cannot resist outside KGA. Finally, [17], [21], [24]–[26] and [27] deploy the forward index, and [16] and the IABKS scheme deploy the inverted index. Thus, the IABKS scheme can be applied in the context of massive data of IoT devices.

The efficiency of [16] and the IABKS scheme in storage cost and computational cost are, respectively, compared in Tables IV and V. The reason we only use [16] to compare with the IABKS scheme is that only these two schemes deploy inverted index.

The IABKS scheme uses asymmetric bilinear groups because several asymmetric instances of bilinear groups possess beneficial properties such as faster operations under the same security level, etc. [32]. To make a scientific and fair comparison, we adopt a general transformation method to transform [16] constructed on symmetric bilinear groups into those constructed on asymmetric bilinear groups, the security proofs of the two schemes can also be transformed in a general way. The symbols involved in the comparison are presented in Table III.

As shown in Table IV, the two schemes are compared in SK size, TK size, and encrypted index size, the three aspects can reflect the storage cost of the schemes from different aspects. The *SK* size and *TK* size in the IABKS scheme are $(N + |S|)L_{\mathbb{G}_1} + L_{\mathbb{G}_2}$ and $(N + |S| + m + 1)L_{\mathbb{G}_1} + 2L_{\mathbb{G}_2}$, respectively. In the majority of cases $m \ll |S|$, and $L_{\mathbb{G}_2}$ is three times as long as $L_{\mathbb{G}_2}$; therefore, the SK size and TK size in the two schemes are close. In terms of the encrypted index size, [16] needs $(|\mathcal{T}|+2)nL_{\mathbb{G}_1} + (|\mathcal{T}|+1)nL_{\mathbb{G}_2}$ to contain $n$ keywords. Since $n$ is always a large number, the encrypted index size in [16] is much larger than that in the IABKS scheme, and with the increase of $n$, the advantage of the IABKS scheme will become more and more obvious.

In Table V, we compare the computational cost of the two schemes in terms of the four algorithms: 1) **KeyGen**; 2) **EncIndex**; 3) **Trap**; and 4) **Search**. Similar to the analysis in Table IV, it shows that the computational cost of **KeyGen** algorithm and **Trap** algorithm in the two schemes are close. Since $n \gg 1$, the computational cost of **EncIndex** algorithm and **Search** algorithm in the IABKS scheme is much less than that in [16]. The main reason for this huge gap is that [16] needs to generate $n$ separate encrypted indices to cover $n$ keywords, and in such an approach, the computation related to access control is computed $n-1$ extra times. Although [16] (VABKS scheme) solves the problem of verifying the search result in ABKS, it brings enormous storage cost and computational cost.

### B. Experimental Analysis

We want to use experiments to verify that our scheme is very suitable for IoT devices and the blockchain network environment. This also requires our DP and Du to undertake less work.

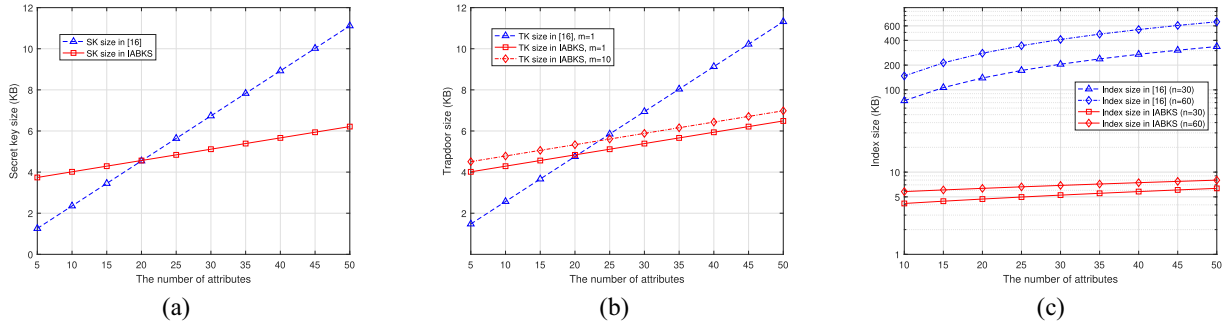


Fig. 4. Storage cost comparison of [16] and IABKS. (a) Secret key size. (b) Trapdoor size. (c) Encrypted index size.

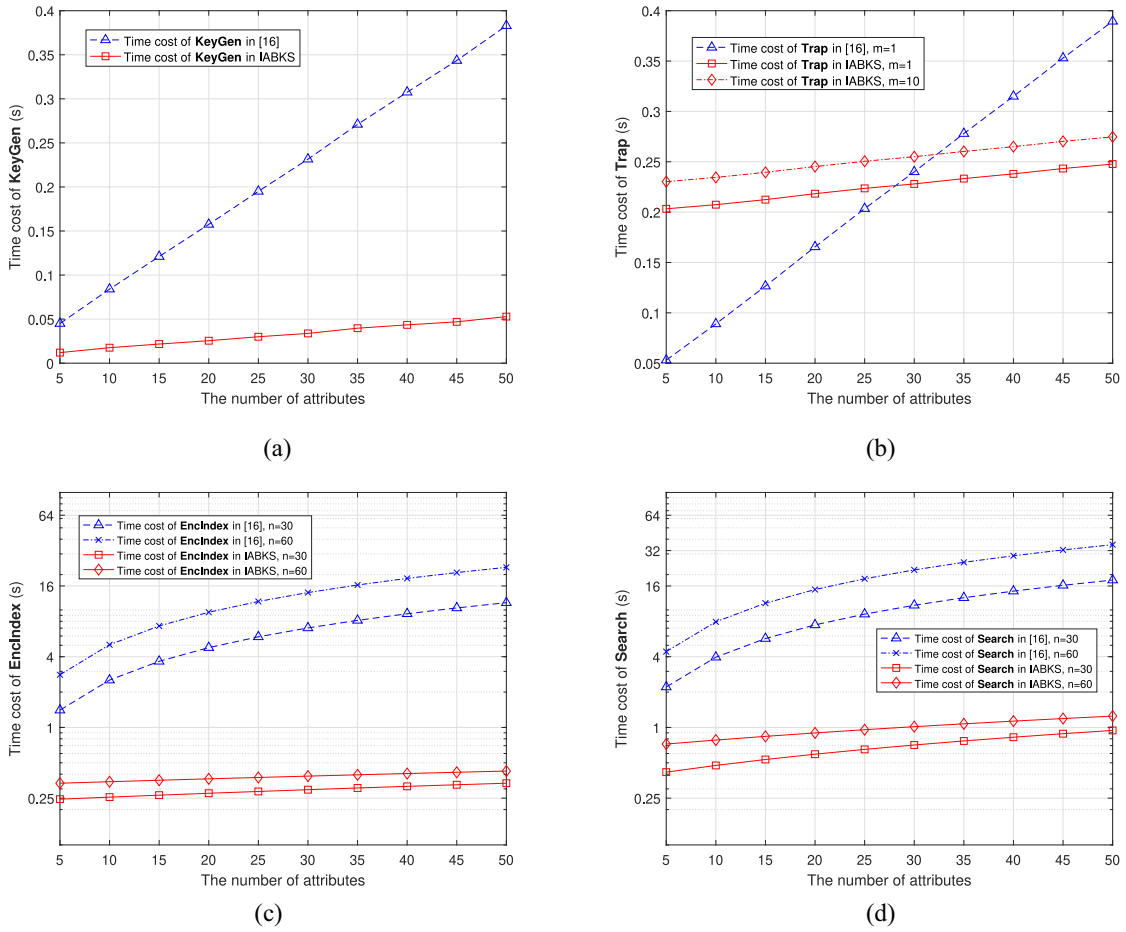


Fig. 5. Computational cost comparison of [16] and IABKS. (a) Time cost of **KeyGen**. (b) Time cost of **Trap**. (c) Time cost of **EncIndex**. (d) Time cost of **Search**.

To compare performance of [16] and the IABKS scheme, we carry out simulations on the two schemes. In the simulation, we use the real-world request for comments (RFCs) database as our test data set. It is based on Charm [31]. Charm is developed based on Python, which is helpful to the rapid prototyping of cryptographic mechanisms and protocols. We load the asymmetric bilinear groups "MNT224" in the PBC library to realize asymmetric bilinear mapping. We select 28 and 57 files from the database and extract 30 and 60 keywords, respectively, ($n = 30$ and $n = 60$). To simplify the comparison, we set $N = 60$ and $M = 10$ and set $m = 1$ and $m = 10$, respectively, the attribute number in the simulation is $\{5, 10, \ldots, 50\}$. All the tests are performed on a computer with an Intel Core i5-4590 CPU at 3.30 GHz and 6-GB RAM. All simulation results are measured 100 times and averaged. The unit of storage cost and computational cost are, respectively, kilobyte and second.

The subgraphs of Fig. 4, respectively, present the relationship between the attribute number and the secret key size, trapdoor size, and encrypted index size. We observe that for the two schemes, the secret key size, trapdoor size, and encrypted index size all approximately linearly increase as the number of attributes increases [note the logarithmic $y$-axis scale in Fig. 4(c)], and the IABKS scheme has a lower growth rate in

all the above aspects. As shown in Fig. 4(a), at the beginning, the secret key size in the IABKS scheme is larger than that in [16], but when the attribute number is bigger than 20, the secret key size in the IABKS scheme is less than that in [16], and the more the number of user's attributes, the more obvious the advantages. A similar analysis is applicable to the trapdoor size in Fig. 4(b). We can also observe that the increase of $m$ leads to a small increase in trapdoor size, and when the attribute number is about 21 and 24, the blue line intersects the two red lines ($m = 1$ and $m = 10$), respectively. From Fig. 4(c), we can find that whether $n = 30$ or $n = 60$, the encrypted index size in [16] is much higher than that in the IABKS scheme, when the number of attributes is 50 and $n = 60$, the encrypted index size in [16] is up to 0.66 MB, which is almost 82 times of the encrypted index size in the IABKS scheme in the same situation.

The four subgraphs in Fig. 5, respectively, display the time cost of the four algorithms **KeyGen**, **Trap**, **EncIndex**, and **Search** versus the number of attribute. The four items tested in the two schemes have an approximate linear relationship with the number of attributes [note the logarithmic y-axis scale in Fig. 5(c) and (d)]. Fig. 5(a) shows that the secret key generation time in [16] is bigger than that in the IABKS scheme. From Fig. 5(b), we can observe that when the number of attributes is small, the IABKS scheme costs more time to generate the trapdoor, but after the number of attributes is more than about 28 and 32, respectively, the situation begins to reverse. From Figs. 4(c) and 5(d), we note that whether $n = 30$ or $n = 60$, [16] requires more time to generate the encrypted index and execute the search process. We can also find that the increase in the number of keywords brings more computational cost to [16]. Indeed, the growth rates of the lines related to [16] in the four subfigures are larger than those related to the IABKS scheme, and some of which are much larger. We can conclude that the results of our simulation are consistent with the theoretical analysis shown in Section VI-A2. In particular, the experimental results fully demonstrate that our scheme is extremely friendly for IoT devices, and can meet the computing needs of the blockchain network environment.

## VII. Conclusion

In this article, we wanted to solve the problem of privacy protection data retrieval of IoT devices in the 6G era, and we can quickly retrieve the data through the security protocol on the premise of protecting data privacy. So, we proposed an IABKS scheme to resist outside KGA in the inverted-based ABKS scheme. The retrieval mode of the proposed scheme is searched with the multikeyword ranked result, which can enhance the retrieval accuracy and user's experience. We established formal security models and the IABKS scheme was proved to be IND-CKA and IND-KGA secure. In the process of scheme design and role selection, we made some compromises in functionality, security, and efficiency. Of course, the ideal scheme should have better security and efficiency, which will be the content of our continuous research. The last but not least, simulations on actual data set showed that our proposed scheme is more efficient than related ABKS schemes, especially in an IoT environment, hence, more suitable for blockchain of IoT in the 6G era.

## References


[1] D. Yu *et al.*, "Distributed broadcasting in dynamic networks," *IEEE/ACM Trans. Netw.*, vol. 29, no. 5, pp. 2142–2155, Oct. 2021, doi: 10.1109/TNET.2021.3087818.

[2] T. Nguyen, N. Tran, L. Loven, J. Partala, M. T. Kechadi, and S. Pirttikangas, "Privacy-aware blockchain innovation for 6G: Challenges and opportunities," in *Proc. 2nd 6G Wireless Summit (6G SUMMIT)*, Mar. 2020, pp. 1–5.

[3] Y. Zou, D. Yu, J. Yu, Y. Zhang, F. Dressler, and X. Cheng, "Distributed Byzantine-resilient multiple-message dissemination in wireless networks," *IEEE/ACM Trans. Netw.*, vol. 29, no. 4, pp. 1662–1675, Aug. 2021.

[4] Y. Lu and X. Zheng, "6G: A survey on technologies, scenarios, challenges, and the related issues," *J. Ind. Inf. Integr.*, vol. 19, Sep. 2020, Art. no. 100158.

[5] M. Xu, C. Liu, Y. Zou, F. Zhao, J. Yu, and X. Cheng, "wChain: A fast fault-tolerant blockchain protocol for multihop wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 10, pp. 6915–6926, Oct. 2021, doi: 10.1109/TWC.2021.3078639.

[6] D. Yu, Y. Zou, Y. Wang, J. Yu, X. Cheng, and F. C. M. Lau, "Implementing abstract MAC layer via inductive coloring in the Rayleigh-fading model," *IEEE Trans. Wireless Commun.*, vol. 20, no. 9, pp. 6167–6178, Sep. 2021.

[7] D. Yu, Y. Zou, Y. Zhang, H. Sheng, W. Lv, and X. Cheng, "An exact implementation of the abstract MAC layer via carrier sensing in dynamic networks," *IEEE/ACM Trans. Netw.*, vol. 29, no. 3, pp. 994–1007, Jun. 2021.

[8] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2004, pp. 506–522.

[9] Z. Guo, H. Zhang, C. Sun, Q. Wen, and W. Li, "Secure multi-keyword ranked search over encrypted cloud data for multiple data owners," *J. Syst. Softw.*, vol. 137, pp. 380–395, Mar. 2018.

[10] 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. Security*, 2006, pp. 89–98.

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

[12] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of ABE ciphertexts," in *Proc. USENIX Security Symp.*, vol. 2011, no. 3, 2011, p. 34.

[13] Y. Chen, Q. Wen, W. Li, H. Zhang, and Z. Jin, "Generic construction of outsourced attribute-based encryption without key escrow," *IEEE Access*, vol. 6, pp. 58955–58966, 2018.

[14] T. V. X. Phuong, G. Yang, and W. Susilo, "Hidden ciphertext policy attribute-based encryption under standard assumptions," *IEEE Trans. Inf. Forensics Security*, vol. 11, pp. 35–45, 2016.

[15] J. Ning, Z. Cao, X. Dong, and L. Wei, "White-box traceable CP-ABE for cloud storage service: How to catch people leaking their access credentials effectively," *IEEE Trans. Dependable Secure Comput.*, vol. 15, no. 5, pp. 883–897, Sep./Oct. 2018.

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

[17] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, 2014, pp. 226–234.

[18] J. W. Byun, H. S. Rhee, H.-A. Park, and D. H. Lee, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," in *Proc. Workshop Secure Data Manage.*, 2006, pp. 75–83.

[19] I. R. Jeong, J. O. Kwon, D. Hong, and D. H. Lee, "Constructing PEKS schemes secure against keyword guessing attacks is possible?" *Comput. Commun.*, vol. 32, no. 2, pp. 394–396, 2009.

[20] Y. Miao, J. Ma, X. Liu, F. Wei, Z. Liu, and X. A. Wang, "M$^2$-ABKS: Attribute-based multi-keyword search over encrypted personal health records in multi-owner setting," *J. Med. Syst.*, vol. 40, no. 11, p. 246, 2016.

[21] Y. Chen, W. Li, F. Gao, Q. Wen, H. Zhang, and H. Wang, "Practical attribute-based multi-keyword ranked search scheme in cloud computing," *IEEE Trans. Services Comput.*, early access, Dec. 18, 2019, doi: 10.1109/TSC.2019.2959306.

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

[23] D. E. Knuth, "The art of computer programming," in *Fundamental Algorithms*. vol. 1, 3rd ed. Redwood City, CA, USA: Addison Wesley Longman Publ., 1997.

[24] 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.

[25] Y. Miao, J. Ma, X. Liu, J. Weng, H. Li, and H. Li, "Lightweight fine-grained search over encrypted data in fog computing," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 772–785, Sep./Oct. 2019.

[26] 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.*, vol. 13, no. 6, pp. 985–998, Nov./Dec. 2020.

[27] K. He, J. Guo, J. Weng, J. Weng, J. K. Liu, and X. Yi, "Attribute-based hybrid Boolean keyword search over outsourced encrypted data," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 6, pp. 1207–1217, Nov./Dec. 2020.

[28] H. S. Rhee, W. Susilo, and H. J. Kim, "Secure searchable public key encryption scheme against keyword guessing attacks," *IEICE Electron. Exp.*, vol. 6, no. 5, pp. 237–243, 2009.

[29] M. Ambrona, G. Barthe, R. Gay, and H. Wee, "Attribute-based encryption in the generic group model: Automated proofs and new constructions," in *Proc. ACM SIGSAC Conf. Comput. Commun. Security* Oct. 2017, pp. 647–664.

[30] V. Shoup, "Lower bounds for discrete logarithms and related problems," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, 1997, pp. 256–266.

[31] J. A. Akinyele *et al.*, "Charm: A framework for rapidly prototyping cryptosystems," *J. Cryptogr. Eng.*, vol. 3, no. 2, pp. 111–128, 2013.

[32] Y. Rouselakis and B. Waters, "Efficient statically-secure large-universe multi-authority attribute-based encryption," in *Proc. Int. Conf. Financ. Cryptogr. Data Security*, 2015, pp. 315–332.