Privacy-Protecting Attribute-Based Conjunctive Keyword Search Scheme in Cloud Storage

Yang Chen[1], *Yang Liu*[1*], *Jin Pan*[1], *Fei Gao*[2], *Emmanouil Panaousis*[3]

[1] *National Computer Network Emergency Response Technical Team/Coordination Center of China, China*
[2] *State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, China*
[3] *Cyber Risk Lab, University of Greenwich, UK*
cylovehehe@163.com, liuyangcert@163.com, panjincert@163.com, gaof@bupt.edu.cn, e.panaousis@greenwich.ac.uk

Abstract

Cloud storage has been deployed in various real-world applications. But how to enable Internet users to search over encrypted data and to enable data owners to perform fine-grained search authorization are of huge challenge. Attribute-based keyword search (ABKS) is a well-studied solution to the challenge, but there are some drawbacks that prevent its practical adoption in cloud storage context. First, the access policy in the index and the attribute set in the trapdoor are both in plaintext, they are likely to reveal the privacy of data owners and users. Second, the current ABKS schemes cannot provide multi-keyword search under the premise of ensuring security and efficiency.

We explore an efficient way to connect the inner product encryption with the access control mechanism and search process in ABKS, and propose a privacy-protecting attribute-based conjunctive keyword search scheme. The proposed scheme provides conjunctive keyword search and ensures that the access policy and attribute set are both fully hidden. Formal security models are defined and the scheme is proved IND-CKA, IND-OKGA, access policy hiding and attribute set hiding. Finally, empirical simulations are carried out on real-world dataset, and the results demonstrate that our design outperforms other existing schemes in security and efficiency.

Keywords: ABKS, Hidden access policy, Hidden attribute set, Conjunctive keyword search, Cloud storage

1 Introduction

In the era of big data, many individuals and industries follow the trend of remotely storing their data to cloud servers, so that they can embrace the advantages of cloud storage, greatly reducing the cost of data maintenance and management, enjoying high-speed retrieval services, etc [1]. Meanwhile, data owner (DO) may not fully trust the cloud server, to protect data privacy, data must be encrypted before being outsourced into the cloud. However, the mechanism of encryption before outsourcing inevitably weakens the flexibility of data retrieval to a certain extent. Moreover, given the actual needs of data owners, there is an urgent need for fine-grained access control to encrypted data. Accordingly, exploring a way for a data owner/user to securely and efficiently retrieve over encrypted data, and for a DO to perform fine grained access control is of extreme importance in the scenarios of cloud storage.

Searchable encryption (SE) [2-3] enables the cloud server to help users search on encrypted data without knowing the keywords. Therefore, it has become a research hotspot in the field of secure storage in recent years. In order for the data owner in the SE scheme to perform fine-grained search authorization, Attribute-based keyword search (ABKS) is put forward [4-5]. Since its introduction, ABKS has received widespread attention from academia and industry due to its potential values on cloud-based applications.

In an ABKS scheme, the data owner determines the keywords and access policy in the index, and uploads the encrypted data and index to the server. The user sends server a trapdoor which includes user's attributes set and the keywords to be queried. Only when the attribute set satisfies the access policy and the keywords in the trapdoor and index matches with each other, the server will return the corresponding encrypted data to the user. However, there are two security drawbacks preventing the practical adoption of ABKS in cloud storage.

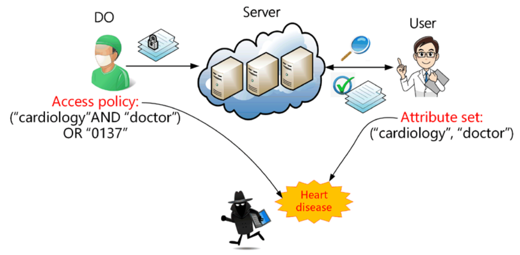


Figure 1. An example of privacy leakage in access policy and attribute set

The first problem is that the access policy in the index and the attribute set in the trapdoor may leak the privacy of data owners and users. The reason is that the access policy and the attribute set are both in plaintext, the adversary may be able to infer the sensitive information from them. Take an electronic medical system as an example, the doctor encrypts the case of the patient Cindy (ID: 0136) and

*Corresponding Author: Yang Liu; E-mail: liuyangcert@163.com
DOI: 10.53106/160792642023012401007

constructs an index. The access policy of the index is shown in Figure 1, only if the attribute set in trapdoor includes the attributes "cardiology" and "doctor", or includes the attribute "id: 0137" can satisfy the access policy. Once Alice (user or server) obtains the index, even if she cannot search effectively, she can still guess that Cindly may have heart disease. Hence, the access policy in the index may leak the privacy of the data owner.

From another perspective, if Alice intercepts the trapdoor of Kimi or Steven, even if Alice cannot know the keywords in the trapdoor, she can still guess that Kimi may be a cardiologist and Steven maybe a patient with ID 0137. Therefore, the attribute set in the trapdoor may leak user's privacy.

The second problem is that the current ABKS schemes cannot provide multi keyword search safely and efficiently. At present, some ABKS schemes can only provide single-keyword search so that the search results contain a large number of irrelevant results, which will greatly reduce the search efficiency. The ABKS schemes that can provide multi keyword search suffer from low efficiency due to impractical model or security risks caused by leaking part of the keyword information. Both of the drawbacks will hinder the practical application of ABKS.

To make it better applied to cloud storage, this paper proposes a **P**rivacy-Protecting **A**ttribute **B**ased Conjunctive **K**eyword **S**earch scheme (PABKS) to solve the two problems.

1.1 Related Work

The first Ciphertext-policy attribute-based encryption (CP-ABE) scheme was put forward by Bethencourt *et al.* in 2007 [6]. It is designed to provide fine-grained access control and it has gradually become a preferred choice of the access control mechanisms in the cloud storage due to its scalability and flexibility. Researchers have greatly strengthened the functionality of CP-ABE over the past decade [7-11].

To provide fine-grained access control and keyword search service simultaneously, Sun *et al.* [4] and Zheng *et al.* [5] combined CP-ABE with SE, and successively put forward ABKS scheme. In an ABKS scheme, the data owner constructs the index with the keywords extracted from the files and the access policy he/she determines. The user sends a trapdoor generated by his/her attributes and interested keywords to the server to make a query. The server runs search algorithm and will return the corresponding results to the user if the attributes in the trapdoor satisfy the access policy and the keywords in the trapdoor match those in the index. Therefore, data owners can authorize their search capabilities to users in a fine-grained method.

ABKS with hidden access policy

To protect the privacy included in the index, researchers have proposed several works to hide the access policy in the ABKS schemes. However, there is space for improvement in the expressiveness of access policy and efficiency for these schemes.

In 2017, Qiu *et al.* proposed an ABKS scheme with a hidden access policy [12]. The access policy is AND-gates, which is not expressive enough. There are n attributes in the system and each attributes have n_i values. Accordingly, the public key size is $\mathcal{O}(\sum_{i=1}^{n} n_i)$. To hide the access policy, the ciphertext includes a large number of redundant information, so that the ciphertext size is $\mathcal{O}(\sum_{i=1}^{n} n_i)$. Therefore, the storage efficiency of the scheme needs to be greatly improved.

In 2019, Miao *et al.* proposed a privacy-preserving ABKS system with hidden access policy in shared multi-owner setting [13]. The scheme hides the access policy in the same way as scheme [12], hence the scheme also has the problems of low storage efficiency and insufficient expressiveness of access policy in scheme [12].

Wang *et al.* utilized multilinear maps to put forward an ABE with fast keyword search construction [14]. The access policy of the construction is AND-gates, and the access policy is preserved efficiently. However, Hu *et al.* pointed out that GGH map which is the major candidate of multilinear map is not secure [15].

ABKS with hidden attribute set

In an ABE scheme, the problem of user's attribute set leakage is not considered (without outsourced decryption), because the user decrypts the ciphertext with his own secret key. However, in an ABKS scheme the attribute set is included in the trapdoor in plaintext, the problem is long-lasting in the literature but should be tackled in practice. To our best knowledge, none of the existing ABKS schemes can hide the attribute set in the trapdoor.

Retrieval Mode of ABKS

The retrieval modes of [4-5, 12] and [13] are single keyword search, that is, there is only one keyword contained in the trapdoor and index. It is worth mentioning that the search mode of [4] can be expanded to conjunctive keyword search by compressing the keywords in the index and trapdoor respectively. However, the requirement for successful search is that the keywords in the two are exactly the same, which limits the usability of the scheme.

Recently, Miao *et al.* proposed a series of ABKS schemes whose retrieval mode are conjunctive keyword search [16-18]. Different from [4], in these schemes the keyword matching occurs when the keywords in the index contain those in the trapdoor. In order to realize this retrieval mode, an ordered keyword set W is set in the public key of these schemes, the trapdoor contains the position information in W of each queried keyword. All users know W (otherwise they cannot execute the scheme), but the server is restricted to obtain W, this may narrow the application scenarios of these schemes.

In 2018, He *et al.* divided the keyword into keyword name and keyword value, and proposed an ABKS scheme that provides boolean keyword search [19] at the expense of exposing keyword names, which can reveal user's retrieval habits and bring security risks to the scheme.

Lately, Chen *et al.* used dual-server model to construct an ABKS scheme that can provide multi-keyword ranked search [20]. There are two servers in the scheme, and the search process is completed by the two server cooperatively and restrictively. However, the assumption of the applied model is that the two servers do not collude, which is a relatively strong assumption. The dual-server model leads to the relatively low search efficiency, and then affects its real-world applications.

Table 1. Feature comparison with existing works

Scheme	Retrieval mode	Access policy	Hidden P^1	Hidden S^2	Resist OKGA
[16-18]	Conjunctive keyword	Access tree	×	×	×
[19]	Boolean keyword	Access tree	×	×	×
[20]	Multi keyword ranked	Access tree	×	×	√
[12]	Single keyword	AND gates on multi-valued attributes	√	×	×
[13]	Single keyword	AND gates on multi-valued attributes	√	×	×
[14]	Single keyword	AND gates	√	×	×
PABKS	Conjunctive keyword	AND gates on multi-valued attributes with wildcards	√	√	√

^{1} P represents access policy. ^{2} S represents attribute set.

1.2 Related Work

This paper presents a privacy protecting attribute based conjunctive keyword search scheme. To our best knowledge, the PABKS is the first ABKS scheme that can hide access policy and the attribute set at the same time. Our scheme provides conjunctive keyword search under the premise of ensuring security and efficiency. The comparison of related schemes with the PABKS scheme is shown in Table 1. From Table 1 and the efficiency comparison between PABKS and the related schemes, it can be seen that we improve the security and practicability of ABKS, so that a more efficient and practical scheme may be applicable to cloud storage.

The main contributions of the paper are as follows:

Security: To improve the security of ABKS, the PABKS scheme is designed to hide the access policy and attribute set. We provide a formal security model and PABKS is proved to be IND-CKA, IND-OKGA, access policy hiding and attribute set hiding.

Conjunctive keyword search: On the premise of ensuring the security of the scheme, the PABKS scheme provides conjunctive keyword search.

Practicability: The access policy is and gates with multi-valued attributes, which is expressive. Besides, comprehensive simulations of the schemes are carried out using real-world dataset, and the simulation results show that the efficiency of the PABKS scheme is practical.

2 Preliminaries

2.1 Asymmetric Bilinear Maps

Let $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ be three multiplicative cyclic groups of prime order p, the generator of $\mathbb{G}_1$ is g_1 and the generator of $\mathbb{G}_2$ is g_2. Let $e:\mathbb{G}_1\times\mathbb{G}_2\rightarrow\mathbb{G}_T$ be an asymmetric bilinear map which satisfies the following three properties [21]:

- **Non-degeneracy**: $e(g_1,g_2)\neq 1$;
- **Bilinearity**: for all $e(\eta_1^{\delta_1},\eta_2^{\delta_2})=e(\eta_1,\eta_2)^{\delta_1\delta_2}$, then holds.
- **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.

2.2 The Generic Bilinear Group Model

The security of the PABKS scheme is proved in the generic bilinear group model [22]. ξ_1, ξ_2 and ξ_T are defined as three random encodings of an additive group $\mathbb{F}_p$, they are injective maps:ξ_1, ξ_2, $\xi_T:\mathbb{F}_p\rightarrow\{0,1\}^m$, $m>3log(p)$. Let $\mathbb{G}_i=\{\xi_i(x):x\in\mathbb{F}_p\}$, $i=1,2,T$. In the model, the adversary can access the oracles that simulate the hash function, group operations in $\mathbb{G}_i$, $i=1,2,T$, and the asymmetric bilinear map $e:\mathbb{G}_1\times\mathbb{G}_2\rightarrow\mathbb{G}_T$, the adversary can also interact with the challenger in the security game.

2.3 Access Policy

We define $\mathcal{U}=\{att_1,att_2,...,att_L\}$ as the attribute universe of the PABKS scheme. att_i consists of a unique attribute name and m attribute values $v_1,v_2,...,v_m$. (If the number of attribute value is less than m, the extra part can be set to empty.)

Assume that user's attribute set is $S=\{S_1',S_2',...,S_L'\}$, S_i' stands for the att_i with its selected value. Let $\mathcal{P}=\{S_1,S_2,...,S_L\}$ be the access policy which can be expressed as an AND gates on multi-valued attributes with wildcards. S_i stands for att_i with its selected attribute value or wildcard '*', and the relationship between S_i and S_j is "and". The wildcard '*' means "don't care", which means all attribute values of this attribute are acceptable. Let $f(S,\mathcal{P})=1$ denote attribute set S satisfies the access policy $\mathcal{P}$, $f(S,\mathcal{P})=0$ denote S does not satisfy $\mathcal{P}$.

3 System Architecture and Security Definition

3.1 Entities in System Architecture

As shown in Figure 2, there are four generic entities: Authority, Data Owner, User and Server involved in the architecture of the PABKS scheme. Detailed description is as follows.

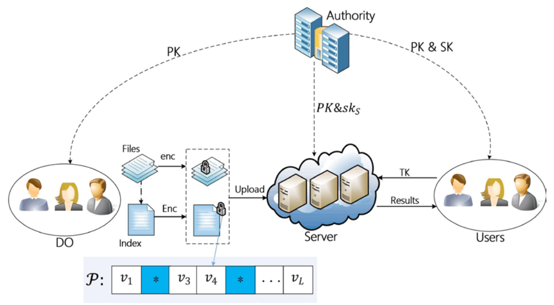


Figure 2. System architecture of PABKS

Authority. The authority is trusted in the system. It is responsible for authenticating the user and generating the public key, secret keys of users and server.

Data Owner (DO). The DO needs to store and share his/her data in the cloud. A DO encrypts the data and constructs the index, then uploads the index and encrypted files to the server.

User. To search the data in the server, the user needs to send his/her trapdoor to the server.

Server. The server stores a large amount of encrypted data and indexes uploaded by different DOs. It needs to complete the search process according to the trapdoor sent by the user.

3.2 Algorithms in PABKS Scheme

The PABKS scheme includes five algorithms: **Setup**, **KeyGen**, **IndGen**, **TKGen** and **Search**.

Setup(1^λ). Given the security parameter λ, the authority runs the algorithm and outputs the public key PK, master key MK and the secret key of the server sk_S.

KeyGen(MK, S). Given the MK and the attribute set S, the authority runs the algorithm and outputs the secret key of the user SK.

IndGen$(PK, \mathcal{P}, W)$. Given the PK, an access policy $\mathcal{P}$ and a keyword set W, the DO runs the algorithm and outputs the encrypted index $\mathcal{I}$.

TKGen(PK, W', SK). Given the PK, the keyword set W' and secret key SK, the user runs the algorithm and outputs the trapdoor TK.

Search$(\mathcal{I}, TK, sk_S)$. Given the index$\mathcal{I}$, trapdoor TK and the secret key of server sk_S, the server runs the algorithm and outputs 1 if S satisfies $\mathcal{P}$ and $W' \subseteq W$; otherwise it outputs a terminator $\perp$.

3.3 Adversarial Model

We provide the following assumptions in the model: the semi-trusted server tries to learn the valuable information while actually executing the protocol. Moreover, the authority and data owner are reliable. Therefore, two types of adversaries are considered:

Type-1 adversary $\mathcal{A}_1$ refers to outside user colluding with the server, it can get the secret key of the server and secret keys and trapdoors of corrupted users.

Type-2 adversary $\mathcal{A}_2$ means that the outside adversary. Note via corrupting specific users, we assume that $\mathcal{A}_2$ can obtain their secret keys and trapdoors.

3.4 Security Model

The PABKS scheme should meet the following security requirements:

Resist Chosen-keyword Attack [5]. It requires that without a matched trapdoor, $\mathcal{A}_1$ and $\mathcal{A}_2$ cannot judge which one of the two challenge keyword sets is in the challenge index.

Resist Outside Keyword Guessing Attack (OKGA) [23]. It requires that $\mathcal{A}_2$ cannot judge which one of the two challenge keyword sets is in the challenge trapdoor.

Access Policy Hiding. It requires that $\mathcal{A}_1$ and $\mathcal{A}_2$ cannot judge which one of the two challenge access policies is in the challenge index.

Attribute Set Hiding. It requires that $\mathcal{A}_2$ can not judge which one of the two challenge attribute sets is in the challenge trapdoor.

The IND-CKA security and access policy hiding for the PABKS scheme is defined via the following game.

CKA Security and $\mathcal{P}$ Hiding 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. C creates an empty table E, an empty set D, an empty keyword list L_{kw} and sets j=0. $\mathcal{A}_1$ can query the following oracles adaptively.

- $\mathcal{O}_{\mathbf{KeyGen}}(S)$: $\mathcal{C}$ executes **KeyGen**(MK, S), sets $D := D \cup \{S\}$ and returns SK to $\mathcal{A}_1$.
- $\mathcal{O}_{\mathbf{TKGen}}(S, W)$: $\mathcal{C}$ sets j=j+1, it executes **KeyGen**(MK, S) to generate SK, then it executes **TkGen**(PK, W, SK) to generate TK. It stores the entry (j,S,W) in table E and returns TK to $\mathcal{A}_1$.

Challenge. $\mathcal{A}_1$ sends two keywords sets W_0, W_1, and two access policy $\mathcal{P}_0$, $\mathcal{P}_1$ to $\mathcal{C}$. It requires

1. $\mathcal{P}_0$ and $\mathcal{P}_1$ have the same number of wildcards.
2. $\forall S \in D$, $f(S, \mathcal{P}_0) \neq 1$, $f(S, \mathcal{P}_1) \neq 1$.
3. for each entry in E, if $f(S, \mathcal{P}_i) = 1$, the corresponding W satisfies $W \not\subseteq W_i$.

$\mathcal{C}$ chooses $b \in_R \{0, 1\}$, executes **IndGen**$(PK, \mathcal{P}_b, W_b)$ to construct $\mathcal{I}^*$ and submits $\mathcal{I}^*$ to $\mathcal{A}_1$.

Phase 2. $\mathcal{A}_1$ continues to query the oracles as in Phase 1, the restriction of this stage is the same as the second and third stages of **Challenge** stage.

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

CKA Security and $\mathcal{P}$ Hiding Game for $\mathcal{A}_2$

Challenge, Phase 2 and **Guess** are the same as those in the last game for $\mathcal{A}_1$. The following definitions are for the Setup and Phase 1 in this game.

Setup. $\mathcal{C}$ executes **Setup** and sends 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.

$\mathcal{O}_{\mathbf{Search}}(I, TK)$: $\mathcal{C}$ executes $\mathbf{Search}$ and returns the result to $\mathcal{A}_2$.

The PABKS scheme is IND-CKA secure and access policy hiding if the advantage of $\mathcal{A}_i$ in the CKA security and access policy hiding game for $\mathcal{A}_i, i \in \{1, 2\}$ is:

$$|\Pr[Exp^{\mathrm{IND-CKA}}_{\mathcal{A}_i, \Pi_{PABKS}}(1^\lambda, U) = 1] - \tfrac{1}{2}| \leq \mathrm{negl}(\lambda)$$

IND-OKGA Security and $\mathcal{S}$ Hiding Game for $\mathcal{A}_2$

Setup. $\mathcal{C}$ executes $\mathbf{Setup}$ algorithm and returns $\mathcal{A}_2$ PK.

Phase 1. $\mathcal{A}_2$ can query the following oracles adaptively.

- $\mathcal{O}_{\mathbf{KeyGen}}(S)$: $\mathcal{C}$ executes $\mathbf{KeyGen}(MK, S)$ and returns SK to $\mathcal{A}_2$.
- $\mathcal{O}_{\mathbf{Trap}}(S, W)$: $\mathcal{C}$ executes $\mathbf{KeyGen}(MK, S)$ to generate SK, then it executes $\mathbf{TkGen}(PK, W, SK)$ and returns TK to $\mathcal{A}_2$.
- $\mathcal{O}_{\mathbf{Search}}(I, TK)$: $\mathcal{C}$ executes $\mathbf{Search}$ and returns the result to $\mathcal{A}_2$.

Challenge. $\mathcal{A}_2$ submits two attribute sets $\mathcal{S}_0$, $\mathcal{S}_1$, two keywords sets W_0, W_1 to $\mathcal{C}$. $\mathcal{C}$ chooses $b \in_R \{0, 1\}$, executes $\mathbf{KeyGen}(MK, \mathcal{S}_b)$ to generate SK, executes $\mathbf{TKGen}(PK, W_b, SK)$ and returns TK^* to $\mathcal{A}_2$.

Phase 2. $\mathcal{A}_2$ continues to query oracles as in **Phase 1**, the requirements of $\mathcal{O}_{\mathbf{Search}}$ are (assume the index $\mathcal{I}$ corresponds to $\mathcal{P}$ and W)

- $f(\mathcal{S}_i, \mathcal{P}) = 1$, then $W_i \nsubseteq W$.

Guess. $\mathcal{A}_2$ outputs a guess b' of b. The experiment returns 1 if and only if $b' = b$.

The PABKS scheme is IND-OKGA secure and attribute set hiding if the advantage of $\mathcal{A}_2$ in IND-OKGA security and attribute set hiding game for $\mathcal{A}_2$ is:

$$|\Pr[Exp^{\mathrm{IND-OKGA}}_{\mathcal{A}_2, \Pi_{PABKS}}(1^\lambda, U) = 1] - \tfrac{1}{2}| \leq \mathrm{negl}(\lambda)$$

4 The Proposed PABKS Scheme

Let N be the maximum of keywords of the files encrypted by the same access policy, and $H : \{0,1\}^* \to \mathbb{Z}_p$ be a secure hash function.

Setup (1^λ): The algorithm executes the group generator algorithm $\mathcal{G}(1^\lambda)$ and obtains asymmetric bilinear groups $(g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, p, e)$. It chooses $u_i \in_R \mathbb{Z}_p^*, i \in [1, L]$. Assume there are L attributes in the universe, the position of attribute i is u_i, each attribute has at most m_0 possible values, let them be $1, ..., m_0$. Moreover, wildcard (meaning 'don't care') is also considered in the access policy. Let N_i, $i \in [0, m_0]$ be $m_0 + 1$ upper bounds defined as:

$N_0 \leq L$: the maximum number of wildcard in an access policy;

$N_i \leq L$: the maximum number of attributes with i^{th} value in an attribute set $\mathcal{S}$.

The algorithm selects $\alpha, \beta, a, d, h, \{d_i\}_{i=0}^{N_0}, \{\delta_i\}_{i=1}^{m_0}$, $\{h_i\}_{i=0}^{N} \in_R Z_p^*$, and computes the public key $PK = (g_1, g_2, g_1^\alpha, \{g_1^{\delta_i}\}_{i=1}^{m_0}, g_2^\beta, g_2^d, g_2^{\frac{\alpha}{h}}, g_2^{\frac{a\beta}{h}}, \{g_2^{d_i}\}_{i=0}^{N_0}$, $\{u_i\}_{i=1}^{L}, \{g_1^{h_i}, g_2^{\frac{1}{h_i}}, g_1^{hh_i}, g_1^{\frac{h}{h_i}}\}_{i=0}^{N})$, the master key is $MK = (\alpha, \beta, a, d, \{d_i\}_{i=0}^{N_0})$, the secret key of server is $sk_S = a$.

IndGen$(PK, \mathcal{P}, W)$: Assume that $\mathcal{P}$ includes: $n_0 \leq N_0$ wildcards at positions $J = \{\pi_1, ..., \pi_{n_0}\}$, $n_i \leq N_i$ attributes with i^{th} value at positions $S_i = \{s_{i1}, ..., s_{in_i}\}$, $i \in [1, m_0]$. Consider $f(i) = \sum_{k=0}^{n_0} a_k i^k = \prod_{k=1}^{n_0} (i - \pi_k)$, the coefficients $a_i, i \in [0, n_0]$ can be efficiently computed using FFT, they are:

$$\begin{aligned}
a_{n_0} &= 1\\
a_{n_0-1} &= -(\pi_1 + \pi_2 + ... + \pi_{n_0})\\
a_{n_0-2} &= (\pi_1\pi_2 + \pi_1\pi_3 + ... + \pi_{n_0-1}\pi_{n_0})\\
&......\\
a_0 &= -(\pi_1 \cdot \pi_2 \cdots \pi_{n_0})
\end{aligned}$$

Then the algorithm computes $\Pi_i = \sum_{k \in S_i} \prod_{\pi_j \in J} (k - \pi_j)$, $i \in [1, m_0]$ and construct vector $\overrightarrow{v} = (v_0, v_1, ..., v_{n_0})$ as: $\overrightarrow{v} = (a_0, a_1, ..., a_{n_0})$.

It selects $c, f \in_R \mathbb{Z}_p$, computes $s = f + \sum_{i=0}^{n_0} v_i, C = g_2^{s/c}$, $C' = \prod_{i=1}^{m_0} g_1^{\delta_i \Pi_i / c}, C'' = g_2^{df/c}, , \{C_j = g_2^{d_j \cdot v_j / c}\}_{j=0}^{n_0}$

Assume $W = \{w_1, ..., w_n\}, n \leqslant N$, it chooses $\{R_i\}_{i=1}^{N-n} \in_R Z_p^*$ and sets:

$$(\bar{b}_1, \bar{b}_2, ..., \bar{b}_N) = (H(w_1), ..., H(w_n), R_1, ..., R_{N-n}).$$

It sets $f'(i) = \sum_{k=0}^{N} a'_k i^k = \prod_{k=1}^{N} (i - \bar{b}_k)$ and uses fast Fourier transform (FFT) [24] algorithm to compute coefficients of $f(i)$. The values of a'_i are:

$$\begin{aligned}
a'_N &= 1\\
a'_{N-1} &= -(\bar{b}_1 + \bar{b}_2 + ... + \bar{b}_N)\\
a'_{N-2} &= (\bar{b}_1\bar{b}_2 + \bar{b}_1\bar{b}_3 + ... + \bar{b}_{N-1}\bar{b}_N)\\
&......\\
a'_0 &= (-1)^N(\bar{b}_1 \cdot \bar{b}_2 \cdots \bar{b}_N)
\end{aligned}$$

It constructs vector $\overrightarrow{v}' = (v'_0, v'_1, ..., v'_N)$ as:$v'_i = a'_i / c$, $\{I_i = g_2^{\frac{\beta s}{c} + \frac{v'_i}{h_i}}\}_{i \in [0,N]}, I' = \prod_{i=0}^{N} g_1^{\frac{h}{h_i} v'_i} = g_1^{h \sum_{i=0}^{N} \frac{v'_i}{h_i}}$, $I'' = g_2^{\frac{a\beta}{h} \cdot \frac{s}{c}}$the index is set as:

$$\mathcal{I} = (C, C', C'', \{C_j\}_{j \in [0, n_0]}, I', I'', \{I_i\}_{i \in [0,N]}).$$

KeyGen$(MK, \mathcal{S})$: Assume user's attribute set includes $n'_i \leq N_i$ attributes with i^{th} value whose positions are $S'_i = \{s'_{i1}, ..., s'_{in'_i}\}$, $i \in [1, m_0]$, $j \in [0, N_0]$. For $i \in [1, m_0]$, it sets

$$x_{ij} = \begin{cases} -\delta_i \sum_{k \in S'_i} k^j, & S'_i \neq \phi \\ 0, & S'_i = \phi \end{cases}$$

Let $\overrightarrow{x_i} = (x_{i0}, ..., x_{iN_0})$, $i \in [1, m_0]$. It chooses $t, b \in_R \mathbb{Z}_p$, and computes $\overrightarrow{x} = (x_0, ..., x_{N_0}) = \sum_{i=1}^{m} \overrightarrow{x_i}$, the secret key is computed as:

$$SK = (K = g_1^{\alpha\beta+t}, K' = g_2^b, K'' = g_1^{t/d}, \{K_j = g_1^{\frac{bx_j+t}{d_j}}\}_{j=0}^{N_0})$$

TKGen(PK, SK, W'): Let $W' = \{w_1', w_2', ..., w_m'\}$ be the keyword set, it chooses $u, \{f_j\}_{j=1}^{m} \in_R Z_p$ and computes vector $\overrightarrow{x}' = (x_0', x_1', ..., x_N')$ as follows:

$x_0' = f_1 \cdot H(w_1')^0 + f_2 \cdot H(w_2')^0 + ... + f_m H(w_m')^0$
$x_1' = f_1 \cdot H(w_1')^1 + f_2 \cdot H(w_2')^1 + ... + f_m H(w_m')^1$
......
$x_N' = f_1 \cdot H(w_1')^N + f_2 \cdot H(w_2')^N + ... + f_m H(w_m')^N$

It then computes $tk = K^u, tk' = K'^u, tk'' = K''^u$, $\{tk_j = K_j^u\}_{j=0}^{N_0}, \{T_j = g_1^{\alpha u + h_j x_j'}\}_{j\in[0,N]}$, $T' = g_2^{\frac{\alpha}{h}u}$, $T'' = \prod_{i=0}^{N} g_1^{hh_i x_i'} = g_1^{h\sum_{i=0}^{N} h_i x_i'}$. The trapdoor is $TK = (tk, tk', tk'', \{tk_j\}_{j\in[0,N_0]}, T', T'', \{T_j\}_{j=0}^{N})$.

Search$(TK, \mathcal{I}, sk_S)$: The algorithm can be divided into two parts. Firstly, it computes

$$\Psi = \frac{e(tk, C)}{e(C', tk')e(tk'', C'')\prod_{i=0}^{n_0} e(tk_i, C_i)} = e(g_1, g_2)^{\alpha\beta su/c}$$

Secondly, it computes

$$\Psi' = [\frac{\prod_{i=0}^{N} e(T_i, I_i)}{e(I', T')e(T'', I''^{1/a})}]^{\frac{1}{N+1}} = e(g_1, g_2)^{\alpha\beta su/c}$$

and checks if $\Psi = \Psi'$, if so, the algorithm returns 1; otherwise it returns 0.

5 Security Analysis

This section formally proves the security of the PABKS scheme with the following theorems.

Theorem 1: $\xi_i, \mathbb{G}_i, (i = 1, 2, T)$ and e are defined in section 2.2. Assume that q is the upper limit of the sum of all the queries of $\mathcal{A}_1$, then the advantage of $\mathcal{A}_1$ in the CKA security and $\mathcal{P}$ hiding game for $\mathcal{A}_1$ is $\mathcal{O}(q^2/p)$.

To prove that the PABKS scheme is access policy hiding, we only need to prove that the adversary cannot distinguish between $\mathcal{P}_0$ and $\mathcal{P}_1$. Here the proof chooses a random access policy $\mathcal{P}_r$, the number of wildcards in $\mathcal{P}_0$, $\mathcal{P}_1$ and $\mathcal{P}_r$ are the same. To prove that the PABKS scheme is IND-CKA secure, we need to prove that the adversary cannot distinguish between W_0 and W_1. Here the proof chooses a set of random strings $W_r, |W_r| \le N$. Then the proof considers the game sequences from $\mathbf{Game}_0$ to $\mathbf{Game}_4$. They are described in a high-level way as follows:

$\mathbf{Game}_0$: The challenge index $\mathcal{I}_0^*$ is computed under $\mathcal{P}_0$ and W_0, specifically, $\mathcal{I}_0^* = \mathbf{IndGen}(PK, \mathcal{P}_0, W_0)$.

$\mathbf{Game}_1$: The challenge index $\mathcal{I}_1^*$ is computed under $\mathcal{P}_0$ and W_r, specifically, $\mathcal{I}_1^* = \mathbf{IndGen}(PK, \mathcal{P}_0, W_r)$.

$\mathbf{Game}_2$: The challenge index $\mathcal{I}_2^*$ is generated under $\mathcal{P}_r$ and W_r, specifically, $\mathcal{I}_2^* = \mathbf{IndGen}(PK, \mathcal{P}_r, W_r)$.

$\mathbf{Game}_3$: The challenge index $\mathcal{I}_3^*$ is generated under $\mathcal{P}_1$ and W_r, specifically, $\mathcal{I}_3^* = \mathbf{IndGen}(PK, \mathcal{P}_1, W_r)$.

$\mathbf{Game}_4$: The challenge index $\mathcal{I}_4^*$ is generated under $\mathcal{P}_1$ and W_1, specifically, $\mathcal{I}_4^* = \mathbf{IndGen}(PK, \mathcal{P}_1, W_1)$.

Clearly, proving Theorem 1 is equivalent to proving that the advantage of $\mathcal{A}_1$ in distinguishing $\mathbf{Game}_0$ and $\mathbf{Game}_4$ is bounded by $\mathcal{O}(q^2/p)$. The proof has four steps.

- **Indistinguishability between $\mathbf{Game}_0$ and $\mathbf{Game}_1$**

We need to prove that the advantage of $\mathcal{A}_1$ in distinguishing between $\mathcal{I}_0^*$ and $\mathcal{I}_1^*$ is negligible. Here we consider a modified game, the challenge keyword set is W_0 and I_ξ is either $g_2^{\frac{\beta s}{c}+\frac{a_\xi'}{ch_\xi}}$ or $g_2^{\frac{\beta s}{c}+\frac{\theta}{h_\xi}}$ (i.e. $v_\xi' = \frac{a_\xi}{c}$ is replaced by θ), $\xi \in [0, N]$. It is not difficult to draw a conclusion that if $\mathcal{A}_1$ has an advantage ϵ in distinguishing $\mathbf{Game}_0$ and $\mathbf{Game}_1$, then $\mathcal{A}_1$ has advantage of at least $\epsilon/2$ in the modified game.

Setup. $\mathcal{B}$ selects $\alpha, \beta, a, d, h, \{d_i\}_{i=0}^{N_0}, \{\delta_i\}_{i=1}^{m_0}, \{h_i\}_{i=0}^{N}$, $\{u_i\}_{i=1}^{L} \in_R \mathbb{F}_p$ computes $(g_1^\alpha, \{g_1^{\delta_i}\}_{i=1}^{m_0}, g_2^\beta, g_2^d, g_2^{\frac{\alpha}{h}}, g_2^{\frac{a\beta}{h}}$, $\{g_2^{d_i}\}_{i=0}^{N_0}, \{g_1^{h_i}, g_2^{\frac{1}{h_i}}, g_1^{hh_i}, g_1^{\frac{h}{h_i}}\}_{i=0}^{N})$ and sends the public key and $sk_S = a$ to $\mathcal{A}_1$.

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

$\mathcal{O}_{\mathbf{KeyGen}}(S_k)$: $\mathcal{B}$ constructs the vector $\overrightarrow{x}^{(k)}$ corresponding to S_k, then chooses $t^{(k)}, b^{(k)} \in_R \mathbb{F}_p$ and computes $SK = (K = g_1^{\alpha\beta+t^{(k)}}, K' = g_2^{b^{(k)}}, K'' = g_1^{t^{(k)}/d}$, $\{\mathrm{K}_j = g_1^{\frac{b^{(k)}x_j^{(k)}+t^{(k)}}{d_j}}\}_{j=0}^{N_0})$, it makes $D = D \cup \{S_k\}$ and sends SK to $\mathcal{A}_1$.

$\mathcal{O}_{\mathbf{Trap}}(S_k, W')$: Assume $W' = \{w_1', ..., w_m'\}$, $\mathcal{B}$ sets j=j+1 and accesses $\mathcal{O}_{\mathbf{KeyGen}}$ to obtain SK, then chooses $u^{(k)}, \{f_j^{(k)}\}_{j=1}^{m} \in_R \mathbb{F}_p$ and computes $\overrightarrow{x}'^{(k)}$. It stores (j,S,W') in E generate the trapdoor and sends to $\mathcal{A}_1$.

Challenge. $\mathcal{A}_1$ submits keyword set $W_0, \xi \in [0, N]$ and an access policy $\mathcal{P}_0$ to $\mathcal{B}$. It requires that

- $\forall S \in D, f(S, \mathcal{P}_0) \neq 1$.
- for each entry in E, if $f(S, \mathcal{P}_0) = 1$, it is required that $W \not\subseteq W_0$ be satisfied for the corresponding W.

$\mathcal{B}$ computes vector $\overrightarrow{v}$ corresponding to $\mathcal{P}_0$ and computes vector $\overrightarrow{v}'$ corresponding to W_0 except that $v_\xi' = \theta$, selects $c, f \in_R \mathbb{F}_p$, let $s = f + \sum_{i=0}^{n_0} v_i$, and genreates corresponding challenge index, then $\mathcal{B}$ returns the challenge index to $\mathcal{A}_1$.

Phase 2. $\mathcal{A}_1$ continues to query oracles as in **Phase 1**. The restriction in this stage is if $f(S, \mathcal{P}_0) = 1$,

- $\mathcal{A}_1$ cannot access $\mathcal{O}_{\mathbf{KeyGen}}$.
- (S,W) cannot be the input to $\mathcal{O}_{\mathbf{Trap}}$ if $W \subseteq W_0$.

Next we analyze the probability of unexpected collisions. Since every oracle query can be treated as a rational function $\vartheta = \kappa/\psi$, where κ and ψ are selected from related variables. The collision occurs when two queries corresponding to two different rational functions are mapped to the same output. Referring to [6] and [25], the probability of happening such events is at most $\mathcal{O}(q^2/p)$. Table 2 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 prove through the following two lemmas that $\mathcal{A}_1$ can tell the difference between $g_2^{\frac{\beta s}{c}+\frac{a'_\xi}{ch_\xi}}$ or $g_2^{\frac{\beta s}{c}+\frac{\theta}{h_\xi}}$ with a negligible advantage.

Table 2. possible $\mathbb{G}_1$ and $\mathbb{G}_2$ queries for $\mathcal{A}_1$

	$\mathbb{G}_1$					$\mathbb{G}_2$					
PK	α	δ_i	h_i	hh_i	$\frac{h}{h_i}$	β		$\frac{\alpha}{h}$	$\frac{\alpha\beta}{h}$	d_i	$\frac{1}{h_i}$
Index	$\sum_{i=1}^{m_0}\frac{\delta_i\prod_i}{c}$		$h\sum_{i=0}^{N}\frac{v'_i}{h_i}$			s/c	df/c	$\frac{d_jv_j}{c}$	$\frac{\alpha\beta s}{hc}$	$\frac{\beta s}{c}+\frac{v'_i}{h_i}$	
SK	$\alpha\beta+t^{(k)}$		$\frac{bx_j^{(k)}+t^{(k)}}{d_j}$		$\frac{t^{(k)}}{d}$	$b^{(t)}$					
TK	$(\alpha\beta+t^{(k)})u^{(k)}$		$\alpha u^{(k)}+h_jx_j'^{(k)}$		$\frac{tu^{(k)}}{d}$	$bu^{(k)}$			$\frac{\alpha u^{(k)}}{h}$		
	$h\sum_{i=0}^{N}h_ix_i'^{(k)}$		$\frac{bx_j^{(k)}+t^{(k)}}{d_j}u^{(k)}$								

Lemma 1: $\mathcal{A}_1$ cannot construct a TK with W and S, s.t. $f(S, \mathcal{P}_0) = 1$ and $W \subseteq W_0$.

This lemma can be proved from two aspects.

(1) Given TK whose S satisfies $f(S, \mathcal{P}) = 1$, take the k^{th} query as an example, we prove $\mathcal{A}_1$ can not change $T_j = g_1^{\alpha u^{(k)}+h_jx_j'^{(k)}}$ in TK to $g_1^{\alpha u^{(k)}+h_jx''_j}$, $x''_j \in_R \mathbb{F}_p$ are chosen by $\mathcal{A}_1$. From Table 2, it is easy to find that $\mathcal{A}_1$ cannot construct the term $\alpha u^{(k)} + h_jx''_j$ in $\mathbb{G}_1$.

(2) We prove that $\mathcal{A}_1$ cannot construct SK such that $f(S, \mathcal{P}_0) = 1$, here we prove an equivalent conclusion that $\mathcal{A}_1$ cannot construct $e(g_1, g_2)^{\gamma\alpha\beta s/c}, \gamma \in \mathbb{F}_p$.

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 2 there is only one *possible* type of output that contains $\gamma\alpha\beta s/c$ in $\mathbb{G}_T$.

It is $(\alpha\beta + t^{(k)}) \cdot s/c = \alpha\beta s/c + t^{(k)}s/c$, then $\mathcal{A}_1$ needs to cancel out the term $t^{(k)}s/c$in $\mathbb{G}_T$. Since $s = f + \sum_{i=0}^{n_0} v_i$, combined with Table 3 we analyze that $\mathcal{A}_1$ can get the term $t^{(k)}s/c$ in $\mathbb{G}_T$ only by doing as the following formula

$$(\sum_{i=1}^{m}\frac{\delta_i\Pi_i}{c})\cdot b + \frac{t^{(k)}}{d}\cdot\frac{df}{c} + \prod_{i=0}^{n_0}\frac{bx_i^{(k)}+t^{(k)}}{d_i}\cdot\frac{d_iv_i}{c}$$
$$= \frac{t^{(k)}s}{c} + \underbrace{\frac{b}{c}[(\overrightarrow{x}^{(k)}, \overrightarrow{v}) + \sum_{i=0}^{m}\delta_i\Pi_i]}_{A}$$

However, this can not be achieved because in the security game, it requires $\forall S \in D, f(S, \mathcal{P}_0) \neq 1$, hence the term A is not 0. Therefore we can conclude that $\mathcal{A}_1$ can not construct $e(g_1, g_2)^{\gamma\alpha\beta s/c}$ and *lemma 1* is proved.

Lemma 2: $\mathcal{A}_1$ can construct $e(g_1, g_2)^{\delta(\frac{\beta s}{c}+\frac{a'_\xi}{ch_\xi})}$ for some g_1^δ that can be composed from the oracles outputs with a negligible probability.

The goal is to query $\Delta = \delta(\frac{\beta s}{c} + \frac{a'_\xi}{ch_\xi})$ in $\mathbb{G}_T$ for some δ in $\mathbb{G}_1$.

For the term $\delta\frac{v'_\xi}{h_\xi}$, since the only method to construct v'_ξ is the linear combination of $h_i \cdot (\frac{\beta s}{c} + \frac{v'_i}{h_i})$, $(i \neq \xi)$. That is, $v'_\xi + \frac{\beta s}{c}\sum_{i \in T}\tau_ih_i, \tau_i \in \mathbb{F}_p$, where T is a set of integers and does not contain ξ. Hence, $\delta = h_\xi$ and $\delta\frac{v'_\xi}{h_\xi} = v'_\xi$. However, from Table 2 we can find that the term $\delta\frac{\beta s}{c} = h_\xi\frac{\beta s}{c}$ in $\mathbb{G}_T$ can not be further constructed.

Therefore, *lemma 2* is proved and $\mathcal{A}_1$ can tell the difference between $g_2^{\frac{\beta s}{c}+\frac{v'_\xi}{h_\xi}}$ and $g_2^{\frac{\beta s}{c}+\frac{\theta}{h_\xi}}$ with a negligible advantage. Further, the advantage of $\mathcal{A}_1$ in distinguishing between $\mathbf{Game_0}$ and $\mathbf{Game_1}$ is bounded by $\mathcal{O}(q^2/p)$.

- **Indistinguishable between $\mathbf{Game}_1$ and $\mathbf{Game}_2$**

Similar with the proof of the indistinguishability between $\mathbf{Game}_0$ and $\mathbf{Game}_1$, the proof considers a modified game, the challenge policy is $\mathcal{P}_0$ and C_ξ is either $g^{d_\xi v_\xi/c}$ or $g^{d_\xi\theta/c}$ (i.e. $v_\xi$is replaced by θ), $\xi \in [0, N_0]$. It is not difficult to draw a conclusion that if $\mathcal{A}_1$ has an advantage ϵ in distinguishing $\mathbf{Game}_1$ and $\mathbf{Game}_2$, then $\mathcal{A}_1$ has advantage at least $\epsilon/2$ in the modified game.

The **Setup**, **Phase 1** and **Phase 2** are the same as those in the proof of section 5.1 except for the restriction on the $\mathcal{O}_{\mathbf{Trap}}$ queries.

Challenge. $\mathcal{A}_1$ submits keyword set W_r and an access policy $\mathcal{P}_0$ to $\mathcal{B}$. It requires that

- $\forall S \in D, f(S, \mathcal{P}_0) \neq 1$.

$\mathcal{B}$ computes vector $\overrightarrow{v}$ corresponding to $\mathcal{P}_0$ except that v_ξ is set as θ, then selects $c, f \in_R \mathbb{F}_p$, and computes $s = f + \sum_{i=0}^{n_0} v_i, C = g_2^{s/c}, C' = \prod_{i=1}^{m_0} g_1^{\delta_i\Pi_i/c}, C'' = g_2^{df/c}$, $\{C_j = g_2^{d_j \cdot v_j/c}\}_{j=0}^{N_0}$, the components $\{I_i\}_{i\in[0,N]}, I', I''$ are randomly selected.

As discussed in section 5.1, the probability of unexpected collusion is $\mathcal{O}(q^2/p)$. The only method for $\mathcal{A}_1$ to tell the difference between $g_2^{d_\xi v_\xi/c}$ and $g_2^{d_\xi\theta/c}$ is to construct

$e(g_1, g_2)^{\delta d_\xi v_\xi / c}$ for some g_1^δ that can be composed from the oracles outputs. Next we prove that $\mathcal{A}_1$ can never construct $\Delta = \delta d_\xi v_\xi / c$ in $\mathbb{G}_T$ for some δ in $\mathbb{G}_1$.

Consider the variable v_ξ, the only method to construct v_ξ is the linear combination of $\frac{bx_j^{(k)} + t^{(k)}}{d_j} \cdot \frac{d_j v_j}{c}, (i \neq \xi)$. That is, $\frac{bv_\xi}{c} + \sum_{i \in T} \tau_i \frac{t^{(k)} v_j}{c}$, where $\tau_i \in \mathbb{F}_p$, T is a set of integers and does not contain ξ. Hence, $\delta = b/d_\xi$, however, we find that Table 2 does not contain the term b/d_ξ in $\mathbb{G}_1$.

Therefore, $\mathcal{A}_1$ can tell the difference between $g_2^{d_\xi v_\xi / c}$ and $g_2^{d_\xi \theta / c}$ with a negligible advantage. Further, the advantage of $\mathcal{A}_1$ in distinguishing between $\mathbf{Game}_1$ and $\mathbf{Game}_2$ is bounded by $\mathcal{O}(q^2/p)$.

The remaining proofs are similar to the above proofs:

- The indistinguishability between $\mathbf{Game}_2$ and $\mathbf{Game}_3$ can be proved in the same way as for $\mathbf{Game}_1$ and $\mathbf{Game}_2$.
- The indistinguishability between $\mathbf{Game}_3$ and $\mathbf{Game}_4$ can be proved in the same way as for $\mathbf{Game}_0$ and $\mathbf{Game}_1$.

Theorem 2: Let $\xi_i, \mathbb{G}_i, (i = 1, 2, T)$ and e be defined in **Theorem 1**, then the advantage of $\mathcal{A}_2$ in the IND-KGA security and $\mathcal{S}$ hiding game for $\mathcal{A}_2$ is $\mathcal{O}(q^2/p)$.

The proof is similar with that This proof is similar to that of Theorem 1. It is omitted here due to space limitation.

6. Performance Analysis

This session analyzes and compares the efficiency of our scheme with related schemes from both theoretical and experimental levels.

6.1 Theoretical Analysis

The efficiency of [12] and the PABKS scheme in storage cost and computation cost are respectively shown in Table 4 and Table 5. And the symbols involved in the comparison are defined in Table 3.

Table 3. Notations 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 an index
m	Number of keywords in an trapdoor
N_0	Maximum number of wildcard in P.
n_0	Number of wildcards in P.
m_0	Maximum number of values of an attribute
N	Maximum number of keywords in an index
L_*	Length of element in *
$\overline{n}$	The number of attribute in the system in scheme [12]

As shown in Table 4, the two schemes are compared in SK size, TK size and index size. They can reflect the storage cost from different perspectives. The SK and TK sizes of the PABKS scheme are and , respectively; while those of scheme [12] are both . Because is smaller than and is smaller than , our storage cost outperforms that of [12], in which the PABKS scheme provides conjunctive keyword search, but the scheme [12] only offers single keyword search. The index size of the PABKS scheme is $2L_{\mathbb{G}_1} + (N + n_0 + 5)L_{\mathbb{G}_2}$, which is much smaller than that of the scheme [12]. The reason is the index used in [12] contains a large number of redundant components which are used to hide the access policy. With the increase of attribute number, the advantage of the PABKS scheme may become more and more prominent.

Table 5 compares the computational cost of four algorithms ($\mathbf{KeyGen}$, $\mathbf{TKGen}$, $\mathbf{IndGen}$ and $\mathbf{Search}$) in the two schemes. The PABKS scheme costs $(N_0 + 3)\mathbb{G}_1 + \mathbb{G}_2$ and $(2N + N_0 + 5)\mathbb{G}_1 + 2\mathbb{G}_2$ in $\mathbf{KeyGen}$ and $\mathbf{TKGen}$ respectively. Because the operations in $\mathbb{G}_1$ is more efficient than in $\mathbb{G}_2$ and N_0 is smaller than $\overline{n}$ in most cases, the PABKS scheme has advantage in the computation. In terms of the computational cost of index generation and search, our scheme needs $(N + m_0 + 1)\mathbb{G}_1 + (N + n_0 + 6)\mathbb{G}_2$ and $(N + n_0 + 7)P + \mathbb{G}_T + \mathbb{G}_2$ respectively, which are roughly equivalent to those of scheme [12]. In addition, the smaller the N is, the less the computational cost our scheme provides (as compared to the scheme [12]).

Table 4. Storage cost theoretical comparison

Scheme	SK size	TK size	Index size
[12]	$(2\overline{n} + 1)L_{\mathbb{G}_2}$	$(2\overline{n} + 1)L_{\mathbb{G}_2}$	$(\overline{n}m_0 + \overline{n} + 1)L_{\mathbb{G}_1} + L_{\mathbb{G}_T}$
PABKS	$(N_0 + 3)L_{\mathbb{G}_1} + L_{\mathbb{G}_2}$	$(N + N_0 + 5)L_{\mathbb{G}_1} + 2L_{\mathbb{G}_2}$	$2L_{\mathbb{G}_1} + (N + n_0 + 5)L_{\mathbb{G}_2}$

Table 5. Computation cost theoretical comparison

Scheme	KeyGen	IndGen	TKGen	Search
[12]	$(2\overline{n} + 2)\mathbb{G}_2$	$(\overline{n} + 1)\mathbb{G}_1 + \overline{n}\mathbb{G}_2 + \mathbb{G}_T$	$(2\overline{n} + 1)\mathbb{G}_2$	$(2\overline{n} + 1)P + 2\mathbb{G}_T$
PABKS	$(N_0 + 3)\mathbb{G}_1 + \mathbb{G}_2$	$(N + m_0 + 1)\mathbb{G}_1 + (N + n_0 + 6)\mathbb{G}_2$	$(2N + N_0 + 5)\mathbb{G}_1 + 2\mathbb{G}_2$	$(N + n_0 + 7)P + \mathbb{G}_T + \mathbb{G}_2$

6.2 Experimental Analysis

To evaluate the actual performance of [12] and the PABKS scheme, we carry out the simulations of the two schemes. The simulations are performed on Charm and the real world Request for Comments Database (RFC) is used as the test dataset. Charm is a framework developed based on Python, which is helpful to the rapid prototyping of cryptographic mechanisms and protocols [26]. We load the asymmetric bilinear groups MNT224 in PBC library to realize the asymmetric bilinear mapping. We select 50 files from the database and extract 100 keywords as the keyword dictionary. To simplify the comparison, we set$\overline{n} = N_0 = \{4, 6, 8, 10, 12, 14, 16, 18, 20\}, m_0 = 4$ n_0 be half of N_0, i.e. $n_0 = \{2, 3, 4, 5, 6, 7, 8, 9, 10\}$. To highlight the difference from the single keyword search in scheme [12], we set N to 1 and 25 respectively. The tests are performed on a computer running an Intel(R) Core(TM) i5-

4690 CUP at 3.50 GHz and 8GB RAM. All the simulation results are averaged over 50 times measures.

The unit of storage cost and computation cost are kilobyte and second respectively.

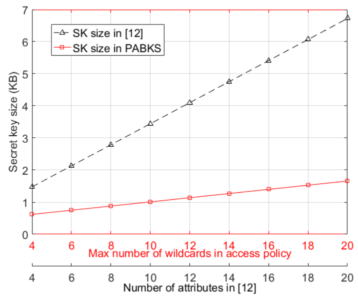
The three subgraphs in Figure 3. respectively shows the secret key size, trapdoor size and index size of scheme [12] and the PABKS scheme. The abscissa of each subgraph has two variables with different colors: $\overline{n}$, the number of attributes in [12], and n_0 or N_0, the number/max number of wildcard in access policy. We notice that the SK size, TK size and index size all increase linearly with the spike of the abscissa respectively, and we find that the slope of the black lines in the three subgraphs are all greater. As shown in Figure 3(a), the SK size in [12] is always larger than that of the PABKS scheme and the gap becomes more and more obvious with the increase of the abscissa. The case of $N = 1$ in Figure 4(b) is similar to that in Figure 3(a). As for $N = 25$, the TK size of the PABKS scheme requires more than that of scheme [12] before $N_0 = \overline{n} = 8$; after that, the situation reverses. We can also conclude that the increase of the N leads to a small increase of the TK size of the PABKS scheme. In Figure 3(c), we can observe that the black line is sandwiched between the two red lines. When $N = 1$, the index size of the PABKS scheme is always less than that of scheme [12], the reason is that the index in scheme [12] is full of a large number of redundant information to protect the access policy. By comparison, the PABKS scheme is more efficient in protecting access policy. When $N = 25$, its index size is large than that of scheme [12], which is mainly because the $L_{\mathbb{G}_2}$ is three times as long as $L_{\mathbb{G}_1}$.

The four subgraphs in Figure. 4 respectively present the time cost of four algorithms: $\mathbf{KeyGen}$, $\mathbf{TKGen}$, $\mathbf{EncIndex}$, $\mathbf{Search}$ of scheme [12] and the PABKS scheme. The analysis of Figure 4(a) and Figure 4(b) is similar to that of Figure 3(a) and Figure 3(b) respectively. Beyond protecting the attribute set in trapdoor, our scheme still has a great advantage in the computation cost of generating trapdoor and secret key as compared to the scheme proposed in [12]. As shown in Figure 4(c), the time cost of $\mathbf{EncIndex}$ of the two schemes is roughly at the same level when single-keyword is considered, and the PABKS scheme starts to consume less time to generate the index than scheme [12] when $n_0 = 5, \overline{n} = 10$. When $N = 25$, the former takes a little more time than the latter. The analysis of Figure 4(c) is applicable to that of Figure 4(d), and even in the case of $N = 25$, the cost of search process is acceptable. Therefore, the simulation results are consistent with the theoretical analysis.

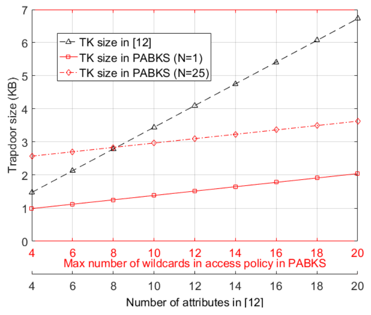
7 Conclusion

We target at improving the security and practicability of ABKS schemes, making it more suitable for cloud storage. We connect the access control process and keyword matching process between the inner product encryption to put forward the PABKS scheme. For the first time, the PABKS scheme can hide the access policy and attribute set, which can better protect the privacy of DO and users. We define a formal security model and the PABKS scheme is proved to be IND-CKA, IND-OKGA, access hiding and attribute set hiding. The scheme provides conjunctive keyword search on the premise of ensuring the security of the scheme, and the access policy is AND-gates on multi-valued attributes with wildcards. The simulations on actual dataset show that the efficiency of PABKS is practical for real-world applications.

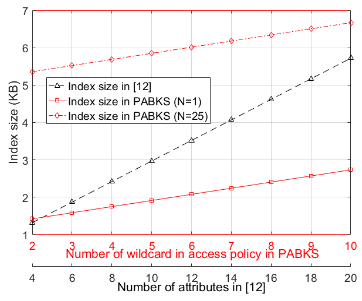
The shortcomings of the scheme are the public key of the proposed scheme is relatively long and the access policy is relatively inflexible, we will improve them in the future work.



(a) Secret key size

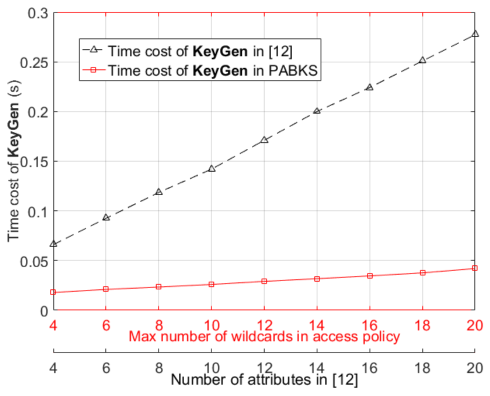


(b) Trapdoor size

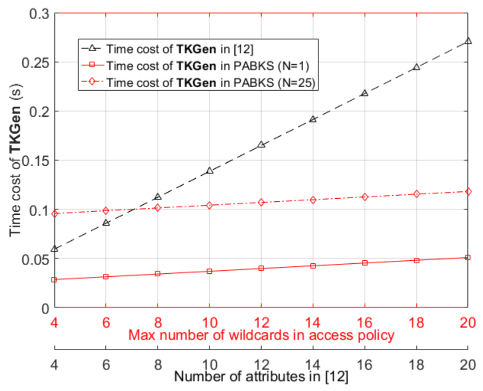


(c) Index size

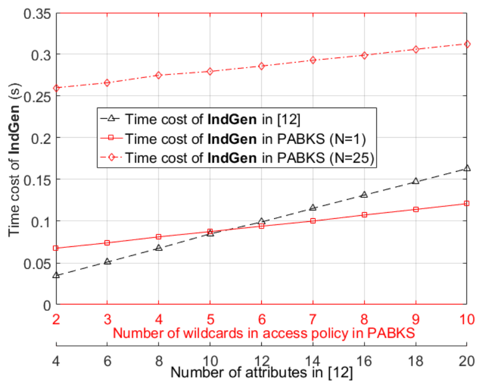
Figure 3. Storage cost theoretical comparison



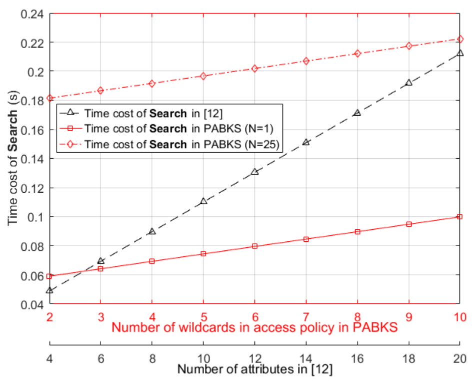
(a) Time cost of **KeyGen**



(b) Time cost of **TKGen**



(c) Time cost of **IndGen**



(d) Time cost of **Search**

Figure 4. Computation cost theoretical comparison

References

[1] A. Oussous, F. Z. Benjelloun, A. A. Lahcen, S. Belfkih, Big Data technologies: A survey, *Journal of King Saud University-Computer and Information Sciences*, Vol. 30, No. 4, pp. 431-448, October, 2018.

[2] D. Boneh, G. D. Crescenzo, R. Ostrovsky, G. Persiano, Public key encryption with keyword search, *International conference on the theory and applications of cryptographic techniques*, Interlaken, Switzerland, 2004, pp. 506-522.

[3] R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky, Searchable symmetric encryption: improved definitions and efficient constructions, *Journal of Computer Security*, Vol. 19, No. 5, pp. 895-934, November, 2011.

[4] W. Sun, S. Yu, W. Lou, Y. T. Hou, H. Li, Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud, *IEEE Conference on Computer Communications*, Toronto, Canada, 2014, pp. 226-234.

[5] Q. Zheng, S. Xu, G. Ateniese, VABKS: verifiable attribute-based keyword search over outsourced encrypted data, *IEEE Conference on Computer Communications*, Toronto, Canada, 2014, pp. 522-530.

[6] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, *IEEE symposium on security and privacy*, Berkeley, CA, USA, 2007, pp. 321-334.

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

[8] Y. Chen, W. Li, F. Gao, W. Yin, K. Liang, H. Zhang, Q. Wen, Efficient attribute-based data sharing scheme with hidden access structures, *The Computer Journal*, Vol. 62, No. 12, pp. 1748-1760, December, 2019.

[9] J. Ning, X. Dong, Z. Cao, L. Wei, X. Lin, White-box traceable ciphertext-policy attribute-based encryption supporting flexible attributes, *IEEE Transactions on Information Forensics and Security*, Vol. 10, No. 6, pp. 1274-1288, June, 2015.

[10] L. Xue, Y. Yu, Y. Li, M. H. Au, X. Du, B. Yang, Efficient attribute-based encryption with attribute revocation for assured data deletion, *Information Sciences*, Vol. 479, pp. 640-650, April, 2019.

[11] M. Xie, Y. Ruan, H. Hong, J. Shao, A CP-ABE scheme based on multi-authority in hybrid clouds for mobile devices, *Future Generation Computer Systems*, Vol. 121, pp. 114-122, August, 2021.

[12] S. Qiu, J. Liu, Y. Shi, R. Zhang, Hidden policy ciphertext-policy attribute-based encryption with keyword search against keyword guessing attack, *Science China Information Sciences*, Vol. 60, No. 5, pp. 1-12, May, 2017.

[13] Y. Miao, X. Liu, K. K. R. Choo, R. H. Deng, J. Li, H. Li, J. Ma, Privacy-preserving attribute-based keyword search in shared multi-owner setting, *IEEE Transactions on Dependable and Secure Computing*, Vol. 18, No. 3, pp. 1080-1094, May-June, 2021.

[14] H. Wang, X. Dong, Z. Cao, Multi-value-independent

ciphertext-policy attribute based encryption with fast keyword search, *IEEE Transactions on Services Computing*, Vol. 13, No. 6, pp. 1142-1151, November-December, 2020.

[15] Y. Hu, H. Jia, Cryptanalysis of GGH map, *International Conference on the Theory and Applications of Cryptographic Techniques*, Vienna, Austria, 2016, pp. 537-565.

[16] Y. Miao, J. Ma, X. Liu, X. Li, Z. Liu, H. Li, Practical attribute-based multi-keyword search scheme in mobile crowdsourcing, *IEEE Internet of Things Journal*, Vol. 5, No. 4, pp. 3008-3018, August, 2018.

[17] Y. Miao, J. Ma, X. Liu, J. Weng, H. Li, H. Li, Lightweight fine-grained search over encrypted data in fog computing, *IEEE Transactions on Services Computing*, Vol. 12, No. 5, pp. 772-785, September-October, 2019.

[18] Y. Miao, J. Ma, X. Liu, X. Li, Q. Jiang, J. Zhang, Attribute-based keyword search over hierarchical data in cloud computing, *IEEE Transactions on Services Computing*, Vol. 13, No. 6, pp. 985-998, November-December, 2020.

[19] K. He, J. Guo, J. Weng, J. Weng, J. K. Liu, X. Yi, Attribute-based hybrid Boolean keyword search over outsourced encrypted data, *IEEE Transactions on Dependable and Secure Computing*, Vol. 17, No. 6, pp. 1207-1217, November-December, 2020.

[20] Y. Chen, W. Li, F. Gao, Q. Wen, H. Zhang, H. Wang, Practical attribute-based multi-keyword ranked search scheme in cloud computing, *IEEE Transactions on Services Computing*, Vol. 15, No. 2, pp. 724-735, March-April, 2022.

[21] R. Ostrovsky, A. Sahai, B. Waters, Attribute-based encryption with non-monotonic access structures, *Proceedings of the 14th ACM conference on Computer and communications security*, Alexandria, Virginia, USA, 2007, pp. 195-203.

[22] V. Shoup, Lower bounds for discrete logarithms and related problems, *International Conference on the Theory and Applications of Cryptographic Techniques*, Konstanz, Germany, 1997, pp. 256-266.

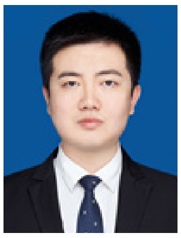
[23] J. W. Byun, H. S. Rhee, H. A. Park, D. H. Lee, Off-line keyword guessing attacks on recent keyword search schemes over encrypted data, *Workshop on secure data management*, Seoul, Korea, 2006, pp. 75-83.

[24] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to algorithms*, MIT press, 2009.

[25] J. T. Schwartz, Fast probabilistic algorithms for verification of polynomial identities, *Journal of the ACM (JACM)*, Vol. 27, No. 4, pp. 701-717, October, 1980.

[26] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, A. D. Rubin, Charm: a framework for rapidly prototyping cryptosystems, *Journal of Cryptographic Engineering*, Vol. 3, No. 2, pp. 111-128, June, 2013.

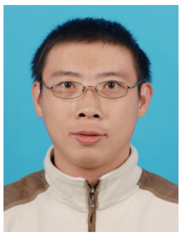
Biographies



Yang Chen received the Ph.D. degree in computer science and technology from the Beijing University of Posts and Telecommunications (BUPT) in 2020. He is currently an engineer in the National Computer Network Emergency Response Technical Team/Coordination Center of China (CNCERT/CC). His research interests include cryptography, cloud computing and information security.

Yang Liu received M.D. degree from Beijing University of Posts and Telecommunications (BUPT) in 2009. She is currently a senior engineer in National Computer Network Emergency Response Technical Team/ Coordination Center of China (CNCERT/CC). Her research interests include network security and information security.

Jin Pan received M.D. degree from Beijing University of Posts and Telecommunications (BUPT) in 2011. He is currently an engineer in National Computer Network Emergency Response Technical Team/ Coordination Center of China (CNCERT/CC). His research interests include network security and blockchain.



Fei Gao received the PhD degree in cryptography from the Beijing University of Posts and Telecommunications (BUPT) in 2007. Currently, he is a professor of BUPT. His research interests include cryptography, information security, and quantum algorithm. He is a member of the Chinese Association for Cryptologic Research.

Emmanouil Panaousis is Professor of Cyber Security at the University of Greenwich, UK. His research expertise is at the area of cyber risk management. His research is funded by the European Commission, the Engineering and Physical Sciences Research Council, and National Cyber Security Centre.