Journal of Systems Architecture 128 (2022) 102533

Contents lists available at ScienceDirect

Journal of Systems Architecture

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

An efficient and access policy-hiding keyword search and data sharing scheme in cloud-assisted IoT☆

Hui Yin [a], Yangfan Li [b,*], Fangmin Li [a], Hua Deng [a], Wei Zhang [a], Keqin Li [c]

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

ARTICLE INFO

Keywords:
Attribute-based encryption
Attribute-based keyword search
Access policy hiding
Cloud computing
Data sharing
Internet of Things (IoT)

ABSTRACT

By leveraging the cloud computing paradigm, the cloud-assisted Internet of Things (IoT) is able to offer the massive storage and highly available computation services for end-users. When the data collected from IoT devices is endlessly gathering to the cloud center, the data security become new challenges. Encrypting data is an effective measure to guarantee data confidentiality. However, traditional encryption techniques make the data searching and access control inoperative. Current attribute-based keyword search (ABKS) techniques provide a feasibility to achieve data searching and access control over encrypted data simultaneously. However, existing ABKS schemes leak sensitive information via clear access policy and may be unsuitable for certain IoT applications, where the access policies in data might contain private information of data owners such as the security number or the residential area of a resident in smart grids. In this work, we design an efficient and policy-hiding attribute-based keyword search and data sharing scheme (PH-ABKS-DS) in cloud-assisted IoT. To the best of our knowledge, this construction is the first PH-ABKS-DS with practical efficiency that is constructed on prime order group. We provide detailed correctness and security analyses for PH-ABKS-DS. Extensive experiments demonstrate that our proposed PH-ABKS-DS is correct and practical.

1. Introduction

Nowadays, Internet of Things (IoT) has become the crucial infrastructure to build smart society and economy. By connecting hundreds of millions of real-world physical objects, including sensors, smart phones, actuators, and any objects that can be connected, IoT achieves information communication and transmission anytime and anywhere. It is the broad connectivity, the ubiquitous accessibility, and the dynamic information processing that IoT is leading to a smart, inclusive and sustainable economy and society.

Currently, IoT is not just an academic pursuit and kinds of IoT applications have been widely deployed in real world such as agriculture irrigation, smart health, industrial control, intelligent transportation [1]. However, with the operations of IoT systems, how to store and deal with the high-volume data collected from the resource-constrained IoT devices every day is an inevitable problem. These data as the valuable wealth to achieve social intellectualization needs to be periodically stored and analyzed over a more powerful platform. In the cloud computing era, a nature solution is in the IoT system to introduce the cloud paradigm with the *inexhaustible* storage and computation capacities. As a result, the cloud-assisted IoT has become a widely adopted and popular IoT architecture [2–5].

While the cloud-assisted IoT is able to offer massive storage and highly-available computation services for end-users, the adoption of cloud computing architecture leads to the data centralization to the cloud center, which incurs new challenges for data security [6]. The security issue of the outsourced data is always a disgusting headache, since once data is uploaded to the cloud platform, the data owners will no more possess the physical control on their private data. It is for this reason that the data owner may be reluctant to store their data to the cloud server in spite of the enormous advantages of the cloud computing model. Encrypting data before storing them to cloud is an effective way to keep data security against malicious attackers and cloud servers [7]. However, traditional block-cipher techniques make the data searching and data access control on ciphertext ineffective. In many potential applications, data searching is an indispensable function to allow data users to quickly locate targets from large-scale cloud data, and an access control measure can prevent the outsourced data from being unauthorized accessed. To enable fine-grained search

☆ Hui Yin and Yangfan Li contributed equally to this paper.

* Corresponding author.
E-mail addresses: yangfanli@hnu.edu.cn (Y. Li), lifangmin@whut.edu.cn (F. Li).

https://doi.org/10.1016/j.sysarc.2022.102533
Received 18 November 2021; Received in revised form 12 March 2022; Accepted 24 April 2022
Available online 10 May 2022

permission control over encrypted cloud data, inspired by attribute-based encryption (ABE) primitive [8] and searchable encryption (SE) primitive [9], researchers proposed the attribute-based keyword search (ABKS) techniques [10–15]. By associating an access policy with an encrypted searchable index, ABKS schemes can guarantee that the search results can be correctly returned to a data user if and only if the data user's attribute set satisfies the access policy and the submitted search query matches the secure index. However, almost all existing ABKS schemes expose the attribute values in the access policy associated with the encrypted index keyword. Those solutions may be unsuitable for the IoT applications, where the access policies might contain private information that data owners and data users are reluctant to disclose. For example, in smart grids, there is an access policy ((“SSN:*223-35-6678*” **AND** “Status: Normal”) **OR** (“Name: Alice” **AND** “Affiliation: Grid Company” **AND** “City: San Francisco” **AND** “Position: Senior Manager”)) that is used to encrypt a customer's readings from smart meters. The access policy indicates that the encrypted readings can only be accessed by the customer with social security number (SSN) *223-35-6678* and *Normal* registration status or by a *senior manager* named *Alice* of *grid company* in *San Francisco*. However, with the clear attribute values in the access policy, the cloud can at least infer two information: (1) Alice is a senior manager of the grid company in San Francisco and (2) the data owner with SSN *223-35-6678* may live in San Francisco. Another example is that if a user's encrypted s-health records contain a clear access policy (“Department: Cardiologist” **AND** “Affiliation: City Hospital)”, the cloud server can infer that the user is suffering a heart illness [16]. These sensitive information in access policies certainly leak the users' privacy. Therefore, developing an efficient, expressive, and access policy-hiding attribute-based keyword search and data sharing (PH-ABKS-DS) scheme in Cloud-assisted IoT applications is of paramount importance.

In [17], Lai et al. proposed an expressive and policy-hiding attribute-based encryption scheme by only exposing the attribute names while hiding the attribute values (referred to as partially hidden access structure). Zhang et al. [16] improved their scheme for more efficient decryption test and practical large universe construction. Very recently, based on Lai et al.'s approach, several expressive and access policy-hiding attribute-based encryption schemes have been proposed for different application scenarios [18–21]. In these schemes, for the access policy mentioned above, the partially hidden versions can be written as ((“SSN:***” **AND** “Status: ***”) **OR** (“Name: ***” **AND** “Affiliation: ***” **AND** “City: ***” **AND** “Position: ***”)). Thus, the sensitive attribute values are hidden. Theoretically, their constructions can be used to develop a policy-hiding ABKS scheme. However, such a scheme is inappropriate for a real search system due to being constructed over the composite order group, which will lead to an impractical search efficiency, especially in the large-scale data set. In this paper, we are enlightened from the idea in those two schemes and leverage an expressive and efficient attribute-based scheme proposed in [22] to construct an efficient and policy-hiding attribute-based keyword search and data sharing scheme (PH-ABKS-DS) for cloud-assisted IoT systems. To the best of our knowledge, this construction is the first PH-ABKS-DS with practical efficiency that is constructed on prime order group. We provide detailed security analysis for our scheme. Extensive experiments also demonstrate that our proposed scheme is correct and practical.

2. Related work

2.1. Searchable encryption

Searchable encryption is an attractive cryptology primitive that allows a server to perform the encrypted keyword based data searching over ciphertext without leaking to the server underlying contents about the encrypted search query and the ciphertext. Depending on the encryption settings constructing SE schemes, searchable encryption is classified to two categories: searchable symmetric encryption (SSE) and public key encryption with keyword search (PEKS). The first practical SSE was proposed in 2000 by Song et al. [23] that requires a linear scan of all data for searching. To improve the search efficiency, Curtomla et al. [24] designed a sub-linear SEE scheme by using an encrypted inverted index structure. In that paper, the authors first employ leakage functions to evaluate the security of a SSE scheme, which have become a formal security definition in the follow-up study. However, that scheme is a static SSE without taking secure data addition and deletion into consideration. To make SSE schemes more practical in real application, SSE schemes had been further developed to the dynamic environment, allowing for secure data addition and deletion with low communication and computation cost [25–28]. Aiming at the security for the dynamic SSE, Zhang et al. developed file injection attacks and proposed that a dynamic SSE scheme needs to achieve forward privacy [29–31], which guarantees that updating a data does not leak more information than what a predefined leakage function leaks. In addition, to prevent the server from conducting dishonest search, verifiable searchable encryption construction [32] is also hot research topic. Those schemes allow the data user to verify the completeness and correctness of the query results [33].

Compared with SSE, while the public key based searchable encryption schemes [34–39] have more expensive search complexity, they can obtain stronger security and implement powerful functionalities, such as conjunctive, subset, and range queries.

Recently, with the wide deployment of IoT applications, researchers start to investigate frameworks, schemes, and systems of searchable encryption in the cloud-assisted IoT application environments [3,4,40–42]. Though these works do not take the data access control into consideration, they significantly promote the further development of searchable encryption in the emerging application fields.

2.2. Attribute-based encryption and attribute-based keyword search

Attribute-based encryption is developed from the identity-based encryption and enables flexible and fine-grained decryption control on ciphertext. The first attribute-based encryption scheme was proposed by Sahai et al. in [8]. In this scheme, if and only if two attribute sets in a private key and a ciphertext are within a certain distance of each other as judged by some metric, the private key is able to decrypt ciphertext. Thus, the decryption control is achieved. To enhance the expressivity of access policy, the access structure is introduced into the ABE scheme. Current ABE schemes are generally divided two categories: ciphertext-policy ABE (CP-ABE) [22,43–45] and key-policy ABE (KP-ABE) [46–48]. In CP-ABE, an access policy is associated with the ciphertext and a set of attributes is embedded into a private key. The position of the access policy and the attribute set is just the opposite in KP-ABE. In both CP-ABE and KP-ABE, a decryption can be conducted if and only if the attribute set has to satisfy the access policy. Due to the flexibility and fine-grainedness of data access control, ABE is very suitable to be applied in the cloud computing data outsourcing systems for secure data sharing, and several related works were present in [49,50]. In these schemes above, the attributes in access policy are often disclosed to the public, which may reveal privacy information. To prevent the sensitive information from being leaked from the access policy, in [17], Lai et al. proposed an expressive and policy-hiding attribute-based encryption scheme by only exposing the attribute names while hiding the attribute values (referred to as partially hidden access structure). Zhang et al. [16] improved their scheme for more efficient decryption test and practical large universe construction. Very recently, based on Lai et al.'s approach, several expressive and access policy-hiding attribute-based encryption schemes have been proposed for different application scenarios [18–21]. Due to being constructed over the composite order group, these schemes need to pay for a relatively heavy computation cost. Though policy-hiding attribute-based encryption schemes based on the prime order group were proposed in [51–53], they only supports

Table 1
Function comparison between our scheme and related works.

Schemes	Data sharing	Keyword search	Policy hiding	Expressiveness	Group order	Large universe
[16,18–21]	√	×	√	LSSS	Composite	√
[17]	√	×	√	LSSS	Composite	×
[51–53]	√	×	√	AND-gates	Prime	×
[10–15]	√	√	×	Access Tree	Prime	√
PH-ABKS-DS	√	√	√	LSSS	Prime	√

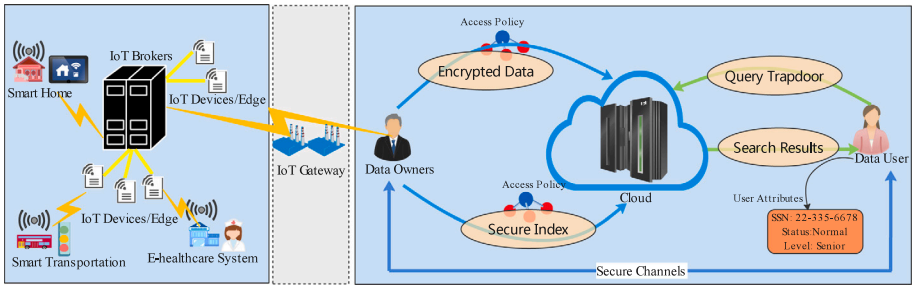


Fig. 1. The system model of PH-ABKS-DS in cloud-assisted IoT.

AND-gates based access policy, which limits the expressivity and the flexibility.

Inspired by searchable encryption primitive and attribute-based encryption primitive, researchers proposed the attribute-based keyword search (ABKS) techniques [10–15]. ABKS can achieve data searching and fine-grained access control over encrypted data simultaneously. However, all existing ABKS schemes expose the attributes in the access policy associated with the secure searchable index, which usually contain user's sensitive information or even directly reflect the contents of the data. Theoretically, the constructions in [16–21] can be used to develop an expressive and policy-hiding attribute-based keyword search scheme. However, Such schemes are based on the composite order bilinear group and will incur an impractical search complexity. This motivates us to explore a PH-ABKS-DS scheme based on the prime order bilinear group with more practical search complexity. We provide a function comparison between our proposed PH-ABKS-DS construction in Section 5 and some related works, as shown in Table 1.

3. Preliminaries

In this section, we briefly review several key instruments used to construct our PH-ABKS-DS scheme.

3.1. Bilinear pairing map

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic multiplicative groups of prime order q and g be a generator of group $\mathbb{G}_1$. Given a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, if e is efficiently computable and satisfies the following two properties (1) $\forall a, b \in G$ and $x, y \in Z_p$, $e(a^x, b^y) = e(a, b)^{xy}$ and (2) $e(g, g) \neq 1$, then we say e is a bilinear pairing map over groups $\mathbb{G}_1$ and $\mathbb{G}_2$.

3.2. Access structure

Let $\{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An access structure is a collection $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, \ldots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.

In ABE, a set of attributes play the role of the parties and an access policy organized by a collection of sets of attributes is equivalent to the access structure.

3.3. Linear secret sharing scheme

A secret sharing scheme (LSSS) π over a set of parties $\mathcal{P}$ is called linear if (1) the shares for each party form a vector over $\mathbb{Z}_p$ and (2) there exists a matrix A with l rows and n columns called the share generation for π. For all $i = 1, \ldots, l$, the ith row of A is labeled by a party $\rho(i)$, where ρ is a function from $\{1, \ldots, l\}$ to $\mathcal{P}$. We generate a random vector $v = (s, r_2, \ldots, r_n)$, where s denotes the secret to be shared and $s, r_2, \ldots, r_n \in \mathbb{Z}_p$, and compute Av that is the vector of l shares of the secret s according to π. The share $(Av)_i$ belongs to party $\rho(i)$.

In ABE, An LSSS π has been widely used to express an access structure $\mathbb{A}$. Suppose that $S \in \mathbb{A}$ is any authorized set and we define the subset $I = \{i : \rho(i) \in S\} \subseteq \{1, 2, \ldots, l\}$ to be a minimum authorized set satisfying (A, ρ) (In ABE, S denotes a set of attributes corresponding to the rows of LSSS matrix). For any I, we can find constants $\{w_i \in \mathbb{Z}_p\}_{i \in I}$ such that $\sum_{i \in I} w_i \lambda_i = s$ holds, for any valid shares $\{\lambda_i\}$ of a secret s according to π. For any $I' \subset I$, no such constants $\{w_i\}$ exist (i.e., any subset of I does not satisfy (A, ρ)). Given (A, ρ), we first find out all minimum authorized sets $I_1, I_2, \ldots$, and further write notation $\Phi_{(A,\rho)}$ to denote the set $\{I_1, I_2, \ldots\}$.

4. Problem formulations

In this section, we describe the system model and present the formal security definition for our proposed PH-ABKS-DS.

4.1. System model

The system model of our proposed PH-ABKS-DS scheme in cloud-assisted IoT involves five types of entities, i.e., a number of IoT brokers, IoT gateways, data owners (individual users or organizations), cloud server, and data users, as shown in Fig. 1. IoT brokers collect data from IoT applications and via IoT gateways send them to the data owner. To guarantee the confidentiality and searchability of the outsourced data, the data owner encrypts data using traditional symmetric encryption, as well as builds encrypted and attribute-based searchable index using our proposed scheme. In particular, according to the access permission of a data file and the search permission of an index keyword, the different access policy is associated with a symmetric key encrypting the data file, and the index keyword, respectively. In our system, the attribute information in the access policy is partially hidden, which means that the non-sensitive attribute names are exposed and the sensitive attribute values are hidden. When a data user wants to request data from the cloud server, he uses his private key to encrypt the interested

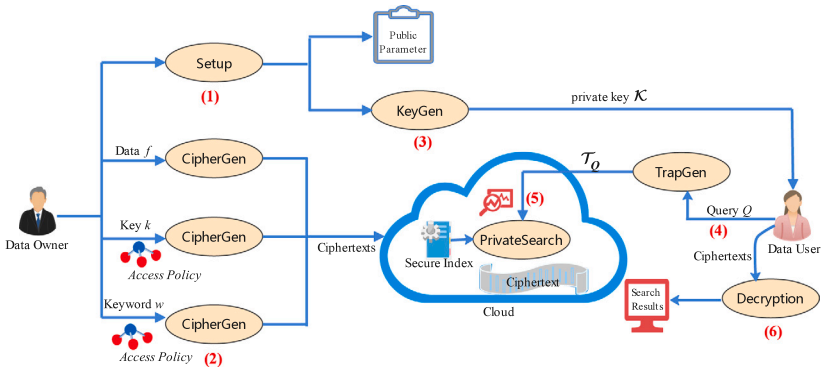


Fig. 2. The overview of PH-ABKS-DS.

keyword and submits the encrypted keyword (query trapdoor) to the cloud server. Upon receiving the query trapdoor, the cloud server conducts the attribute-based keyword search over encrypted index. On one hand, for each search result (ciphertext) returned from the cloud server, if and only if the data user has the access permission to the underlying data, he can recover the corresponding symmetric key to decrypt the ciphertext. On the other hand, obviously, if the data user has not the search permission of the index keyword, nothing will be returned. That is, if a data user wishes obtain some data files containing a certain keyword, then his attribute set has to satisfy the access policies in the keyword and the data files simultaneously.

4.2. Security definition

Formally, the security requirements of a PH-ABKS-DS scheme is that the cloud server is prohibited from inferring any useful information from the secure index, query trapdoor, and encrypted data files. Obviously, the confidentiality of data files can be guaranteed by the well-known symmetric encryption. Therefore, we focus on the security of index keywords and query keywords. In order to provide a formal security proof, we give the security definition based on the following adaptively chosen-keyword search attack game between a polynomial time adversary $\mathcal{A}$ and a challenger $\mathcal{B}$.

Adaptively Chosen-Keyword Search Attack Game (ACKSA):

Setup. $\mathcal{B}$ runs an algorithm **Setup** to set up a system public parameter **P** and a master key **MK**. $\mathcal{B}$ sends **P** to $\mathcal{A}$ and keeps **MK** secret.

Phase 1. In this phase, $\mathcal{A}$ requests query trapdoors $T_{Q_1}, \ldots, T_{Q_n}$ of keywords $Q_1, \ldots, Q_n$ by adaptively querying oracles $\mathcal{O}_{\mathbf{KenGen}}$ and $\mathcal{O}_{\mathbf{TrapGen}}$, where $Q_i (1 \le i \le n)$ corresponds to a set S_i of attributes.

- $\mathcal{O}_{\mathbf{KenGen}}(S_i)$: $\mathcal{B}$ runs an algorithm `KeyGen` to generate the private key $\mathcal{K}_i (1 \le i \le n)$.
- $\mathcal{O}_{\mathbf{TrapGen}}(\mathcal{K}_i, Q_i)$: $\mathcal{B}$ runs an algorithm `TrapGen` to generate the query trapdoor $\mathcal{T}_{Q_i} (1 \le i \le n)$ and sends it to $\mathcal{A}$.

Challenge. $\mathcal{A}$ picks up a challenging LSSS policy $\mathcal{M}^*$ and two keywords w_0 and w_1, which are sent to $\mathcal{B}$. The restriction is that if $S_i (1 \le i \le n)$ satisfies $\mathcal{M}^*$, then $w_0, w_1 \notin \{Q_1, \ldots, Q_n\}$. $\mathcal{B}$ randomly chooses a bit $b \in \{0, 1\}$ and encrypts w_b with $\mathcal{M}^*$ as I_{w_b}, which is given to $\mathcal{A}$.

Phase 2. $\mathcal{A}$ repeats **Phase 1.** There is a restriction that if an attribute set S_x corresponding to a requested trapdoor $\mathcal{T}_{Q_x}$ satisfies $\mathcal{M}^*$, then $w_0, w_1 \ne Q_x$.

Guess. $\mathcal{A}$ receives the ciphertext I_{w_b} and requests the search output between I_{w_b} and $\mathcal{T}_{Q_i} (1 \le i \le n)$ by adaptively querying oracle $\mathcal{O}_{\mathbf{PrivateSearch}}$.

- $\mathcal{O}_{\mathbf{PrivateSearch}}(I_{w_b}, \mathcal{T}_{Q_i})$: $\mathcal{A}$ runs an algorithm `PrivateSearch`, which outputs 1 if and only if S_i satisfies (A, ρ) and w_b is equal to Q_i; otherwise outputs 0.

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

We define the advantage that $\mathcal{A}$ wins *ACKSA* game to be $Adv = \mathbf{Pr}[b = b'] - \frac{1}{2}$.

5. Construction of PH-ABKS-DS scheme

Our PH-ABKS-DS scheme is composed of `Setup`, `KeyGen`, `CipherGen`, `TrapGen`, `PrivteSearch`, and `Decryption` six polynomial time algorithms. As shown in Fig. 2, we describe an overview of the PH-ABKS-DS scheme, whose work flow involves the following steps.

(1) The data owner first initializes the run environment to generate a system public parameter **P** and a master key **MK**.
(2) The data owner generates ciphertexts for outsourced index keywords and data files. A data file f is encrypted under an assigned symmetric key k. To achieve secure and fine-grained authorization search as well as data sharing, each index keyword w and symmetric key k are encrypted under a policy-hiding access structure, respectively, according to w's search permission and f's access permission.
(3) When a data user u with attribute set S joins in the system, the data owner for him generates a private key $\mathcal{K}$, where S is planted into $\mathcal{K}$.
(4) When the data user u wishes to search data files containing keyword Q, u encrypts Q to generate Q's trapdoor $\mathcal{T}_Q$ using his private key $\mathcal{K}$.
(5) The cloud server performs private authorization search between secure index and $\mathcal{T}_Q$. If there exists an encrypted index keyword that is equal to Q and u has the search permission to the index keyword, the cloud server will return a set $\mathcal{R}$ of ciphertexts containing Q.
(6) For each ciphertext in $\mathcal{R}$, if u has the access permission to its plaintext, u can recover the symmetric key generating the ciphertext to obtain the goal data; otherwise, the ciphertext is discarded.

Next, we describe the implementation of each algorithm to construct our proposed PH-ABKS-DS scheme.

5.1. Run environment initialization

The `Setup` algorithm takes a security parameter κ as input and outputs the system public parameter **P** and the master key **MK**. **P** is opened and **MK** is kept secret by the data owner. The detail of this algorithm is present in Algorithm 1.

Algorithm 1 Setup

Input:
Security parameter κ.
Output:
System public parameter **P**, master key **MK**.
1: Choose two multiplicative cycle groups $\mathbb{G}_1$ and $\mathbb{G}_2$ of prime order q and define a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$. Let g be a generator of $\mathbb{G}_1$ and χ be the maximum number of system attributes.
2: Construct two Cryptographic hash functions: $H_1 : \{0,1\}^* \to \mathbb{G}_1$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_q^*$.
3: Choose two random exponents from $\mathbb{Z}_q^*$: α and β.
4: Calculate g^α, $e(g,g)^\alpha$, and g^β.
5: $\mathbf{P} = (\mathbb{G}_1, \mathbb{G}_2, e, g, q, \chi, H_1, H_2, e(g,g)^\alpha, g^\alpha, g^\beta)$.
6: $\mathbf{MK} = \alpha$.
7: **return** **P**, **MK**.

Algorithm 2 KeyGen

Input:
System Public parameter **P**, master key **MK**, and a data user u's attribute set S.
Output:
u's private keys $\mathcal{K}$.
1: Parse S as (Π_S, S), where $\Pi_S \subseteq \{1,2,...,\chi\}$ and $S = \{s_i\}_{i \in \Pi_S}$.
2: Choose a random exponent t from $\mathbb{Z}_q^*$.
3: Compute $K_1 = g^\alpha$, $K_2 = g^{\beta t}$, and $K_2 = g^t$.
4: **for** each attribute $s \in S$ **do**
5: Compute $K_s = H_1(s)^t$.
6: **end for**
7: **return**

$$\mathcal{K} = (K_1 = g^\alpha, K_2 = g^{\beta t}, K_3 = g^t, \forall s \in S : K_s = H_1(s)^t).$$

5.2. Key generation

The KenGen algorithm takes **P**, **MK**, and a data user u's attribute set S as input, outputs the private key for the data user. The detail of this algorithm is present in Algorithm 2. Actually, in the PH-ABKS scheme, the data owner runs algorithm KenGen to grants the keyword based search permission and data access right to the data user. With the private key $\mathcal{K}$, the data user can generate the legal query trapdoor in terms of his interested query keyword and then decrypt the symmetric key used to encrypt query results. We assume that, in our PH-ABKS-DS system, there exist secure communication channels to transfer information between the data owner and the data user.

5.3. Index keyword and data encryption

CipherGen algorithm is the ciphertext generation algorithm in PH-ABKS-DS scheme, which consists of three sub-algorithms. The detail of this algorithm is present in Algorithm 3. Given a collection $F = \{f_1, \ldots, f_n\}$ of files and a collection of index keywords $W = \{w_1, \ldots, w_m\}$ extracted from F, by repeatedly running Algorithm 3, F and W are encrypted as $\widehat{F} = \{(\widehat{\mathcal{G}_1}, \widehat{f_1}), \ldots, (\widehat{\mathcal{G}_n}, \widehat{f_n})\}$ and $\widehat{W} = \{\mathcal{I}_{w_1}, \ldots, \mathcal{I}_{w_m}\}$, respectively, where $\mathcal{G}_i$ is the symmetric key seed used to encrypt data f_i. Each $\widehat{\mathcal{G}_i}$ and $\mathcal{I}_{w_j}$ are associated with a special access policy specifying the decryption permission of f_i and the search permission of w_j. Different from the original ABE scheme in [22] and existing ABKS schemes, in our PH-ABKS-DS, the attribute values in the access policy are hidden for preventing the sensitive information leakage from the access policy. Nevertheless, our scheme can carry out correct private search and decryption in a policy-hiding manner. We do this by introducing for the attribute value corresponding to the ith row of A a random $h_{\rho(i)}$ and binding it with the corresponding inner product $\lambda_i = v \cdot A_i$ in ciphertext components C and D (See line 5 in Algorithm 3.1 and line 17 in Algorithm 3.3). When search or decryption, the random $h_{\rho(i)}$ can be canceled out but obtain the required value λ_i. After encryption, the ciphertexts $\widehat{W}$ and $\widehat{F}$ are stored at cloud server, respectively, and each $\mathcal{I}_{w_i}$ points to the encrypted files containing the keyword w_i.

Algorithm 3 CipherGen

Input:
System public parameter **P**, a symmetric key seed $\mathcal{G}$, a data file f, and an index keyword w.
Output:
$\mathcal{G}$'s ciphertext $\widehat{\mathcal{G}}$, f's ciphertext $\widehat{f}$, and w's ciphertext $\mathcal{I}_w$.

Algorithm 3.1 IndexWordEnc
1: Define an LSSS policy $\mathbb{L}_w = (A, \rho, \mathcal{H})$, where A is an $l \times n$ matrix and a group of random values $\mathcal{H} = (h_{\rho(1)}, ..., h_{\rho(l)}) \in \mathbb{Z}_q^l$.
2: Choose a random value $s_w \in \mathbb{Z}_q^*$ and generate a random vector $v_w = (s_w, x_2, ..., x_n)$.
3: Choose a group of random values $\gamma_1, ..., \gamma_l$ from $\mathbb{Z}_q^*$.
4: **for** i=0 to l **do**
5: Compute:
(1) $\lambda_i = v_w \cdot A_i$, where A_i denotes the ith row of A.
(2) $C_{\Delta_w,i} = g^{\beta\lambda_i} H_1(\rho(i))^{-\gamma_i h_{\rho(i)}}$.
(3) $D_{\Delta_w,i} = g^{\gamma_i h_{\rho(i)} H_2(w)}$.
6: **end for**
7: Compute $C_{\Delta_w} = e(g,g)^{\alpha s_w H_2(w)}, C'_{\Delta_w} = g^{s_w H_2(w)}$.
8: Denote the ciphertext of index keyword w as

$$\mathcal{I}_w = ((A,\rho), C_{\Delta_w}, C_{\Delta'_w}, \{C_{\Delta_w,i}, D_{\Delta_w,i}\}_{1 \le i \le l}).$$

Algorithm 3.2 DataEnc
9: Let SKE = (Gen, Enc, Dec) be a symmetric encryption.
10: Choose a random seed $\mathcal{G}$ from $\mathbb{G}_2$.
11: Generate a symmetric key sk by running SKE.Gen($\mathcal{G}$).
12: Encrypt f as $\widehat{f}$ by running SKE.Enc(sk, f).

Algorithm 3.3 KeyEnc
13: Define an LSSS policy $\mathbb{L}_f = (A', \rho', \mathcal{H}')$, where A' is an $l \times n$ matrix and $\mathcal{H}' = (h'_{\rho(1)}, ..., h'_{\rho(l)}) \in \{1,2,...,\chi\}$.
14: Choose two random vectors $v_f = (s_f, x_2, ..., x_n)$ and $v'_f = (s'_f, x'_2, ..., x'_n)$.
15: Choose two groups of random values $r_1, ..., r_l$ and $r'_1, ..., r'_l$ from $\mathbb{Z}_q^*$.
16: **for** i=0 to l **do**
17: Compute:
(1) $\lambda_i = v_f \cdot A'_i$ and $\lambda'_i = v'_f \cdot A'_i$, where A'_i denotes the ith row of matrix A'.
(2) $C_{\Delta_f,i} = g^{\beta\lambda_i} H_1(\rho(i))^{-r_i h'_{\rho(i)}}, D_{\Delta_f,i} = g^{r_i h'_{\rho(i)}}$ and
$C_{\Lambda_f,i} = g^{\beta\lambda'_i} H_1(\rho(i))^{-r'_i h'_{\rho(i)}}, D_{\Lambda_f,i} = g^{r'_i h'_{\rho(i)}}$.
18: **end for**
19: Compute $C_{\Delta_f} = \mathcal{G}e(g,g)^{\alpha s_f}, C'_{\Delta_f} = g^{s_f}$ and
$C_{\Lambda_f} = e(g,g)^{\alpha s'_f}, C'_{\Lambda_f} = g^{s'_f}$.
20: Denote the ciphertext of random seed $\mathcal{G}$ as

$$\widehat{\mathcal{G}} = ((A',\rho'), C_{\Delta_f}, C_{\Delta'_f}, \{C_{\Delta_f,i}, D_{\Delta_f,i}\}_{1 \le i \le l},$$
$$C_{\Lambda_f}, C_{\Lambda'_f}, \{C_{\Lambda_f,i}, D_{\Lambda_f,i}\}_{1 \le i \le l}).$$

21: **return** $\widehat{\mathcal{G}}$, $\widehat{f}$, $\mathcal{I}_w$.

5.4. Trapdoor generation

Using his private key $\mathcal{K}$, a data user u runs TrapGen algorithm to encrypt a query keyword Q to generate the query trapdoor. The

detail of this algorithm is present in Algorithm 4. In the algorithm, the random value ξ randomizes query trapdoor, which can guarantee that the identical query keywords have complete random query trapdoors. The feature is known as the trapdoor unlinkability and can prevent the adversary from inferring query information from the search histories.

Algorithm 4 TrapGen

Input:
System Public parameter **P**, the data user u's private key $\mathcal{K}$, and a query keyword Q.
Output:
Query trapdoor $\mathcal{T}_Q$.
1: Parse $\mathcal{K}$ as $\mathcal{K} = (K_1 = g^{\alpha}, K_2 = g^{\beta t}, K_3 = g^{t}, \forall s \in S : K_s = H_1(s)^t)$.
2: Choose a random exponent ξ from $\mathbb{Z}_q^*$.
3: Compute $T_1 = g^{\alpha} g^{\beta t \xi}$ and $T_2 = g^{t\xi H_2(Q)}$.
4: **for** each K_s in $\mathcal{K}$ **do**
5: Compute $T_s = H_1(s)^{t\xi}$.
6: **end for**.
7: **return** $\mathcal{T} = (T_1 = g^{\alpha} g^{\beta t \xi}, T_2 = g^{t\xi H_2(Q)}, \forall s \in S : T_s = H_1(s)^{t\xi})$.

5.5. Private search

In our PH-ABKS-DS scheme, the PrivateSearch algorithm is conducted by the cloud server and achieves the private match between the encrypted index keyword and the query trapdoor. If and only if the following two conditions hold simultaneously: (1) the underlying index keyword w in ciphertext $\mathcal{I}_w$ is identical to the query keyword Q in ciphertext $\mathcal{T}$ and (2) the attribute set embedded in $\mathcal{T}$ satisfies the LSSS matrix associated with $\mathcal{I}_w$, the algorithm outputs a set of encrypted data files that contain the keyword Q. Otherwise, the algorithm outputs an empty set. Different from traditional ABKS schemes, due to the hidden attribute values in access policy, the algorithm has to first calculate all minimum sets $\Phi_{(A,\rho)}$ satisfying the access policy. Then, if there exists a set $I \in \Phi_{(A,\rho)}$ that makes the following equation to be true, this implicitly indicates the two conditions above hold simultaneously,

$$C_{\Delta_w} = e(C'_{\Delta_w}, T_1) \Big/ \prod_{i \in I} \left(e(C_{\Delta_w,i}, T_2) e(D_{\Delta_w,i}, T_{\rho(i)}) \right)^{\omega_i},$$

where $C_{\Delta_w}, C'_{\Delta_w}, C_{\Delta_w,i}, D_{\Delta_w,i}$ and $T_1, T_2, T_{\rho(i)}$ are the ciphertext components in $\mathcal{I}_w$ and $\mathcal{T}$, respectively; $i \in I$ and $\rho(i)$ corresponds to the attribute in the ith row of A as well as $\rho(i)$ is equal to an attribute $s \in S$, S is the attribute set embedded in that private key encrypting Q; $\sum_{i \in I} \omega_i A_i = (1, 0, \ldots, 0)$. We will give the correctness proof of the algorithm in the next section.

Algorithm 5 PrivateSearch

Input:
A ciphertext $\mathcal{I}_w$ of index keyword w and a query trapdoor $\mathcal{T}$ of query keyword Q.
Output:
A collection $\widehat{\mathcal{R}}$ of encrypted data files containing keyword Q.
1: Calculate set $\Phi_{A,\rho}$ from (A, ρ).
2: Check if there exists a set $I \in \Phi_{A,\rho}$ that satisfies $\{\rho(i) | i \in I\}$ and

$$C_{\Delta_w} = e(C'_{\Delta_w}, T_1) \Big/ \prod_{i \in I} \left(e(C_{\Delta_w,i}, T_2) e(D_{\Delta_w,i}, K_{\rho(i)}) \right)^{\omega_i},$$

where $\sum_{i \in I} \omega_i A_i = (1, 0, ..., 0)$.
3: **if** such an I exists **then**
4: **return** $\widehat{\mathcal{R}} = \{(\widehat{\mathcal{G}_j}, \widehat{f_j}), ..., (\widehat{\mathcal{G}_k}, \widehat{f_k})\}$, where each $\widehat{f}$ contains the keyword Q.
5: **else**
6: **return** $\widehat{\mathcal{R}} = \emptyset$.
7: **end if**

5.6. Data decryption

The data user runs Decryption algorithm to decrypt the search result set $\widehat{\mathcal{R}}$ returned by the cloud servers. The detail of this algorithm is present in Algorithm 6. For each encrypted data file $(\widehat{\mathcal{G}}, \widehat{f})$ in $\widehat{\mathcal{R}}$, if and only if the data user owns the access permission to data file f, his private key $\mathcal{K}$ can recover from $\widehat{\mathcal{G}}$ the symmetric seed $\mathcal{G}$, by which f can be recovered from $\widehat{f}$. Otherwise, $\widehat{f}$ will be discarded. Essentially, we use the decryption algorithm in [22] except for hiding the attribute values in the access policy. Here, we first need to perform a test to find out a set $I \in \Phi_{A,\rho}$ (Line 6) and then use such an I to decrypt $\widehat{\mathcal{G}}$ (Lines 8 and 9). After obtaining the symmetric seed $\mathcal{G}$, the data user generates the actual symmetric by running sk = SKE.Gen($\mathcal{G}$) that can be used to decrypt the encrypted data file.

Algorithm 6 Decryption

Input:
a collection $\widehat{\mathcal{R}}$ of encrypted data files and the data user's private key $\mathcal{K}$.
Output:
a collection $\mathcal{R}$ of data files.
1: **if** $\widehat{\mathcal{R}} = \emptyset$ **then**
2: **return** $\mathcal{R} = \emptyset$.
3: **else**
4: **for** each $(\widehat{\mathcal{G}}, \widehat{f})$ in $\widehat{\mathcal{R}}$ **do**
5: Calculate set $\Phi_{A,\rho}$ from (A, ρ).
6: Check if there exists a set $I \in \Phi_{A,\rho}$ that satisfies $\{\rho(i) | i \in I\}$ and

$$C_{\Lambda_f} = \frac{e(C_{\Lambda'_f}, K_1 K_2)}{\prod_{i \in I} \left(e(C_{\Lambda_f,i}, K_3) e(D_{\Lambda_f,i}, K_{\rho(i)}) \right)^{\omega_i}},$$

where $\sum_{i \in I} \omega_i A_i = (1, 0, ..., 0)$.
7: **if** such an I exists **then**
8: Use I to compute

$$\frac{e(C_{\Delta'_f}, K_1 K_2)}{\prod_{i \in I} \left(e(C_{\Delta_f,i}, K_3) e(D_{\Delta_f,i}, K_{\rho(i)}) \right)^{\omega_i}} = e(g, g)^{\alpha s_f}.$$

9: Compute

$$C_{\Delta_f} / e(g, g)^{\alpha s_f} = \mathcal{G} e(g, g)^{\alpha s_f} / e(g, g)^{\alpha s_f} = \mathcal{G}.$$

10: Run SKE.Gen($\mathcal{G}$) to generate the symmetric key sk.
11: Decrypt $\widehat{f}$ as f by running SKE.Dec($sk, \widehat{f}$).
12: $\mathcal{R} = \mathcal{R} \cup \{f\}$.
13: **end if**
14: **end for**
15: **return** $\mathcal{R} = \{f_i, ..., f_k\}$.
16: **end if**

6. Correctness and security analysis of PH-ABKS-DS

6.1. Correctness of private search

Given an encrypted index keyword $\mathcal{I}_w$ and a query trapdoor $\mathcal{T}_Q$, according to Algorithm 5, we first calculate all minimum authorized sets $\Phi_{(A,\rho)}$, if Q is equal to w and there exists a set $I \in \Phi_{(A,\rho)}$ that satisfies

$$C_{\Delta_w} = e(C'_{\Delta_w}, T_1) \Big/ \prod_{i \in I} \left(e(C_{\Delta_w,i}, T_2) e(D_{\Delta_w,i}, T_{\rho(i)}) \right)^{\omega_i},$$

Table 2
The ciphertext about index keyword and query keyword that $\mathcal{A}$ can see.

$\mathcal{I}_w$	$\mathcal{T}_Q$	Intermediate results
$e(g,g)^{\alpha s_w H_2(w)}$		$e(g,g)^{s_w \alpha H_2(w)} e(g,g)^{s_w H_2(w) \beta t \xi}$
$g^{s_w H_2(w)}$	$g^{t\xi H_2(Q)}$	$e(g,g)^{H_2(Q)\beta t \xi s_w}$
$g^{\gamma_i h_{\rho(i)} H_2(w)}$		$e(g,g)^{s_w \alpha H_2(w)}$

where $\sum_{i\in I} \omega_i A_i = (1,0,\ldots,0)$. The private search is correct, because

$$\begin{aligned} e(C'_{\Delta_w}, T_1) &= (g^{s_w H_2(w)}, g^{\alpha} g^{\beta t \xi}) \\ &= e(g^{s_w H_2(w)}, g^{\alpha}) e(g^{s_w H_2(w)}, g^{\beta t \xi}) \\ &= e(g,g)^{s_w \alpha H_2(w)} e(g,g)^{s_w H_2(w) \beta t \xi} \end{aligned}$$

and

$$\begin{aligned} &\prod_{i\in I} \left(e(C_{\Delta_w,i}, T_2) e(D_{\Delta_w,i}, T_{\rho(i)}) \right)^{\omega_i} \\ &= \prod_{i\in I} \Big(e(g^{\beta\lambda_i} H_1(\rho(i))^{-\gamma_i h_{\rho(i)}}, g^{t\xi H_2(Q)}) \cdot \\ &\qquad e(g^{\gamma_i h_{\rho(i)} H_2(w)}, H_1(\rho(i))^{t\xi}) \Big)^{\omega_i} \\ &= \prod_{i\in I} \Big(e(g^{\beta\lambda_i}, g^{t\xi H_2(Q)}) e(H_1(\rho(i))^{-\gamma_i h_{\rho(i)}}, g^{t\xi H_2(Q)}) \cdot \\ &\qquad e(g^{\gamma_i h_{\rho(i)} H_2(w)}, H_1(\rho(i))^{t\xi}) \Big)^{\omega_i} \\ &= \prod_{i\in I} \Big(e(g^{\beta\lambda_i}, g^{t\xi H_2(Q)}) \cdot \\ &\qquad e(H_1(\rho(i)), g)^{-\gamma_i h_{\rho(i)} t\xi H_2(Q) + \gamma_i h_{\rho(i)} t\xi H_2(w)} \Big)^{\omega_i} \\ &\overset{w=Q}{=\!=} \prod_{i\in I} \Big(e(g,g)^{\lambda_i H_2(Q)\beta t\xi} \Big)^{\omega_i} \\ &= e(g,g)^{H_2(Q)\beta t\xi \sum_{i\in I} \lambda_i \omega_i} = e(g,g)^{H_2(Q)\beta t\xi \sum_{i\in I} v_w A_i \omega_i} \\ &= e(g,g)^{H_2(Q)\beta t\xi \sum_{i\in I} \lambda_i \omega_i} \\ &= e(g,g)^{H_2(Q)\beta t\xi (s_w, x_2, \ldots, x_n)(1,0,\ldots,0)} \\ &= e(g,g)^{H_2(Q)\beta t\xi s_w}. \end{aligned}$$

Thus, we have

$$\begin{aligned} &e(C'_{\Delta_w}, T_1) \Big/ \prod_{i\in I} \left(e(C_{\Delta_w,i}, T_2) e(D_{\Delta_w,i}, T_{\rho(i)}) \right)^{\omega_i} \\ &= e(g,g)^{s_w \alpha H_2(w)} e(g,g)^{s_w H_2(w) \beta t\xi} \Big/ e(g,g)^{H_2(Q)\beta t\xi s_w} \\ &\overset{w=Q}{=\!=} e(g,g)^{s_w \alpha H_2(w)} = C_{\Delta_w}. \end{aligned}$$

By derivation above, we can observe that, actually, the private search processes implicitly contains the following two tests: (1) whether there exists a set $I \in \Phi_{(A,\rho)}$ embedded in $\mathcal{T}_Q$ satisfies the access policy in $\mathcal{I}_w$ and (2) whether the query keyword Q is equal to the index keyword w. As long as one of those two conditions does not hold, the above equation will return to be false. Not that, in a successful search ($w = Q$), the algorithm does not need to determine which concrete set I in $\Phi_{(A,\rho)}$ that makes the equation to be true.

6.2. Correctness of decrypting symmetric key

Different from `PrivateSearch` algorithm, in a successful decryption, the `Decrytpion` algorithm must find out the exact set I from $\Phi_{(A,\rho)}$, by using it to further conduct the decryption of the symmetric key. To do that, we introduce the redundant ciphertext components $C_{\Lambda_f}, C_{\Lambda'_f}$, and $\{C_{\Lambda_f,i}, D_{\Lambda_f,i}\}_{1\le i\le l}$ in the ciphertext $\widehat{\mathcal{G}}$, which will help the algorithm find out the correct set $I \in \Phi_{(A,\rho)}$ by verifying

$$C_{\Lambda_f} \overset{?}{=\!=} e(C_{\Lambda'_f}, K_1 K_2) \Big/ \prod_{i\in I} \left(e(C_{\Lambda_f,i}, K_3) e(D_{\Lambda_f,i}, K_{\rho(i)}) \right)^{\omega_i}.$$

Further, the algorithm uses the set I to decrypt $\widehat{\mathcal{G}}$ by computing

$$\begin{aligned} &e(C_{\Delta'_f}, K_1 K_2) \Big/ \prod_{i\in I} \left(e(C_{\Delta_f,i}, K_3) e(D_{\Delta_f,i}, K_{\rho(i)}) \right)^{\omega_i} \\ &= \frac{e(g^{s_f}, g^{\alpha} g^{\beta t})}{\prod_{i\in I} \left(e(g^{\beta\lambda_i} H_1(\rho(i))^{-r_i h'_{\rho(i)}}, g^t) e(g^{r_i h'_{\rho(i)}}, H_1(\rho(i))^t) \right)^{\omega_i}} \\ &= \frac{e(g^{s_f}, g^{\alpha}) e(g^{s_f}, g^{\beta t})}{\prod_{i\in I} e(g^{\beta\lambda_i}, g^t)^{\omega_i}} = \frac{e(g^{s_f}, g^{\alpha}) e(g^{s_f}, g^{\beta t})}{e(g,g)^{\beta t \sum_{i\in I} \lambda_i \omega_i}} \\ &= \frac{e(g^{s_f}, g^{\alpha}) e(g^{s_f}, g^{\beta t})}{e(g,g)^{\beta t \sum_{i\in I} v_f A_i \omega_i}} \\ &= e(g,g)^{\alpha s_f}, \end{aligned}$$

where $\sum_{i\in I} \omega_i A_i = (1,0,\ldots,0)$. Finally, the algorithm recovers the symmetric key $\mathcal{G}$ encrypting file f by computing

$$C_{\Delta_f} / e(g,g)^{\alpha s_f} = \mathcal{G} e(g,g)^{\alpha s_f} / e(g,g)^{\alpha s_f} = \mathcal{G}.$$

Substantially, we utilize the Waters ABE scheme [22] to construct an expressive and policy-hiding ABE to secure the symmetric key. For achieving the hidden access policy, we introduce a group of redundant ciphertext components, and a random exponent $h_{\rho(i)}$ multiplying by the random value r_i. The subtle but non-trivial adaptation cannot affect the correctness of the original scheme.

6.3. Security analysis

Obviously, in PH-ABKS-DS scheme, the security of the data files and the symmetric keys can be guaranteed by the semantically secure symmetric encryption and the underlying attribute-based encryption scheme in [22], respectively. Therefore, we focus on the security analysis of the private search algorithm.

By directly observing the ciphertexts $\mathcal{I}_w$, $\mathcal{T}_Q$, and the intermediate results the `PrivateSearch` algorithm produces, as shown in Table 2, an adversary $\mathcal{A}$ cannot obtain any useful information about the index keyword w and the query keyword Q, since $\mathcal{A}$ has to solve the Discrete Logarithm problem and Integer Factorization problem. The main reason is that in our scheme the plaintext binding with random value(s) is encrypted as the power of a group element in $\mathbb{G}_1$ or $\mathbb{G}_2$. As a result, no decryption algorithm can recover the plaintext from the corresponding ciphertext due to DL and IF hardness problems. We can also say that each element in Table 2 is indistinguishable from a truly random element in $\mathbb{G}_1$ or $\mathbb{G}_2$ when using the random s, γ, ξ for an encryption every time.

On the other hand, the proposed $ACKSA$ game tells us that the adversary $\mathcal{A}$ is allowed to query all oracles to attack our private search scheme. Therefore, given the challenge index keyword w_b, $\mathcal{A}$ can guess the correct value of b with probability 1 by querying oracle $\mathcal{O}_{\mathbf{PrivateSearch}}(\mathcal{I}_{w_b}, \mathcal{T}_{Q_i}) = 1$, where Q_i is a certain trapdoor keyword that has been queried. This may be the sole way for the adversary $\mathcal{A}$ to break through our scheme. However, we will prove that $\mathcal{A}$ can output the guess b' such that $b = b'$ with probability at most 1/2 under the $ACKSA$ game.

Theorem 1. *Our proposed PH-ABKS-DS scheme is semantically secure against the adaptively chosen-keyword search attack. A probabilistic polynomial time adversary $\mathcal{A}$ cannot win the ACKSA game with a non-negligible advantage such that he cannot break through our scheme to obtain any useful keyword information.*

Proof. **Setup.** The challenger $\mathcal{B}$ runs `Setup` algorithm to send the public system parameter $(\mathbb{G}_1, \mathbb{G}_2, e, g, q, \chi, H_1, H_2, e(g,g)^{\alpha}, g^{\alpha}, g^{\beta})$ to $\mathcal{A}$ and keep α secret. In our proof, H_1 is modeled as the random oracle and H_2 is a one-way hash function. $\mathcal{B}$ builds a global table $\mathbf{T}$ as the simulation of random oracle H_1.

Phase 1. When requesting the ith query trapdoor of keyword $Q_i (1 \le i \le n)$ for the attribute set S_i, $\mathcal{A}$ queries $\mathcal{O}_{\mathbf{KenGen}}$ and $\mathcal{O}_{\mathbf{TrapGen}}$ as follows.

$\mathcal{B}$ chooses a random $t \in \mathbb{Z}_q^*$ and computes $g^{\beta t}$ and g^t. For each attribute $s \in S$, $\mathcal{B}$ computes $g^{r_{(s)}t}$. If for the first time s is queried, $\mathcal{B}$ chooses a newly random value $r_{(s)}$ and pushes $(s, r_{(s)})$ into **T**; otherwise $r_{(s)}$ is fetched from **T** by retrieving s. $\mathcal{B}$ returns $\mathcal{K} = (g^{\alpha}, g^{\beta t}, g^t, \forall s \in S_i : K_s = g^{r_{(s)}t})$ as the response to oracle $\mathcal{O}_{\textbf{KenGen}}$. Next, $\mathcal{B}$ chooses a random ξ and returns $\mathcal{T}_{Q_i} = (T_1 = g^{\alpha} g^{\beta t \xi}, T_2 = g^{t\xi H_2(Q_i)}, \forall s \in S_i : T_s = g^{r_{(s)} t\xi})$ to $\mathcal{A}$ as the response to oracle $\mathcal{O}_{\textbf{TrapGen}}$.

Challenge. $\mathcal{A}$ defines a challenging access policy $\mathcal{M}^* = (A, \rho, \mathcal{H})$, where A is an $l \times n$ matrix and $\mathcal{H}$ is a group of random elements $(h_{\rho(1)}, \ldots, h_{\rho(l)}) \in \mathbb{Z}_q^l$. Two keywords w_0 and w_1 are chosen with the restriction that if $S_i (1 \le i \le n)$ satisfies $\mathcal{M}^*$, then $w_0, w_1 \notin \{Q_1, \ldots, Q_n\}$. $\mathcal{M}^*$, w_0, and w_1 are sent to the challenger $\mathcal{B}$, who chooses a random bit $b \in \{0, 1\}$ and encrypts w_b as follows. First, $\mathcal{B}$ chooses a random vector$v_{w_b} = (s_{w_b}, x_2, \ldots, x_n)$ and l random values $\gamma_1, \ldots, \gamma_l$. Second, for each attribute $\rho(i)(1 \le i \le l)$, if $\rho(i)$ has been queried before, $\mathcal{B}$ fetches the random value $r_{(\rho(i))}$ from table **T** by retrieving $\rho(i)$; otherwise, a newly random $r_{(\rho(i))}$ is chosen and $(\rho(i), r_{(\rho(i))})$ is pushed into **T**; $\mathcal{B}$ uses $r_{(\rho(i))}$ to compute $C_{\Delta_{w_b},i} = g^{\beta\lambda_i} g^{-\gamma_i h_{\rho(i)} r_{(\rho(i))}}$ and $D_{\Delta_{w_b},i} = g^{\gamma_i h_{\rho(i)} H_2(w_b)}$, where $\lambda_i = v_{w_b} \cdot A_i$ and A_i denotes the ith row of A. Finally, $\mathcal{B}$ computes $C_{\Delta_{w_b}} = e(g,g)^{\alpha s_{w_b} H_2(w_b)}$, $C'_{\Delta_{w_b}} = g^{s_{w_b} H_2(w_b)}$. $\mathcal{B}$ sends the ciphertext

$$\mathcal{I}_{w_b} = ((A, \rho), C_{\Delta_{w_b}}, C_{\Delta'_{w_b}}, \{C_{\Delta_{w_b},i}, D_{\Delta_{w_b},i}\}_{1 \le i \le l})$$

to $\mathcal{A}$, who observes the output of $\mathcal{O}_{\textbf{PrivateSearch}}(\mathcal{I}_{w_b}, \mathcal{T}_{Q_i})$ to decide b's value, where Q_i is the trapdoor keyword that has been queried before.

Phase 2. $\mathcal{A}$ repeats **Phase 1.** There is a restriction that if an attribute set S_x corresponding to a requested trapdoor $\mathcal{T}_{Q_x}$ satisfies $\mathcal{M}^*$, then $w_0, w_1 \neq Q_x$.

Guess. Obviously, if $\mathcal{A}$ can find out keyword Q and let $\mathcal{O}_{\textbf{PrivateSearch}}(\mathcal{I}_{w_b}, \mathcal{T}_Q) = 1$, by Q $\mathcal{A}$ can correctly output b, where Q a certain trapdoor keyword that has been queried before. Without loss of generality, we assume that $b = 1$, and there exist two cases in the query as follows.

(1) $Q = w_1$. We execute the following private search test:

$$e(C'_{\Delta_{w_1}}, T_1) \Big/ \prod_{i \in I} \left(e(C_{\Delta_{w_1},i}, T_2) e(D_{\Delta_{w_1},i}, T_{\rho(i)}) \right)^{\omega_i}.$$

Because

$$e(C'_{\Delta_{w_1}}, T_1) = e(g,g)^{s_{w_1} \alpha H_2(w_1)} e(g,g)^{s_{w_1} H_2(w_1) \beta t \xi}$$

and

$$\begin{aligned} &\left(e(C_{\Delta_{w_1},i}, T_2) e(D_{\Delta_{w_1},i}, T_{\rho(i)}) \right)^{\omega_i} \\ &= \left(e(g^{\beta\lambda_i}, g^{t\xi H_2(Q)}) \cdot e(g^{\gamma_i h_{\rho(i)} r_{(\rho(i))}}, g^{t\xi})^{-H_2(Q)+H_2(w_1)} \right)^{\omega_i} \\ &= \left(e(g,g)^{\lambda_i H_2(Q) \beta t \xi} \right)^{\omega_i} \qquad (Q = w_1), \end{aligned}$$

we have

$$\begin{aligned} &e(C'_{\Delta_{w_1}}, T_1) \Big/ \prod_{i \in I} \left(e(C_{\Delta_{w_1},i}, T_2) e(D_{\Delta_{w_1},i}, T_{\rho(i)}) \right)^{\omega_i} \\ &= \frac{e(g,g)^{s_{w_1} \alpha H_2(w_1)} e(g,g)^{s_{w_1} H_2(w_1) \beta t \xi}}{\prod_{i \in I} \left(e(g,g)^{\lambda_i H_2(Q) \beta t \xi} \right)^{\omega_i}}. \end{aligned}$$

In this case, the *ACKSA* game has the restriction that the attribute set S_Q in $\mathcal{T}_Q$ cannot satisfy challenging access policy $\mathcal{M}^*$ associated with $\mathcal{I}_{w_1}$. Thus, the `PrivateSearch` algorithm cannot find constants $\{w_i \in \mathbb{Z}_p\}_{i \in I}$ to recover the secret s_{w_1}. As a result, we have

$$\frac{e(g,g)^{s_{w_1} \alpha H_2(w_1)} e(g,g)^{s_{w_1} H_2(w_1) \beta t \xi}}{\prod_{i \in I} \left(e(g,g)^{\lambda_i H_2(Q) \beta t \xi} \right)^{\omega_i}} \neq e(g,g)^{\alpha s_{w_1} H_2(w_1)},$$

so that $\mathcal{O}_{\textbf{PrivateSearch}}(\mathcal{I}_{w_1}, \mathcal{T}_Q) = 0$.

(2) $Q \neq w_1$. In this case, the oracle $\mathcal{O}_{\textbf{PrivateSearch}}(\mathcal{I}_{w_1}, \mathcal{T}_Q)$ certainly outputs 0 whether the attribute set S_Q satisfies the challenging access policy $\mathcal{M}^*$ or not, because

$$\begin{aligned} &\left(e(C_{\Delta_{w_1},i}, T_2) e(D_{\Delta_{w_1},i}, T_{\rho(i)}) \right)^{\omega_i} \\ &= \left(e(g^{\beta\lambda_i}, g^{t\xi H_2(Q)}) \cdot e(g^{\gamma_i h_{\rho(i)} r_{(\rho(i))}}, g^{t\xi})^{-H_2(Q)+H_2(w_1)} \right)^{\omega_i} \\ &\neq \left(e(g,g)^{\lambda_i H_2(Q) \beta t \xi} \right)^{\omega_i} \qquad (Q \neq w_1), \end{aligned}$$

we have,

$$\begin{aligned} &e(C'_{\Delta_{w_1}}, T_1) \Big/ \prod_{i \in I} \left(e(C_{\Delta_{w_1},i}, T_2) e(D_{\Delta_{w_1},i}, T_{\rho(i)}) \right)^{\omega_i} \\ &\neq \frac{e(g,g)^{s_{w_1} \alpha H_2(w_1)} e(g,g)^{s_{w_1} H_2(w_1) \beta t \xi}}{\prod_{i \in I} \left(e(g,g)^{\lambda_i H_2(Q) \beta t \xi} \right)^{\omega_i}} \qquad (Q \neq w_1) \\ &\neq e(g,g)^{\alpha s_{w_1} H_2(w_1)}. \end{aligned}$$

Therefore, from the perspective of the adversary, $\mathcal{A}$ can output the guess b' such that $b = b'$ with probability at most $1/2$ under the $ACKSA$ game. □

7. Experimental evaluation

In order to experimentally verify the correctness and evaluate the performance of the PH-ABKS-DS scheme, we implement this construction in Java language with the widely used Java pairing based cryptography library JPBC [54]. In our experimentations, a prime order elliptic curve $y^2 = x^3 + x$ with symmetric pairing operations is chosen. We evaluate the time cost of `KeyGen`, `GipherGen`, `TrapGen`, `Decrytion` four algorithms in a Windows 7 system with 2.30 GHz Intel Core i5-6200 CPU, 4 GB memory and the private search algorithm `PrivateSearch` in a Windows 7 with 3.60 GHz Inter Core i7-7700 CPU, 16 GB memory.

7.1. Time cost evaluation for `KeyGen` and `TrapGen` algorithms

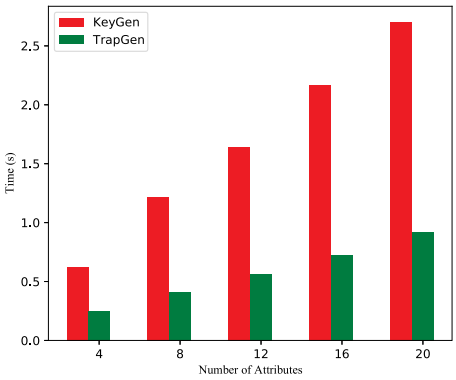


Fig. 3. The time cost of key generation and trapdoor generation.

Fig. 3 shows the time cost of key generation and trapdoor generation when varying the number of attributes. We can observe that both of them linearly increase with the increasing number of attributes. With the same number of attributes, the `KeyGen` algorithm needs to spend more time than `TrapGen` algorithm. This is because that the `KeyGen` algorithm involves the time-consuming hash operation H_1. When the number of attributes is set to be 20, the time to encrypt a search query is less than 1 s, which is very efficient.

7.2. Time cost evaluation for `CipherGen` algorithm

The `CipherGen` algorithm consists of three sub-algorithms. In the experimentation, as the well-designed AES can be used to instantiate the symmetric encryption `SKE` for `DataEnc`, we focus on the performance evaluations of `IndexWordEnc` and `KeyEnc`. Fig. 4 shows the

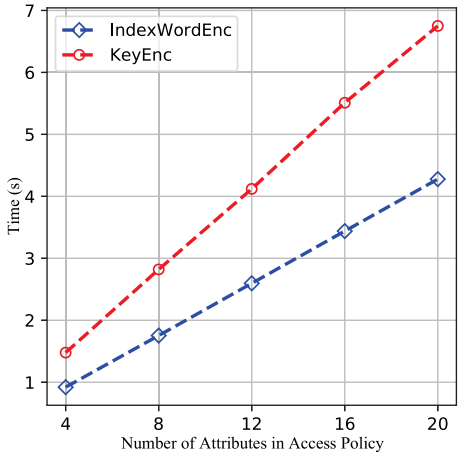


Fig. 4. The time cost of an index keyword encryption and a symmetric key encryption.

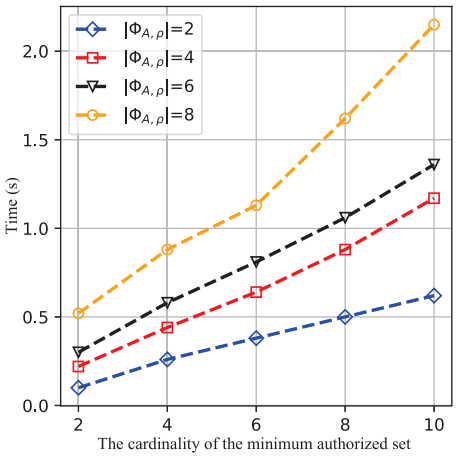


Fig. 5. The time cost of the private match.

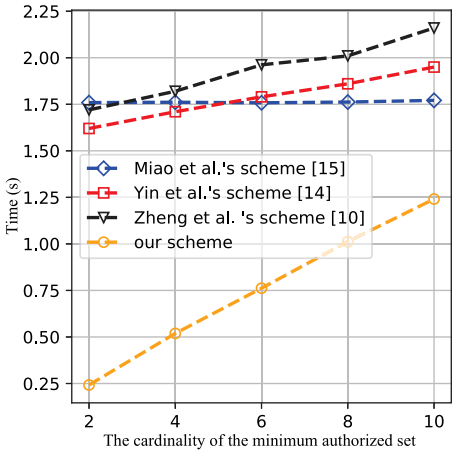


Fig. 6. The time cost of the private match for different schemes.

time cost of the index keyword encryption and the symmetric key encryption. Both of them are linear to the number of attributes in the access policy (i.e., the number of rows of the LSSS matrix). The `KeyEnc` algorithm needs to generate a group of redundant ciphertexts used to find out the minimum authorized set for decryption, which results in the more time cost due to requiring twice exponentiation operations as many as the `IndexWordEnc` algorithm. For example, when setting the number of attributes in the access policy to be 20, encrypting an index keyword and a key needs to spend about 4.28 s and 6.76 s, respectively. Though the index keyword encryption is a time-consuming process, it is one-time operation.

7.3. Time cost evaluation for `PrivateSearch` algorithm

We run the `PrivateSearch` algorithm at the server side to evaluate its time cost. Fig. 5 demonstrates the time cost of the private match between the query trapdoor and the encrypted index keyword under different size of $\Phi_{A,\rho}$ when varying the number of attributes in the minimum authorized set I. We can observe that the larger the sizes of I and $\Phi_{A,\rho}$ are, the more time would be spent on the private match. Moreover, in the experimentations, we find out that the size of $\Phi_{A,\rho}$ has the larger influence on the time cost of the `PrivateSearch` algorithm. This is because that it needs to linearly check from $\Phi_{A,\rho}$ whether there exist an $I \in \Phi_{A,\rho}$ satisfying the current private match, which results in a number of relatively time-consuming pairing operations. Obviously, the average check complexity is $\mathcal{O}((1+|\Phi_{A,\rho}|)/2)$. This is a tradeoff in our scheme between the search efficiency and the access policy hiding. For all this, our private match is still efficient as the proposed PH-ABKS-DS is constructed on the prime order group.

Fig. 6 shows the time cost of the privacy match for different schemes when varying the number of attributes in the minimum authorized set I and fixing the number of attributes of the data user to be 20. The experimental results demonstrate that except Miao et al.'s scheme [15] (we only implement their single keyword scheme, namely ABKS-HD), the execution time of the privacy match in PH-ABKS-DS, [10], and [14] grows almost linearly with the number of attributes in the minimum authorized set. We can see that our proposed PH-ABKS-DS has an obvious performance advantage. Moreover, PH-ABKS-DS is the only ABKS scheme with hidden access policy in these ABKS schemes above.

7.4. Time cost evaluation for `Decryption` algorithm

Similar to the `PrivateSearch` algorithm, the time cost of the `Decryption` algorithm is affected by the sizes of I and $\Phi_{A,\rho}$, where I denotes the minimum authorized set satisfying the current decryption and $\Phi_{A,\rho}$ denotes the set of all the minimum authorized set, and $I \in \Phi_{A,\rho}$. Moreover, the size of $\Phi_{A,\rho}$ has a larger influence on the time cost of the decryption. Fig. 7 shows the practical experimentation results when varying the sizes of I and $\Phi_{A,\rho}$, respectively. Compared with the `PrivateSearch` algorithm, the decryption process produces more time overhead, which is caused by two underlying reasons. One reason is that the `Decryption` algorithm is evaluated at the client machine with slightly weaker computation ability. The other one is that there is an indispensable decryption process to recover the original symmetric key, while the `PrivateSearch` algorithm has not the explicit decryption. On the other hand, compared to the existing expressive and policy-hiding ABE schemes, which are designed based on the composite order group, our proposed scheme bears most efficient decryption efficiency.

To provide performance comparisons of `Decryption` algorithm between our proposed PH-ABKS-DS scheme and other state-of-the-art related works, we implement the access policy-hiding ABE constructions in [16,17], and [53]. The experimental results show that, as shown in Fig. 8, the execution time of `Decryption` algorithm in our scheme is much less than that of schemes [16,17]. For example, when setting the number of attributes in set I to be 10, the time cost is about 1.6 s, 16.5 s, and 32.2 s in PH-ABKS-DS, [16], and [17], respectively. This also confirms that the policy-hiding ABE scheme based on composite order group is not suitable to construct the attribute-based keyword search scheme. Compared with the construction in [53], PH-ABKS-DS needs a little more execution time on decryption. However,

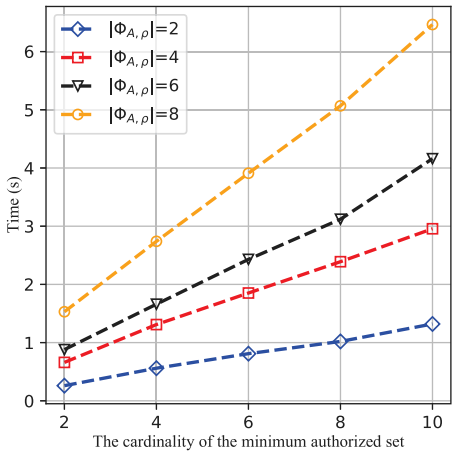


Fig. 7. Execution time of `Decryption` algorithm.

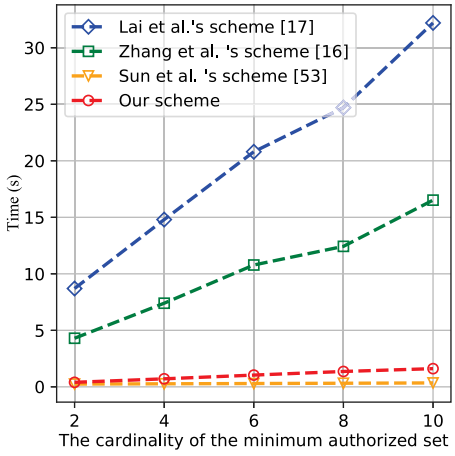


Fig. 8. Execution time of `Decryption` algorithm for different schemes.

the scheme [53] only supports AND-gates policy, and PH-ABKS-DS and those two schemes in [16,17] are based on LSSS access policy and have more flexible expressivity in practice.

8. Conclusion

In this paper, we investigate the efficient and policy-hiding attribute-based keyword search and data sharing scheme in cloud-assisted IoT. Since the existing expressive and policy-hiding ABE schemes were constructed on the composite order group, they are not suitable to be used to design the attribute-based keyword search system due to the high computation overhead. By studying, we find out that the policy-hiding ABE scheme can also be constructed based on the prime order group. Based on this conclusion, we instantiate a policy-hiding ABE over prime order group by utilizing the Waters ABE scheme, and further develop it as our proposed PH-ABKS-DS scheme in cloud-assisted IoT. The PH-ABKS-DS scheme can achieve the fine-grained and secure keyword search and data sharing with a practical and ideal efficiency. However, like all existing ABKS schemes, the proposed PH-ABKS-DS only considers the static data set and does not provide a mechanism to securely and dynamically update data. In our future work, we will explore the dynamical ABKS scheme with the secure and efficient data addition and deletion.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

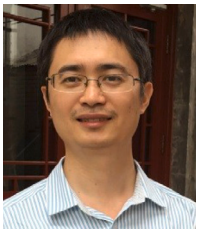
Acknowledgments

The work is supported by the National Natural Science Foundation of China under Grant Nos. 61972058, 61902123, and 61772088; The Natural Science Foundation of Hunan Province, China under Grant Nos. 2021JJ30760 and 2021JJ40636; Key Research Projects of Provincial Education Department of Hunan under Grant No. 20A041; Outstanding Youth Research Project of Provincial Education Department of Hunan under Grant Nos. 20B064 and 21B0775; Hui Yin and Yangfan Li contributed equally to this paper.

References

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, M. Zorzi, Internet of things for smart cities, IEEE Internet Things J. 1 (1) (2014) 22–32.
[2] H. Deng, Z. Qin, L. Sha, H. Yin, A flexible privacy-preserving data sharing scheme in cloud-assisted IoT, IEEE Internet Things J. 7 (12) (2020) 11601–11611.
[3] T. Wang, Q. Yang, X. Shen, T.R. Gadekallu, W. Wang, K. Dev, A privacy-enhanced retrieval technology for the cloud-assisted internet of things, IEEE Trans. Ind. Inf. (2021) http://dx.doi.org/10.1109/TII.2021.3103547.
[4] H. Xiong, T. Yao, H. Wang, J. Feng, S. Yu, A survey of public key encryption with search functionality for cloud-assisted IoT, IEEE Internet Things J. (2021) http://dx.doi.org/10.1109/JIOT.2021.3109440.
[5] S. Qi, Y. Lu, W. Wei, X. Chen, Efficient data access control with fine-grained data protection in cloud-assisted IIoT, IEEE Internet Things J. 8 (4) (2021) 2886–2899.
[6] K. Ren, C. Wang, Q. Wang, Security challenges for the public cloud, IEEE Internet Comput. 16 (1) (2012) 69–73.
[7] S. Kamara, K. Lauter, Cryptographic cloud storage, in: International Conference on Financial Cryptography and Data Security, 2010, pp. 136–149.
[8] A. Sahai, B. Waters, Fuzzy identity-base encryption, in: EUROCRYPT, 2005, pp. 457–473.
[9] C. Bosch, P. Harter, W. Jonker, A. Peter, A survey of provably secure searchable encryption, ACM Comput. Surv. 47 (2) (2014) 1–51.
[10] Q. Zheng, S. Xu, G. Ateniese, Vabks: Verifiable attribute-based keyword search over outsourced encrypted data, in: IEEE INFOCOM, 2014, pp. 522–530.
[11] W. Sun, S. Yu, W. Lou, Y.T. Hou, H. Li, Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud, in: IEEE INFOCOM, 2014, pp. 226–234.
[12] Y. Miao, J. Ma, X. Liu, X. Li, Z. Liu, H. Li, Practical attribute-based multi-keyword search scheme in mobile crowdsourcing, IEEE Internet Things J. 5 (4) (2018) 3008–3018.
[13] H. Yin, J. Zhang, Y. Xiong, L. Ou, F. Li, S. Liao, K. Li, CP-ABSE: A ciphertext-policy attribute-based searchable encryption scheme, IEEE Access 7 (1) (2019) 5682–5694.
[14] H. Yin, Z. Qin, J. Zhang, H. Deng, F. Li, K. Li, A fine-grained authorized keyword secure search scheme with efficient search permission update in cloud computing, J. Parallel Distrib. Comput. 135 (2020) 56–69.
[15] Y. Miao, J. Ma, X. Liu, X. Li, Q. Jiang, J. Zhang, Attribute-based keyword search over hierarchical data in cloud computing, IEEE Trans. Serv. Comput. 13 (6) (2020) 985–998.
[16] Y. Zhang, D. Zheng, R.H. Deng, Security and privacy in smart health: Efficient policy-hiding attribute-based access control, IEEE Internet Things J. 5 (3) (2018) 2130–2145.
[17] J. Lai, R.H. Deng, Y. Li, Expressive CP-ABE with partially hidden access structures, in: ACM Symposium on Information, Computer and Communications Security, 2012.
[18] P. Zeng, Z. Zhang, R. Lu, K.R. Choo, Efficient policy-hiding and large universe attribute-based encryption with public traceability for internet of medical things, IEEE Internet Things J. 8 (13) (2021) 10963–10972.
[19] Q. Li, Y. Zhang, T. Zhang, H. Huang, Y. He, J. Xiong, HTAC: Fine-grained policy-hiding and traceable access control in mHealth, IEEE Access 8 (2020) 123430–123439.
[20] S.D.M. Satar, M. Hussin, Z.M. Hanapi, M.A. Mohamed, Towards virtuous cloud data storage using access policy hiding in ciphertext policy attribute-based encryption, Future Internet 13 (2021) 279.
[21] T. Gan, Y. Liao, Y. Liang, Z. Zhou, G. Zhang, Partial policy hiding attribute-based encryption in vehicular fog computing, Soft Comput. 25 (2021) 10543–10559.

[22] B. Waters, Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization, in: Public Key Cryptography, 2011, pp. 53–70.
[23] D. Song, D. Wagner, A. Perrig, Practical techniques for searches on encrypted data, in: IEEE Symposium on Security and Privacy, 2000, pp. 44–55.
[24] R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky, Searchable symmetric encryption: improved deinitions and efficient constructions, in: ACM Conference on Computer and Communications Security, Vol. 19, 2006, pp. 79–88.
[25] S. Kamara, C. Papamanthou, T. Roeder, Dynamic searchable symmeteric encryption, in: ACM Conference on Computer and Communications Security, 2012, pp. 965–976.
[26] S. Kamara, C. Papamanthou, Parallel and dynamic searchable symmetric encryption, in: Financial Cryptography and Data Security, Springer Berlin Heidelberg, 2013, pp. 258–274.
[27] F. Hahn, F. Kerschbaum, Searchabel encryption with secure and efficient updates, in: ACM Conference on Computer and Communications Security, 2014, pp. 310–320.
[28] E. Stefanov, C. Papamanthou, E. Shi, Practical dynamic searchable encryption with small leakage, in: Network and Distributed System Security Symposium, 2014.
[29] R. Bost, *Σοφος*: FOrward secure searchable encryption, in: ACM Conference on Computer and Communications Security, 2016, pp. 1143–1154.
[30] K.S. Kim, M. Kim, D. Lee, J.H. Park, W. Kim, Forward secure dynamic searchable symmetric encryption with efficient updates, in: ACM Conference on Computer and Communications Security, 2017, pp. 1449–1463.
[31] X. Song, C. Dong, D. Yuan, Q. Xu, M. Zhao, Forward private searchable symmetric encryption with optimized I/O efficiency, IEEE Trans. Dependable Secure Comput. (2018) http://dx.doi.org/10.1109/TDSC.2018.2822294.
[32] M.S. Mohamad, G.S. Poh, Verifiable structured encrytpion, in: International Conference on Information Security and Cryptology, Springer, Berlin Heidelberg, 2013, pp. 137–156.
[33] H. Yin, Z. Qin, J. Zhang, L. Ou, K. Li, Achieving secure, universal, and fine-grained query results verification for secure search scheme over encrypted cloud data, IEEE Trans. Cloud Comput. 9 (1) (2021) 27–39.
[34] D. Boneh, G.D. Crescenzo, R. Ostrovsky, G. Persiano, Public-key encryption with keyword search, in: EUROCRYPR, 2004, pp. 506–522.
[35] P. Golle, J. Staddon, B. Waters, Secure conjunctive keyword search over encrypted data, in: Applied Cryptography and Network Security, 2004, pp. 31–45.
[36] D. Boneh, B. Waters, Conjunctive, subset, and range queries on encrypted data, in: Theory of Cryptography Conference, 2007, pp. 535–554.
[37] P. Xu, H. Jin, Q. Wu, W. Wang, Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack, IEEE Trans. Comput. 62 (11) (2013) 2266–2277.
[38] H. Yin, Z. Qin, L. Ou, K. Li, A privacy-enhanced and efficient search method for encrypted data in cloud computing, J. Comput. Syst. Sci. 90 (2017) 14–27.
[39] H. Yin, Z. Qin, J. Zhang, L. Ou, F. Li, K. Li, Secure conjunctive multi-keyword ranked search over encrypted cloud data for multiple data owners, Future Gener. Comput. Syst. 100 (2019) 689–700.
[40] W. Wang, P. Xu, D. Liu, L.T. Yang, Z. Yan, Lightweighted secure searching over public-key ciphertexts for edge-cloud-assisted industrial IoT devices, IEEE Trans. Ind. Inf. 16 (6) (2020) 4221–4230.
[41] M.S. Rahman, I. Khalil, N. Moustafa, A.P. Kalapaaking, A. Bouras, Blockchain-enabled privacy-preserving verifiable query framework for securing cloud-assisted industrial internet of things systems, IEEE Trans. Ind. Inf. (2021) http://dx.doi.org/10.1109/TII.2021.3105527.
[42] X. Zhang, C. Xu, H. Wang, Y. Zhang, S. Wang, FS-PEKS: Lattice-based forward secure public-key encryption with keyword search for cloud-assisted industrial internet of things, IEEE Trans. Dependable Secure Comput. 18 (3) (2021) 1019–1032.
[43] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in: IEEE Symposium on Security and Privacy, 2007, pp. 321–334.
[44] L. Cheng, C. Newport, Provably secure ciphertext policy ABE, in: ACM Conference on Computer and Communications Security, 2007, pp. 456–465.
[45] A. Lewko, B. Waters, Decentralizing attribute-based encryption, in: EUROCRYPT, 2011, pp. 547–567.
[46] V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encryption data, in: ACM Conference on Computer and Communications Security, 2006, pp. 89–98.
[47] R. Ostrovsky, A. Sahai, B. Waters, Attribute-based encryption with non-monotonic access structures, in: ACM Conference on Computer and Communications Security, 2007, pp. 195–203.
[48] M. Chase, S. Chow, Improving privacy and security in multi-authority attribute-based encryption, in: ACM Conference on Computer and Communications Security, 2009, pp. 121–130.
[49] S. Yu, C. Wang, K. Ren, W. Lou, Achieving secure, scalable, and fine-grained data access control in cloud computing, in: IEEE International Conference on Computer Communications, 2010, pp. 534–542.
[50] J. Hur, D.K. Noh, Attribute-based access control with efficient revocation in data outsourcing systems, IEEE Trans. Parallel Distrib. Syst. 22 (7) (2011) 1214–1221.
[51] T. Nishide, K. Yoneyama, K. Ohta, Attribute-based encryption with partially hidden encryptor specified access structures, in: Applied Cryptography and Network Security, 2008, pp. 111–129.
[52] T.V.X. Phuong, G. Yang, W. Susilo, Hidden ciphertext policy attribute-based encryption under standard assumptions, IEEE Trans. Inf. Forensics Secur. 11 (1) (2016) 35–45.
[53] J. Sun, H. Xiong, X. Liu, Y. Zhang, X. Nie, R.H. Deng, Lightweight and privacy-aware fine-grained access control for IoT-oriented smart health, IEEE Internet Things J. 7 (7) (2020) 6566–6575.
[54] http://gas.dia.unisa.it/projects/jpbc/index.html.



Hui Yin received his M.S. in Computer Software and Theory from Central South University, China, in 2008; and Ph.D. degree from the College of Information Science and Engineering at Hunan University, China, in 2018. He is currently an associate professor at the School of Computer Engineering and Applied Mathematics, Changsha University, China. His interests include data security and privacy, applied cryptography.

Yangfan Li is currently a Ph.D. student in Computer Science, Hunan University, China. His research interest includes, information security, parallel and distributed computing, machine learning and deep learning. He received the bachelor degree of engineering in the School of Automation, Huazhong University of Science and Technology.

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

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

Wei zhang received his M.S. degree and Ph.D. degree in the College of Computer Science and Electronic Engineering, Hunan University. Now he is working in Changsha University, Changsha, China. His research interests include privacy protection, wireless networks, signal processing, and machine learning.

Keqin Li is a SUNY Distinguished Professor of computer science with the State University of New York. He is also a Distinguished Professor with Hunan University, China. His current research interests include cloud computing, fog computing and mobile edge computing, energy-efficient computing and communication, embedded systems and cyber–physical systems, heterogeneous computing systems, big data computing, high performance computing, CPU–GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He has authored or coauthored more than 760 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He has chaired many international conferences. He is currently an associate editor of the ACM Computing Surveys and the CCF Transactions on High Performance Computing. He has served on the editorial boards of the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Computers, the IEEE Transactions on Cloud Computing, the IEEE Transactions on Services Computing, and the IEEE Transactions on Sustainable Computing. He is an IEEE Fellow.