An Attribute-Based Searchable Encryption Scheme for Cloud-Assisted IIoT

Hui Yin, Wei Zhang, Hua Deng, Zheng Qin, and Keqin Li, *Fellow, IEEE*

***Abstract*—The searchable encryption (SE) is a particular case of structured encryption, which has been intensively researched in the secure cloud storage system. By constructing a structured secure index, such as encrypted multimaps (EMMs), encrypted inverted index (EII), etc., SE can achieve efficient keyword search over the encrypted data set. However, existing SE constructions do not take search permissions into consideration, resulting in the lack of a mechanism of the data access control, which may not be suitable for Industrial Internet of Things (IIoT) applications, since an integrated industrial system contains all kinds of data with rigorous access permissions. In this article, we construct an attribute-based SE (ABSE) construction for a cloud-assisted IIoT application scenario. By designing the novel access policy-based structured secure index and the attribute-based search token, our construction achieves fine-grained keyword search privilege control over encrypted IIoT data as well as the same search complexity as the traditional SE. To the best of our knowledge, this is the first ABSE construction. We provide the correctness and security proofs for our construction. Experimental evaluation results in a real-world data set show the correctness and the practical search efficiency of the proposed ABSE.**

***Index Terms*—Attribute-based encryption (ABE), cloud-assisted Industrial Internet of Things (IIoT), searchable encryption (SE), secure index.**

I. Introduction

NOWADAYS, Industrial Internet of Things (IIoT) is playing an increasingly significant role in industry 4.0 [1]. By connecting a larger number of physical objects in industrial systems, including sensors, intelligent meters, RFID tags, and other intelligent terminal devices, IIoT can realize remote monitoring and management in modern industrial sectors [2], which has become an infrastructure to develop a smart, automatical, and sustainable modern industrial system.

Manuscript received 11 October 2022; revised 9 January 2023; accepted 3 February 2023. Date of publication 7 February 2023; date of current version 7 June 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 61972058, Grant 61902123, Grant 62002106, Grant 62002112, and Grant U20A20174; in part by the Natural Science Foundation of Hunan Province under Grant 2021JJ30760, Grant 2021JJ40636, and Grant 2021JJ40117; and in part by the Research Project of Provincial Education Department of Hunan under Grant 20A041, Grant 20B064, Grant 22B0822, and Grant 21B0775. *(Corresponding author: Wei Zhang.)*

Hui Yin, Wei Zhang, and Hua Deng are with the College of Computer Science and Engineering, Changsha University, Changsha 410022, China (e-mail: yhui@ccsu.edu.cn; zweihnu@hnu.edu.cn).

Zheng Qin is with the College of Computer Science and Electronic Engineering, Hunan University, Changsha 410008, China.

Keqin Li is with the Department of Computer Science, State University of New York at New Paltz, New Paltz, NY 12561 USA.

Digital Object Identifier 10.1109/JIOT.2023.3242964

Data is the vitally important assets for modern industrial systems. How to store and manage massive industrial data from IIoT day by day is an urgent need to resolve for industrial participants. Just under such a background, the cloud-assisted IIoT architecture emerges. By uploading the data via IoT gateways, enterprises can greatly reduce the local IT investments as well as enjoy the advantages of cloud computing, such as on-demand resource allocation, low cost, etc. However, the data outsourcing paradigm of cloud computing gives rise to a worrisome problem, i.e., the data security, since the outsourced data is no longer within the trust domain of data providers [3]. Encrypting data and storing ciphertexts at the cloud server can effectively mitigate risks of data leakage [4]. However, traditional encryption makes data retrieving for a data user a very challenging issue [5]. Though every time the data user can download all encrypted data and then decrypt them locally to pick up the goal data he is interested in, it is obviously impractical in the real-world application due to the expensive communication and computation costs.

The researchers rose to the challenge by proposing searchable encryption (SE) primitive, which enables a server to execute data searching over the structured secure index via a search token. Song et al. [6] presented the first practical SE scheme. Subsequent research refined the search complexity, security, and dynamics [7], [8], [9], [10]. Since SE focuses on the structured secure index, such as encrypted inverted index (EII) [7], encrypted multimaps (EMMs) [11], X-Set [12], etc., it is a particular case of structured encryption [13]. However, existing SE constructions do not take search permissions of a data user into consideration, resulting in the lack of the mechanism of the access control to encrypted data. A data user possesses *unlimited* search capabilities and can submit any search keyword to retrieve data he is interested in. The SE without the access control mechanism may not be suitable for IIoT applications, since an integrated industrial system contains all kinds of data with rigorous access permissions. For example, technical drawings cannot be accessed by a worker or even CFO, and CTO is not usually allowed to obtain financial statements.

By leveraging the attribute-based encryption (ABE) [14] primitive, recently proposed attribute-based keyword search (ABKS) techniques [15], [16], [17], [18], [19] can simultaneously achieve fine-grained search privilege control and textual matching between two encrypted keywords. In those schemes, a keyword is encrypted under an access policy and a query is encrypted using a set of attributes. If the attribute set first satisfies the access policy and then the keyword is textually equal to

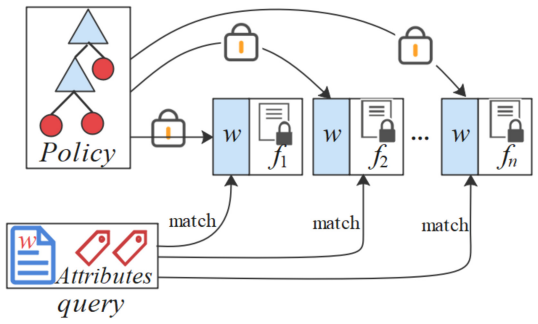


Fig. 1. Naive application of existing ABKS schemes.

the query, those schemes return TURE. We call the former process *search authorization* and the latter process *keyword text match*. Unfortunately, all existing ABKS schemes only explore an attribute-based match approach between two encrypted keywords and lack a structured secure index over real data set to support practical search efficiency. Therefore, those ABKS schemes are intrinsically not SE without the structured index. A naive application of existing ABKS schemes over a real data set is shown Fig. 1. Assume that there are n data files that contain a keyword w, generally, they are represented as a form of keyword–file pairs $(w, f_1), \ldots, (w, f_n)$ in the SE. The data provider specifies an access policy according to w's search permission, and in each pair, the keyword w is, respectively, encrypted under the access policy. A query keyword w with a set of attributes satisfying the access policy can obtain data files $f_1, \ldots, f_n$ by linearly matching encrypted keyword–file pairs. As a result, the time-consuming pairing and exponentiation operations in ABKS schemes are naturally proportional to the number of search results, which will result in a high search complexity $\mathcal{O}(n)$, especially, when n is large. We test the ABKS scheme in [15], when setting the number of attributes to be 12 and the number of search results to be 1000, the search time needs about 160 s.

Therefore, our goal is to develop a *truly* practical attribute-based SE (ABSE) with search complexity $\mathcal{O}(1)$ for real-world IIoT application. We achieve this goal by designing the access policy-based structured secure index for encrypted data set and the attribute-based search token for a query keyword, which can guarantee that the time-consuming ABKS is performed at most once regardless of the sizes of keywords and data files.

Contributions: In this article, we make three key contributions as follows.

1) Aiming at the information retrieval over large volume data, data security, and access control requirements in the cloud-assisted IIoT, we construct an ABSE scheme. By designing the novel access policy-based structured secure index and the attribute-based search token, our construction achieves fine-grained keyword search privilege control with practical search complexity. To the best of our knowledge, this is the first ABSE construction.
2) We give the formal security definition for ABSE and provide correctness and security proofs for our scheme.
3) We implement ABSE to experimentally evaluate the search efficiency of ABSE in the real-world data set. Also, we conduct experimental evaluation comparisons with similar work. The results demonstrate that ABSE is efficient and able to achieve fine-grained search permission control.

II. Related Work

A. Searchable Encryption

SE is a particular case of structured encryption [13], which enables a server to execute keyword-based information retrieval over ciphertexts via the well-designed structured secure index. Song et al. [6] proposed the first practical SE, but whose search complexity is linear in the size of the data files. Curtmola et al. [7] proposed sublinear search complexity SE construction by using EII, whose search complexity is only linear with the number of matching data files. Since then, the EII becomes a basic framework to develop various advanced SE schemes such as dynamic ones [8], [9], [10], [20], [21]. The dynamic SE allows the server to securely update (add and delete) data with the accepted communication and computation overhead. As adding or deleting data may reveal additional information to the server, the dynamic constructions need to achieve forward privacy [22], [23] and backward privacy [24], [25] against several leakage-abuse attacks such as the file injection attack [26]. However, all of the existing SE schemes do not take search permissions of data users into consideration, resulting in the lack of the mechanism of the access control to encrypted data.

B. Attribute-Based Keyword Search

By using attributes to define a decryptor's decryption capacity, the ABE primitive can achieve fine-grained and flexible access control over ciphertexts. Sahai and Waters proposed the first ABE scheme [14]. Goyal et al. [27] and Bethencourt et al. [28] introduced more expressive ABE, key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE). No matter what construction you choose, a successful decryption has to require that the attributes satisfy the access policy. Owning to the good feature that is able to achieve fine-grained access control over encrypted data, the subsequent researches are very active. Many variants of ABE were proposed to acquire more excellent properties, such as provable security [29], flexible expressivity [30], multiple authorities [31], hidden access structure [32], etc. By leveraging ABE, recently proposed ABKS techniques [15], [16], [17], [18], [19] can simultaneously achieve fine-grained search privilege control and textual matching between two encrypted keywords. In those schemes, a keyword is encrypted under an access policy and the other keyword is encrypted using a set of attributes. If the attribute set first satisfies the access policy and then those two keywords are textually equal, those schemes return TURE. Unfortunately, all existing ABKS schemes only explore an attribute-based match approach between two encrypted keywords and lack a structured secure index over real data set to support practical search efficiency. Therefore, those ABKS schemes are intrinsically not a SE without structured encrypted index. Directly employing them in the real-world IIoT data set will impose an impractical search complexity due to a

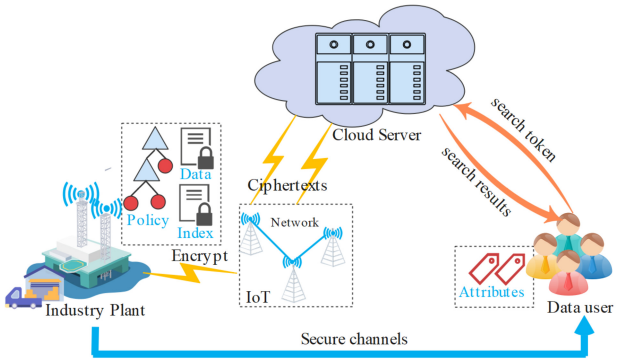


Fig. 2. System model.

large number of public-key operations, such as pairing and exponentiation.

III. Problem Formulations

A. System Model

The system model of ABSE for cloud-assisted IIoT involves three entities, the industry plant (data owner), the cloud server, and the data user, as shown in Fig. 2. The industry plant produces and encrypts data and builds a secure searchable index for efficient searchability over encrypted data. All ciphertexts are uploaded to the cloud via IoT devices. Different from the traditional SE, the secure index is associated with access policies to specify search permissions. When an authorized data user wishes to retrieve data containing a search keyword, he needs to generate a search token embedding the search keyword and his identities (attributes). Receiving the search token, the cloud executes a private matching between the token and the secure index. If the search keyword matches an index and the attributes satisfy the access policy in the index, the cloud server will return a set of search results to the data user in the encrypted manner. In this model, the secure channels are necessary, by which the data owner issues keys to the authorized data user.

B. Threat Model

Generally, the cloud is considered as an honest-but-curious passive adversary, who will follow security protocols, but may steal the outsourced data driven by the curiosity or the economic purpose. A secure SE scheme should prevent the cloud from inferring outsourced data from secure index and search token, but allows for appropriate information leakages, which is usually formalized by a group of predefined leakage functions. Finally, we assume that both the data owner and data user are fully trusted.

C. Security Definition

The security of an SE is usually captured by a called *leakage function* $\mathcal{L} = \{\mathcal{L}_{\mathsf{BuildIndex}}, \mathcal{L}_{\mathsf{Search}}\}$, where $\mathsf{BuildIndex}$ and Search are two algorithms in the SE scheme. If the information revealed from a real-world SE scheme is no more than that revealed from $\mathcal{L}$, we say the scheme is a secure SE. In general, the security definition based on the simulation-based framework [7] is formalized by an ideal-world experiment $\mathsf{IdealExp}$ versus a real-world experiment $\mathsf{RealExp}$. We define those two experiments as follows.

1) $\mathsf{RealExp}_{\mathcal{A}}(k)$: The experiment takes a security parameter k and a database DB as input and runs the real-world $\mathsf{BuildIndex}$ algorithm to encrypt DB and build secure searchable index over DB. A probabilistic polynomial time (PPT) adversary $\mathcal{A}$ performs polynomially bounded times search queries over $\mathsf{BuildIndex}(k, \mathsf{DB})$ and receives real transcripts generated from the real-world algorithm Search. $\mathcal{A}$ outputs a bit $b \in \{0, 1\}$.
2) $\mathsf{IdealExp}_{\mathcal{A},\mathcal{S}}(k)$: The experiment takes a security parameter k and a database DB as input, and there exists a simulator $\mathcal{S}$ that runs the leakage function $\mathcal{L}_{\mathsf{BuildIndex}}$ to encrypt DB and build secure searchable index over DB. The generated transcripts $\mathcal{S}(\mathcal{L}_{\mathsf{BuildIndex}}(k, \mathsf{DB}))$ are returned $\mathcal{A}$. $\mathcal{A}$ performs polynomially bounded times search queries over $\mathcal{S}(\mathcal{L}_{\mathsf{BuildIndex}}(k, \mathsf{DB}))$ and receives ideal transcripts generated from the simulation-world algorithm $\mathcal{S}(\mathcal{L}_{\mathsf{Search}})$. $\mathcal{A}$ outputs a bit $b \in \{0, 1\}$.

Definition 1 ($\mathcal{L}$ Adaptive Security): An SE is $\mathcal{L}$-adaptively secure if for any PPT adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that

$$\left|\Pr\left(\mathsf{RealExp}_{\mathcal{A}}(k) = 1\right) - \Pr\left(\mathsf{IdealExp}_{\mathcal{A},\mathcal{S}}(k) = 1\right)\right| \leq \mathsf{negl}(k)$$

where k is a security parameter, and $\mathsf{negl}(k)$ is a negligible function in k. We say $\mathsf{negl}(k)$ is negligible if $\mathsf{negl}(k) < 1/\mathsf{poly}(k)$ for any positive polynomial $\mathsf{poly}(\cdot)$.

More intuitively, the security definition states that for any PPT adversary $\mathcal{A}$, there exists a simulator $\mathcal{S}$ such that $\mathcal{A}$ cannot distinguish the outputs of $\mathsf{RealExp}_{\mathcal{A}}(k)$ from $\mathsf{IdealExp}_{\mathcal{A},\mathcal{S}}(k)$ with the nonnegligible advantage.

TABLE I
Notations for ABSE Construction

Notation	Description
k	system security parameter
$\boldsymbol{F}$	data file collection
F	a data file, $F \in \boldsymbol{F}$
$\mathtt{id}(F)$	the file identifier of F, $\mathtt{id}(F) \in \{0,1\}^k$
$\delta(\boldsymbol{F})$	the keyword set extracted from $\boldsymbol{F}$
w	a keyword, $w \in \delta(\boldsymbol{F})$
$\boldsymbol{F}(w)$	the set of data files (identifiers) containing keyword w

IV. Attribute-Based Searchable Encryption

First, we introduce several widely used notations in the SE, as shown in Table I, which are also used in our ABSE construction.

A. Construction

1) System Initialization: Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplication cyclic groups of prime order p, and g be a generator of $\mathbb{G}$. Define two hash functions

$$H_1 : \{0,1\}^* \rightarrow \mathbb{G}$$
$$H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$$

Algorithm 1: $\mathsf{BuildIndex}(\delta(\boldsymbol{F}), \{\boldsymbol{F}(w)\}_{w\in\delta(\boldsymbol{F})}, \boldsymbol{F})$

Input : The sets $\delta(\boldsymbol{F})$, $\{\boldsymbol{F}(w)\}_{w\in\delta(\boldsymbol{F})}$, $\boldsymbol{F}$
Output: The hash maps **T**, **M**, and **P**

1 **for** *Each w in* $\delta(\boldsymbol{F})$ **do**
2 Initialize an empty list $\mathsf{List}_w = \emptyset$;
3 Sample a random private-key $\mathcal{K}_0$ from $\{0,1\}^k$;
4 **for** $i = 1$ *to* $|\boldsymbol{F}(w)| - 1$ **do** // **Construct list** List_w **without containing the head node**
5 Generate private-key $\mathcal{K}_i = \mathsf{SKE.Gen}(1^k)$;
6 Construct the ith node in $\mathtt{L}_w$ as $\mathsf{N}_i = \langle \mathtt{id}(F_i)||\mathcal{K}_i||\mathcal{N}_{K_1}(w||(i+1))\rangle$;
7 Encrypt $\mathtt{N}_i$ with $\mathcal{K}_{i-1}$ as ciphertext $\mathtt{ciph}_{\mathtt{N}_i} = \mathsf{SKE.Enc}_{\mathcal{K}_{i-1}}(\mathtt{N}_i)$;
8 Store $\mathtt{ciph}_{\mathtt{N}_i}$ in map **M** at position $\mathcal{N}_{K_1}(w||i)$, $\mathbf{M}[\mathcal{N}_{K_1}(w||i)] \leftarrow \mathtt{ciph}_{\mathtt{N}_i}$;
9 **end**
10 Construct the last node in List_w as $\mathtt{N}_{|\boldsymbol{F}(w)|} = \langle \mathtt{id}(F_{|\mathbf{D}(w)|})||0^k||\mathtt{NULL}\rangle$;
11 Encrypt $\mathtt{N}_{|\boldsymbol{F}(w)|}$ with $K_{|\boldsymbol{F}(w)|-1}$ as ciphertext $\mathtt{ciph}_{\mathtt{N}_{|\boldsymbol{F}(w)|}} = \mathsf{SKE.Enc}_{K_{|\boldsymbol{F}(w)|-1}}(\mathtt{N}_{|\boldsymbol{F}(w)|})$;
12 Store $\mathtt{ciph}_{\mathtt{N}_{|\boldsymbol{F}(w)|}}$ in map **M** at position $\mathcal{N}_{K_1}(w|||\boldsymbol{F}(w)|)$, $\mathbf{M}[\mathcal{N}_{K_1}(w|||\boldsymbol{F}(w)|)] \leftarrow \mathtt{ciph}_{\mathtt{N}_{|\boldsymbol{F}(w)|}}$.
13 **end**
14 **for** *Each w in* $\delta(\boldsymbol{F})$ **do** // **Generate head node for each list** List_w
15 Specify an LSSS access policy (A, ρ), where A is an $l \times n$ matrix;
16 Choose a random vector $\overrightarrow{v} = (s, y_2, ..., y_n) \in \mathbb{Z}_p^n$;
17 **for** $i = 1$ *to* l **do**
18 Calculate $\lambda_i = \overrightarrow{v} A_i$, where A_i denotes the ith row of A;
19 **end**
20 Choose random $r_1, ..., r_l \in \mathbb{Z}_p^*$ and encrypt w as
$\mathcal{I}_1^w = e(g,g)^{\alpha s(H_2(w))^2}, \mathcal{I}_2^w = g^{sH_2(w)}, \{C_i^w = g^{x\lambda_i H_2(w)} H_1(\rho(i))^{-r_i H_2(w)}, D_i^w = g^{r_i H_2(w)}\}_{1\le i\le l}$;
21 Construct the head node of list List_w as $\mathcal{N}_{K_1}(w||1)||\mathcal{K}_0$;
22 Encrypt the head node as $\big(\mathcal{N}_{K_1}(w||1)||\mathcal{K}_0\big) \oplus \mathcal{F}_{K_2}(w)$;
23 Store the ciphertext in map **T** at position $\mathcal{H}(\mathcal{I}_1^w)$,
24 $\mathbf{T}[\mathcal{H}\big(e(g,g)^{\alpha s(H_2(w))^2}\big)] \leftarrow \big(\mathcal{N}_{K_1}(w||1)||\mathcal{K}_0\big) \oplus \mathcal{F}_{K_2}(w)$;
25 Store $\big(\mathcal{I}_2^w, \{C_i^w, D_i^w\}_{1\le i\le l}\big)$ in map **P** at position $\mathcal{R}_{K_3}(w)$,
26 $\mathbf{P}[\mathcal{R}_{K_3}(w)] \leftarrow \big(\mathcal{I}_2^w, \{C_i^w, D_i^w\}_{1\le i\le l}\big)$;
27 **end**
28 Return **T**, **M**, and **P**.

and three pseudo-random functions and a one-way hash function as follows:

$$\mathcal{R}: \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^k$$
$$\mathcal{N}: \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^k$$
$$\mathcal{F}: \{0,1\}^k \times \{0,1\}^* \to \{0,1\}^{2k}$$
$$\mathcal{H}: \{0,1\}^* \to \{0,1\}^k.$$

Let $e: \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a computationally efficient bilinear map with properties 1) $\forall X, Y \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p^*$, $e(X^a, Y^b) = e(X, Y)^{ab}$ and 2) $e(g, g) \neq 1$. If $e(X^a, Y^b) = e(Y^b, X^a) = e(X, Y)^{ab}$, e is a symmetric bilinear map. Choose two random $\alpha, x \in \mathbb{Z}_p^*$ and compute $e(g,g)^\alpha, g^x$. The data owner keeps α and x secret and opens other information to the public. Finally, let $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ be a symmetric encryption with ciphertext space $\mathcal{CIP}$ and $K_1, K_2, K_3 \in \{0,1\}^k$ be three symmetric keys.

2) Secure Searchable Index Construction: In order to achieve sublinear search complexity, we utilize the EII framework to construct the secure searchable index. Different from the traditional only-hashing encryption to keywords, we propose to bind an access policy to each keyword and produce the attribute-based ciphertext. Thus, on the encrypted data set, our scheme can achieve efficient search via a structured index and data access control via an attribute-based access policy, simultaneously. Algorithm 1 describes the implementation details of the secure searchable index construction. Each keyword w is associated with an encrypted posting list List_w. In List_w, expect the head node, the ith node N_i contains information $\langle \mathtt{id}(F_i)||\mathcal{K}_i||\mathcal{N}_{K_1}(w||(i+1))\rangle$, where $\mathtt{id}(F_i)$ is the identifier of data file F_i containing w, $\mathcal{K}_i$ a symmetric key encrypting N_{i+1}, and $\mathcal{N}_{K_1}(w||(i+1))$ looks like a *pointer* representing the storage address of node N_{i+1}; the fields $\mathcal{K}_i$ and $\mathcal{N}_{K_1}(w||(i+1))$ of the last node are set as 0^k and NULL, respectively. All nodes are encrypted and are stored in a hash map (lines 1–14 in Algorithm 1).

Next, each keyword w is specified a linear secret sharing scheme (LSSS [29]) access policy (A, ρ), where A is an $l \times n$ matrix and ρ is a function mapping each row of A to an attribute, which determines who have permission to search on it, and is encrypted by taking the following steps.

1) Generate a vector $\overrightarrow{v} = (s, y_2, \ldots, y_n)$ from randomly chosen $s, y_2, \ldots, y_n \in \mathbb{Z}_p^*$.
2) Calculate $\lambda_i = \overrightarrow{v} A_i$, for $i = 1$ to l.

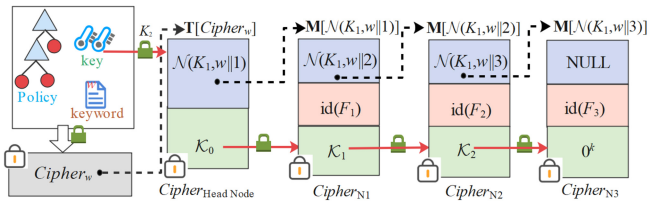


Fig. 3. Example of a posting list in the secure index construction.

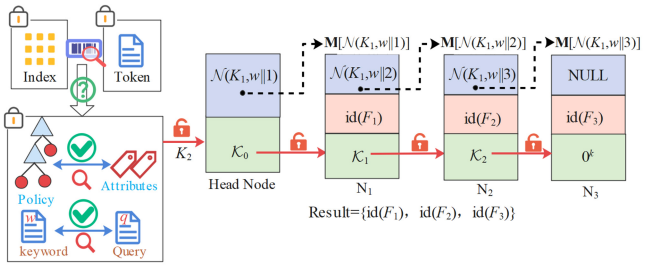


Fig. 4. Example for the search over the secure index.

3) Randomly choose $r_1, \ldots, r_l \in \mathbb{Z}_p^*$ and encrypt w as (lines 15–21 in Algorithm 1)

$$\mathcal{I}_1^w = e(g,g)^{\alpha s(H_2(w))^2}, \mathcal{I}_2^w = g^{sH_2(w)}$$
$$\left\{C_i^w = g^{x\lambda_i H_2(w)} H_1(\rho(i))^{-r_i H_2(w)}, D_i^w = g^{r_i H_2(w)}\right\}_{1\le i\le l}.$$

Here, we refer to $\mathcal{I}_1^w, \mathcal{I}_2^w, \{C_i^w, D_i^w\}$ as the search permission components of keyword w.

Finally, the algorithm constructs the head node $\mathcal{N}_{K_1}(w||1)||\mathcal{K}_0$ for list List_w, where $\mathcal{N}_{K_1}(w||1)$ *points to* the first node $\mathsf{N}_1 = \langle \mathtt{id}(F_1)||\mathcal{K}_1||\mathcal{N}_{K_1}(w||2)\rangle$ in List_w and $\mathcal{K}_0$ is used to encrypt N_1. The algorithm continues to compute $\mathcal{H}(\mathcal{I}_1^w)$ and $\mathcal{R}_{K_3}(w)$ and stores the head node $\mathcal{N}_{K_1}(w||1)||\mathcal{K}_0$ in $\mathbf{T}$ at position $\mathcal{H}(\mathcal{I}_1^w)$ and $(\mathcal{I}_2^w, \{C_i^w, D_i^w\})$ in $\mathbf{P}$ at position $\mathcal{R}_{K_3}(w)$ (lines 22–26 in Algorithm 1). Fig. 3 shows an example of an encrypted posting list with respect to a certain keyword w, where

$$\begin{aligned}
\mathrm{Cipher}_w &= \mathcal{H}(\mathcal{I}_1^w) = \mathcal{H}\left(e(g,g)^{\alpha s(H_2(w))^2}\right)\\
\mathrm{Cipher}_{\text{Head Node}} &= \left(\mathcal{N}_{K_1}(w||1)||\mathcal{K}_0\right) \oplus \mathcal{F}_{K_2}(w)\\
\mathrm{Cipher}_{\mathrm{N}_i} &= \mathsf{SKE.Enc}_{\mathcal{K}_{i-1}}\\
\mathrm{N}_i &= \left\langle \mathtt{id}(F_i)||\mathcal{K}_i||\mathcal{N}_{K_1}(w||(i+1))\right\rangle.
\end{aligned}$$

3) Private Key Generation and Distribution: In our system, the data owner plays the role of the authority to generate and distribute a private key for a data user. Having the private key, the data user encrypts a search keyword to generate the search token he wants to search. Assume that there is a data user with a set $\mathcal{U}$ of attributes, the data owner generates a private key as follows.

1) Choose a random $t \in \mathbb{Z}_p^*$ and compute $k_1 = g^\alpha$, $k_2 = g^{xt}$, $k_3 = g^t$.
2) Compute $\forall u \in \mathcal{U} : k_u = H_1(u)^t$, for each attribute u in $\mathcal{U}$.
3) Distribute the following private key

$$\mathcal{K} = \left(K_2, K_3, k_1 = g^\alpha, k_2 = g^{xt}, k_3 = g^t \quad \forall u \in \mathcal{U} : k_u = H_1(u)^t\right)$$

to the data user via secure communication channels.

4) Search Token Generation: When an authorized data user with assigned key $\mathcal{K}$ wants to obtain data files matching keyword q he has access to, the data user encrypts q using $\mathcal{K}$ to generate a search token as follows. On input $\mathcal{K}$ and q, the algorithm chooses random $v \in \mathbb{Z}_p^*$ and encrypts q as

$$\begin{aligned}
\mathsf{Token} = \Big(&\mathcal{R}_{K_3}(q), \mathcal{F}_{K_2}(q), \left(k_1(k_2)^v\right)^{H_2(q)} = g^{\alpha H_2(q)} g^{xtvH_2(q)}\\
&(k_3)^{vH_2(q)} = g^{tvH_2(q)}\\
&\forall u \in \mathcal{U} : (k_u)^{vH_2(q)} = H_1(u)^{tvH_2(q)}\Big).
\end{aligned}$$

5) Search Over Secure Index: When receiving a search token, the server executes a private search over the secure index and returns matching data files to the data user. We describe the implementation in Algorithm 2. On input the search token Token and the secure index ($\mathbf{T}$, $\mathbf{M}$, $\mathbf{P}$), the algorithm first retrieves $\mathbf{P}[\mathcal{R}_{K_3}(q)]$. If $\mathbf{P}[\mathcal{R}_{K_3}(q)] \neq \emptyset$ (this means there exists a keyword w in the secure index such that $w = q$), then the cloud server from secure index obtains keyword w's the permissions components $(\mathcal{I}_2^w)$, $\{\{C_i^w, D_i^w\}_{1\le i\le l}\}$. Then, if attribute set $\mathcal{U}$ in Token satisfies the access policy (A, ρ) associated with keyword w, the algorithm can find a subset $I \subset \{1, 2, \ldots, l\}$ and a set of constants $\{\omega_i \in \mathbb{Z}_p^*\}_{i\in I}$ such that $\Sigma_{i\in I}\omega_i A_i = (1, 0, \ldots, 0)$, where I is defined as $I = \{i : \rho(i) \in \mathcal{U}\}$ and further computes

$$\chi = e\left(\mathcal{I}_2^w, \eta\right) \Big/ \prod_{i\in I}\left(e\left(C_i^w, \sigma\right) e\left(D_i^w, \beta_{\rho(i)}\right)\right)^{\omega_i}$$

and

$$\phi = \mathcal{R}_{K_3}(\chi).$$

Finally, the algorithm determines in the secure index whether exists an encrypted posting list with respect to keyword $w(q)$ by retrieving $\mathbf{T}[\phi]$. If $\mathbf{T}[\phi] \neq \emptyset$, the algorithm iteratively decrypts all nodes to obtain file identifiers containing keyword w (lines 11–21 in Algorithm 2). Fig. 4 shows an example for the search and decryption to the goal posting list.

Obviously, if and only if the attribute set in the search token satisfies the policy in a posting list and the keywords in the search token and the posting list are identical, can the search algorithm return a group of file identifiers. Thus, the access control and data searching over encrypted data can be achieved simultaneously, with search complexity $\mathcal{O}(1)$.

B. Correctness

Theorem 1: Given a search token Token, if in the secure index, there exists an encrypted posting list with respect to keyword w that satisfies 1) the attribute set $\mathcal{U}$ contained in Token satisfies the access policy (A, ρ) associated with the posting list and 2) the search keyword q in Token is equal

Algorithm 2: Search(Token, **T**, **M**, **P**)

Input : The search token Token, and hash maps **T**, **M**, and **P**
Output: The search result set Result

Parse Token as $\mathcal{R}_{K_3}(q), \mathcal{F}_{K_2}(q), \eta = g^{\alpha H_2(q)} g^{xtvH_2(q)}, \sigma = g^{tvH_2(q)}, \beta = \{\forall u \in \mathcal{U} : H_1(u)^{tvH_2(q)}\}$;
Retrieve $\mathbf{P}[\mathcal{R}_{K_3}(q)]$ and Let $\varphi = \mathbf{P}[\mathcal{R}_{K_3}(q)]$;
if $\varphi = \emptyset$ **then**
  Return;
end
else
  Parse φ as $\mathcal{I}_2^w = g^{sH_2(w)}, \{C_i^w = g^{x\lambda_i H_2(w)} H_1(\rho(i))^{-r_i H_2(w)}, D_i^w = g^{r_i H_2(w)}\}_{1 \le i \le l}$);
  if *exists a subset* $I \subset \{1, 2, ..., l\}$ *and a set of constants* $\{\omega_i \in \mathbb{Z}_p^*\}_{i \in I}$ *such that* $\Sigma_{i \in I} \omega_i A_i = (1, 0, ..., 0)$ **then**
    Compute $\chi = e(\mathcal{I}_2^w, \eta) \Big/ \prod_{i \in I} \left(e(C_i^w, \sigma) e(D_i^w, \beta_{\rho(i)}) \right)^{\omega_i}$;
    Compute $\phi = \mathcal{H}(\chi)$;
    if $\boldsymbol{T}[\phi] \neq \emptyset$ **then**
      Set $c = 1, i = 1$, Result $= \emptyset$;
      Decrypt the head node in List_w by computing $\mathbf{T}[\phi] \oplus \mathcal{F}_{K_2}(q)$ as $\mathcal{N}_{K_1}(w||c)||\mathcal{K}_0$;
      while $\mathcal{K}_{i-1} \neq 0^k$ **do**
        Obtain node in List_w as $\mathtt{ciph}_{\mathrm{N}_i} = \mathbf{M}[\mathcal{N}_{K_1}(w||c)]$;
        Invoke $\mathsf{SKE.Dec}_{\mathcal{K}_{i-1}}(\mathtt{ciph}_{\mathrm{N}_i})$ to decrypt $\mathtt{ciph}_{\mathrm{N}_i}$ as $\left(\mathtt{id}(F_i)||\mathcal{K}_i||\mathcal{N}_{K_1}(w||(c+1))\right)$;
        Result = Result $\cup \{\mathtt{id}(F_i)\}$;
        $c = c + 1, i = i + 1$;
      end
    end
  end
end
Return Result.

to w, the search algorithm is able to return search results correctly.

Proof: According to Algorithm 2, we know that the search algorithm returns final file identifiers by retrieving hash map $\mathbf{T}[\mathcal{H}(\mathcal{I}_1^w)]$, where $\mathcal{I}_1^w = e(g, g)^{\alpha s (H_2(w))^2}$ is a search permission component of keyword w and is calculated in the secure index construction phase. In the search phase, a value χ to locate a goal posting list is calculated from $e(\mathcal{I}_2^w, \eta) / \prod_{i \in I} (e(C_i^w, \sigma) e(D_i^w, \beta_{\rho(i)}))^{\omega_i}$, where $(\mathcal{I}_2^w, \{C_i^w, D_i^w\})$ is the permission components of keyword w, and (η, σ, β) is the search token Token containing the attribute set $\mathcal{U}$ and the search keyword q. It is easy to see that if $\mathcal{I}_1^w = \chi$, the search algorithm returns search results correctly.

Now, we prove that, given a search token Token, if an encrypted posting list with respect to keyword w satisfies 1) the attribute set $\mathcal{U}$ contained in Token satisfies the access policy (A, ρ) associated with the posting list and 2) the search keyword q in Token is equal to w, then $\mathcal{I}_1^w = \chi$.

Since $\mathcal{U}$ satisfies (A, ρ), we can find a subset $I \subset \{1, 2, \ldots, l\}$ and a set of constants $\{\omega_i \in \mathbb{Z}_p^*\}_{i \in I}$ such that $\Sigma_{i \in I} \omega_i A_i = (1, 0, \ldots, 0)$, where I is defined as $I = \{i : \rho(i) \in \mathcal{U}\}$. The algorithm computes

$$
\begin{aligned}
e\left(\left(\mathcal{I}_2^w\right), \eta\right) &= \left(g^{sH_2(w)}, g^{\alpha H_2(q)} g^{xtvH_2(q)}\right) \\
&= e\left(g^{sH_2(w)}, g^{\alpha H_2(q)}\right) e\left(g^{sH_2(w)}, g^{xtvH_2(q)}\right) \\
&= e(g, g)^{s\alpha H_2(q) H_2(w)} e(g, g)^{sxtvH_2(q) H_2(w)}
\end{aligned}
$$

and

$$
\begin{aligned}
&\prod_{i \in I} \left(e\left(C_i^w, \sigma\right) e\left(D_i^w, \beta_{\rho(i)}\right) \right)^{\omega_i} \\
&= \prod_{i \in I} \left(e\left(g^{x\lambda_i H_2(w)} H_1(\rho(i))^{-r_i H_2(w)}, g^{tvH_2(q)}\right) \right. \\
&\qquad \left. \cdot\, e\left(g^{r_i H_2(w)}, H_1(\rho(i))^{tvH_2(q)}\right) \right)^{\omega_i} \\
&= \prod_{i \in I} \left(e\left(H_1(\rho(i))^{-r_i H_2(w)}, g^{tvH_2(q)}\right) \cdot e\left(g^{x\lambda_i H_2(w)}, g^{tvH_2(q)}\right) \right. \\
&\qquad \left. \cdot\, e\left(g^{r_i H_2(w)}, H_1(\rho(i))^{tvH_2(q)}\right) \right)^{\omega_i} \\
&= \prod_{i \in I} \left(e\left(g^{x\lambda_i H_2(w)}, g^{tvH_2(q)}\right) \right. \\
&\qquad \left. \cdot\, e\left(H_1(\rho(i)), g\right)^{-r_i tvH_2(w)H_2(q) + r_i tvH_2(w)H_2(q)} \right)^{\omega_i} \\
&= \prod_{i \in I} \left(e\left(g^{x\lambda_i H_2(w)}, g^{tvH_2(q)}\right) \right)^{\omega_i} \\
&= e\left(g^{xH_2(w)}, g^{tvH_2(q)}\right)^{\sum_{i \in I} \lambda_i \omega_i} \\
&= e\left(g^{xH_2(w)}, g^{tvH_2(q)}\right)^{\sum_{i \in I} \overrightarrow{v} A_i \omega_i} \\
&= e\left(g^{xH_2(w)}, g^{tvH_2(q)}\right)^{(s, y_2, \ldots, y_n)(1, 0, \ldots, 0)} \\
&= e\left(g^{xH_2(w)}, g^{tvH_2(q)}\right)^{s}.
\end{aligned}
$$

TABLE II
STORAGE OVERHEAD FOR DATA OWNER, DATA USER, AND CLOUD SERVER

Entity	Storage	Bit-length (bits)
Data Owner	K_1, K_2, K_3	$3k$
Data User	$K_2, K_3, k_1, k_2, k_3, \{k_u\}_{u\in\mathcal{U}}$	$2k + (3 + \lvert\mathcal{U}\rvert)\lvert\mathbb{G}\rvert$
Cloud Server	$\mathbf{T}$	$2k\lvert\delta(\boldsymbol{F})\rvert$
	$\mathbf{M}$	$3k\sum_{w\in\delta(\boldsymbol{F})}\lvert\boldsymbol{F}(w)\rvert$
	$\mathbf{P}$	$\lvert\delta(\boldsymbol{F})\rvert(2l+1)\lvert\mathbb{G}\rvert$
	$2k\lvert\delta(\boldsymbol{F})\rvert + 3k\sum_{w\in\delta(\boldsymbol{F})}\lvert\boldsymbol{F}(w)\rvert + \lvert\delta(\boldsymbol{F})\rvert(2l+1)\lvert\mathbb{G}\rvert$	

Therefore, we have

$$\begin{aligned}\chi &= e\big(\mathcal{I}_2^w, \eta\big)\Big/\prod_{i\in I}\Big(e\big(C_i^w, \sigma\big)e\big(D_i^w, \beta_{\rho(i)}\big)\Big)^{\omega_i}\\ &= \frac{e(g,g)^{s\alpha H_2(w)H_2(q)}e(g,g)^{sxtvH_2(w)H_2(q)}}{e\Big(g^{xH_2(w)}, g^{tvH_2(q)}\Big)^s}\\ &= e(g,g)^{s\alpha H_2(w)H_2(q)}\\ &\overset{w=q}{=\!=} e(g,g)^{\alpha s(H_2(w))^2}\\ &= \mathcal{I}_1^w\\ &\Rightarrow \mathcal{H}(\chi) = \mathcal{H}\big(\mathcal{I}_1^w\big).\end{aligned}$$

The proof is completed. ∎

C. Performance Analysis

Obviously, similar to the traditional SE based on the EII, the asymptotic complexities for computation and communication of the search algorithm achieve optimal sublinear complexity and are $\mathcal{O}(1)$ and $\mathcal{O}(|\boldsymbol{F}(w)|)$, respectively. Correspondingly, the decryption complexity is $\mathcal{O}(|\boldsymbol{F}(w)|)$, which is only related with the number of data files containing keyword w. Differently, our scheme achieves keyword search authorization by introducing the attribute-based public-key cryptosystem, which imposes extra computational cost due to relatively time-consuming pairing and exponentiation operations. Fortunately, in a search, the keyword search authorization is just performed at most once regardless of the number of keywords (posting lists) in the secure index such that the asymptotic complexity is $\mathcal{O}(1)$. This is an excellent feature to achieve our ABSE system with the practical search efficiency. In terms of the storage cost, the data owner needs to store three keys K_1, K_2, and K_3, the data user stores keys $K_2, K_3, k_1, k_2, k_3, \{k_u\}_{u\in\mathcal{U}}$, and the cloud server stores three hash maps $\mathbf{T}$, $\mathbf{M}$, $\mathbf{P}$, whose bit lengths are described in Table II, where notation $|*|$ denotes the cardinality if $*$ is a set, or the bit length if $*$ is an element. In the next section, we will conduct a group of experiments in a real data set to evaluate our scheme.

D. Security

Definition 2 (Leakage Function $\mathcal{L}_{\mathsf{buildIndex}}$): Given a security parameter k and a database DB denoted by $\boldsymbol{F}$, $\mathcal{L}_{\mathsf{buildIndex}} = (|\delta(\boldsymbol{F})|, |\boldsymbol{F}(w)|)$. That is, the $\mathsf{buildIndex}$ algorithm reveals information no more than the number of distinct keywords extracted from $\boldsymbol{F}$ and the number of file identifiers containing keyword w.

Definition 3 (Leakage Function $\mathcal{L}_{\mathsf{Search}}$): Given a security parameter k, secure index $(\mathbf{M}, \mathbf{T}, \mathbf{P})$, and a Token with keyword q, $\mathcal{L}_{\mathsf{Search}} = (\boldsymbol{F}(q), |\boldsymbol{F}(q)|)$. That is, the search algorithm reveals information no more than the file identifiers and the number of file identifiers containing keyword q.

Theorem 2: If $\mathcal{R}$, $\mathcal{N}$, and $\mathcal{F}$ are three pseudo-random functions, $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ is a semantically secure symmetric encryption, and $\mathcal{H}$ is a one-way hash function, the proposed ABSE is $\mathcal{L}$-adaptively secure, where $\mathcal{H}$ is modeled as a random oracle.

Proof: We prove there exists a polynomial-time simulator $\mathcal{S}$ that can conduct a perfect simulation based on the outputs of leakage functions $\mathcal{L}_{\mathsf{buildIndex}}$ and $\mathcal{L}_{\mathsf{Search}}$ (i.e., the trace of a history H [7]) such that for any PPT adversary $\mathcal{A}$, $\mathsf{RealExp}_{\mathcal{A}}(k)$ and $\mathsf{IdealExp}_{\mathcal{A},\mathcal{S}}(k)$ are indistinguishable.

Suppose that $\mathcal{S}$ is given $\mathcal{L}_{\mathsf{buildIndex}} = (|\delta(\boldsymbol{F})|, |\boldsymbol{F}(w)|)$ and $\mathcal{L}_{\mathsf{Search}} = \{(\boldsymbol{F}(q_1), |\boldsymbol{F}(q_1)|), \ldots, (\boldsymbol{F}(q_n), |\boldsymbol{F}(q_n)|)\}$, $\mathcal{S}$ conducts the following simulations to complete the experiment $\mathsf{IdealExp}_{\mathcal{A},\mathcal{S}}(k)$.

1) *Simulating $\boldsymbol{M}$:* For $1 \le i \le n$, $\mathcal{S}$ runs lines 1–13 in Algorithm 1 to generate the simulation $\mathbf{M}^*$ of $\mathbf{M}$, where $\delta(\boldsymbol{F}) = \{q_1, \ldots, q_n\}$ and $\boldsymbol{F}(w) \in \{\boldsymbol{F}(q_1), \ldots, \boldsymbol{F}(q_n)\}$. For $n < i \le (|\delta(\boldsymbol{F})| - n)$, it sets $\mathbf{M}^*[r_i^*] \leftarrow c_i^*$, where r_i^* and c_i^* are randomly chosen from $\{0,1\}^k$ and $\mathcal{CIP}$, respectively. The semantic security of SKE and the indistinguishability of pseudo-random function $\mathcal{N}$ guarantee that $\mathbf{M}^*$ and $\mathbf{M}$ are indistinguishable.

2) *Simulating $\boldsymbol{P}$:* For $1 \le i \le |\delta(\boldsymbol{F})|$, $\mathcal{S}$ chooses random $R_i^* \in \{0,1\}^k$ and $\mathcal{I}_i^* \in \mathbb{G}, \{C_{i,j}^*, D_{i,j}^* \in \mathbb{G}\}_{1\le j\le l}$, sets the simulation of $\mathbf{P}$ as

$$\mathbf{P}^*\big[R_i^*\big] \leftarrow \Big(\mathcal{I}_i^*, \Big\{C_{i,j}^*, D_{i,j}^*\Big\}_{1\le j\le l}\Big).$$

Recall that when computing the permission components $\mathcal{I}_2, \{C_i^w, D_i^w\}_{1\le i\le l}$ of a keyword, we use the random exponents s, r, and $-r$, respectively. Therefore, from the perspective of an adversary, they are indistinguishable from random elements in $\mathbb{G}$. The indistinguishability of the pseudo-random function $\mathcal{R}$ and the randomness of $\mathcal{I}_2$ and $\{C_j^w, D_j^w\}_{1\le j\le l}$ guarantee that $\mathbf{P}^*$ and $\mathbf{P}$ are indistinguishable.

3) *(Simulating $\boldsymbol{T}$):* $\mathcal{S}$ maintains a hash table $\mathbf{H}$. For $1 \le i \le |\delta(\boldsymbol{F})|$, $\mathcal{S}$ chooses two random strings $H_i^* \in \{0,1\}^k, X_i^* \in \{0,1\}^{2k}$ and sets $\mathbf{T}^*[H_i^*] \leftarrow X_i^*$. For $1 \le i \le n$, $\mathcal{S}$ continues to take the following steps.

1) Choose random $f_i^* \in \{0,1\}^{2k}, \eta_i^* \in \mathbb{G}, \sigma_i^* \in \mathbb{G}, \beta_i^* = \{\forall u \in \mathcal{U} : h_i^*(\rho(j)) \in \mathbb{G}, \rho(j) = u, j \in [1, |\mathcal{U}|]\}$.
2) Find the subset $I \subset \{1, 2, \ldots, l\}$ and a set of constants $\{\omega_j \in \mathbb{Z}_p^*\}_{j\in I}$ such that $\Sigma_{j\in I}\omega_j A_j = (1, 0, \ldots, 0)$ and compute

$$\chi_i^* = e\big(\mathcal{I}_i^*, \eta^*\big)\Big/\prod_{j\in I}\Big(e\Big(C_{i,j}^*, \sigma^*\Big)e\Big(D_{i,j}^*, h_i^*(\rho(j))\Big)\Big)^{\omega_j}$$

where $\mathcal{I}_i^*$, $C_{i,j}^*$, and $D_{i,j}^*$ are gotten from $\mathbf{P}^*$ according to the subscript i.
3) Set

$$\begin{aligned}\mathbf{H}\big[\chi_i^*\big] &\leftarrow H_i^*\\ \mathbf{T}^*\big[H_i^*\big] &\leftarrow \big(\mathcal{N}_{K_1}(q_i||1)||\mathcal{K}_0\big) \oplus f_i^*\end{aligned}$$

where $(\mathcal{N}_{K_1}(q_i||1)||\mathcal{K}_0)$ is the address of the first node of List_{q_i} stored in $\mathbf{M}^*$.

The indistinguishability of the pseudo-random function $\mathcal{F}$ and the XOR-based encryption scheme guarantee that $\mathbf{T}^*$ and $\mathbf{T}$ are indistinguishable.

4) *Simulating Token:* $\mathcal{S}$ sets

$$\mathsf{Token}_i^* = \left(R_i^*, f_i^*, \eta_i^*, \sigma_i^*, \beta_i^*\right)$$

when a search query Token_i^* is performed, if χ_i^* is in $\mathbf{H}$, the random oracle $\mathcal{H}$ answers H_i^* in $\mathbf{H}[\chi_i^*]$; otherwise, $\mathcal{H}$ returns a random value (line 10 in Algorithm 2).

Parse Token as $R = \mathcal{R}_{K_3}(q), f = \mathcal{F}_{K_2}(q), \eta = g^{\alpha H_2(q)} g^{xtvH_2(q)}, \sigma = g^{tvH_2(q)}, \beta = \{\forall u \in \mathcal{U} : H_1(u)^{tvH_2(q)}\}$, the indistinguishability between R (Resp., f) and R^* (Resp., f^*) can be immediately guaranteed by the indistinguishability of the pseudo-random function $\mathcal{R}$ (Resp., $\mathcal{F}$). Recall that when computing η, σ, and β, we introduce the random exponent v, by which they are fully randomized. From the perspective of an adversary, they are indistinguishable from random elements η^*, σ^*, and β^*. Therefore, Token and Token^* are indistinguishable for any PPT adversary.

In conclusion, according to the indistinguishability statements above, the simulation $\mathsf{IdealExp}_{\mathcal{A},\mathcal{S}}(k)$ has the same trace as $\mathsf{RealExp}_{\mathcal{A}}(k)$, which are indistinguishable for any PPT adversary $\mathcal{A}$. ■

V. Experimental Evaluation

We implement our scheme in the Java platform and use a real data set *Enron Email Data Set*[1] to evaluate the performance in a Windows 10 system with 3.60-GHz Inter Core i7-7700 CPU, 16-GB memory. The *Enron Email Data Set* was collected and prepared by the CALO Project (Cognitive Assistant that Learns and Organizes), and now contains a total of about 1.7-GB text files in the latest version, organized into folders. The pseudo-random functions $\mathcal{R}$ and $\mathcal{N}$ are instantiated with HMAC-SHA256 and $\mathcal{F}$ is instantiated with HMAC-SHA512, respectively. We use AES CTR (no padding) for $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$. The java pairing-based cryptography[2] (JPBC) library is used to support the group operations over an elliptic curve *Type A*,[3] including pairing, exponentiation, etc. In our experiments, for simplicity, all keywords are encrypted under an access policy of the form "a_1 AND a_2 AND, ... AND a_n" with only AND gate, where n denotes the number of attributes. By using the approach proposed in [31], we can convert the Boolean formula form to the corresponding LSSS policy, as shown in Fig. 5.

$$\begin{array}{l} \rho(1)=a_1 \\ \rho(2)=a_2 \\ \rho(3)=a_3 \\ \\ \rho(n)=a_n \end{array} \begin{pmatrix} 1 & 1 & 1 & 1 & \cdots & 1 \\ 0 & 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

LSSS access policy (A, ρ), A is a matrix with n rows, ρ is a function mapping each row of A to an attribute.

LSSS li d t t k d

Fig. 5. LSSS access policy used to encrypt keywords.

We first conduct three groups of experiments with different parameters to evaluate the search efficiency for our ABSE scheme, as shown in Fig. 6(a)–(c). We can observe that the time cost of the search is mainly determined by the number of matching data files, as shown in Fig. 6(a), and is not affected by the size of keywords and data files, as shown in Fig. 6 and (c). Due to the use of EII, the optimal sublinear search complexity is achieved in our construction, similar to the traditional SE. For example, when the numbers of matching data files and attributes are set to 10 000 and 16, respectively, the search time is about 0.76 s, which is extremely efficient in the real-world application. On the other hand, the search time of our ABSE construction is also affected by the number of attributes due to the attribute-based search authorization. However, the results do not change the practical search complexity since search authorization only needs to be executed at most once in the whole search process. For example, when varying the number of attributes from 4 to 16 with step size 4 and fixing the number of matching data files as 10 000, the time cost of the search increases from about 0.19 to about 0.35 s, which does not cause a notable degradation of the search efficiency.

To demonstrate the search efficiency of ABSE through comparisons with similar schemes, we also implement the classical SE SSE-1 proposed in [7] based on the inverted index framework and the first ABKS construction CP-ABKS proposed in [16]. In order to let CP-ABKS work in the real-world data set, we place CP-ABKS to the construction as shown in Fig. 1. To make the comparisons fair, we fix the number of attributes to be 12 ($n = 12$) in all experimental evaluations. Fig. 7(a)–(c) shows the search efficiencies of all of the three schemes are affected by the number of matching data files (search results) and are independent of the size of data files and index keywords. More importantly, experimental results demonstrate that although ABSE needs a little more time expended on the search than SSE-1, it is still practical in the real-world data set; however, CP-ABKS obtains search results at the cost of the expensive search overhead in the same experimental setting. For example, when the number of matching data files is up to 10 000, the needed time in ABSE, SSE-1, and CP-ABKS is about 0.28, 0.19, and 15.7×10^2 s. Compared to SSE-1, ABSE achieves the fine-grained keyword search authorization with a little extra time cost, and to CP-ABKS, ABSE achieves extremely efficient information retrieval in the real data set.

VI. Conclusion

In this article, we investigate the ABSE construction for cloud-assisted IIoT architecture. Motivated by SE, ABE, and ABKS techniques, we propose the first ABSE construction by designing the novel access policy-based structured secure

[1] http://nlp.cs.aueb.gr/software

[2] http://gas.dia.unisa.it/projects/jpbc/index.html

[3] http://http://gas.dia.unisa.it/projects/jpbc/docs/ecpg.html#TypeA

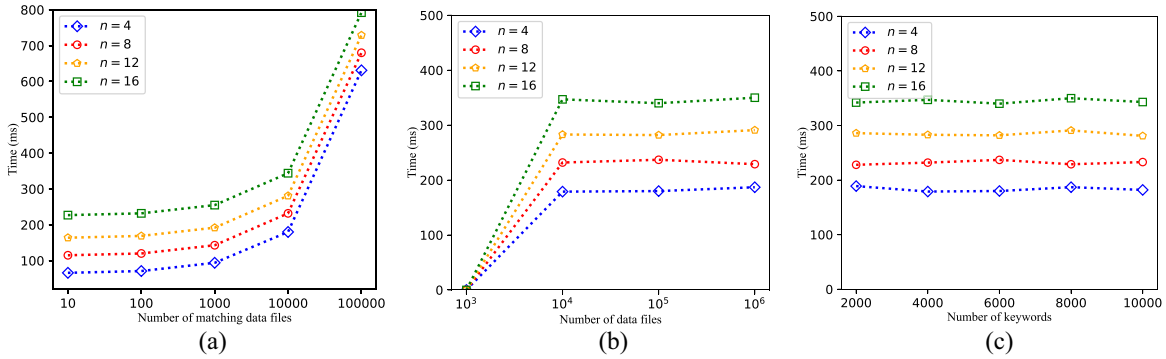


Fig. 6. Time cost of the search (a) when varying the number of matching data files with different number of attributes, (b) when varying the number of data files and fixing the number of matching data files to be 10 000 with different number of attributes, and (c) when varying the number of keywords and fixing the number of matching data files to be 10 000 with different number of attributes.

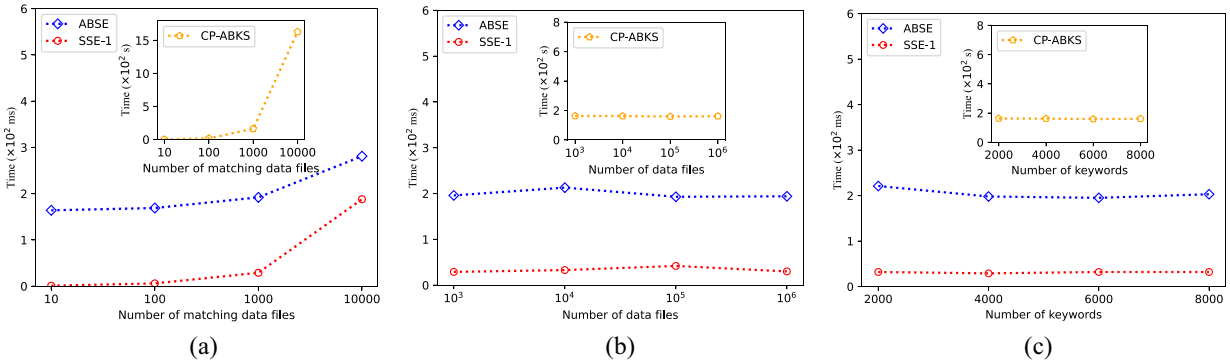


Fig. 7. Time cost of the search when fixing the number of attributes to be 12 ($n = 12$) and (a) varying the number of matching data files, (b) varying the number of data files and fixing the number of matching data files to be 1000, and (c) varying the number of keywords and fixing the number of matching data files to be 1000.

index and the attribute-based search token. Our construction achieves fine-grained keyword search privilege control and data searching over encrypted IIoT data. More importantly, our scheme is with the same search complexity as the traditional SE, which is only linearly dependent on the number of matching data files. The experimental evaluations on the real data set also show the practical search efficiency. For example, when the number of matching data files is 10 000, the search time in ABSE is only about 0.28 s, which is practical in the real-world application. Our experimental comparisons also demonstrate that compared to traditional SE, ABSE can achieve the fine-grained keyword search authorization with the same search complexity, and to traditional ABKS, ABSE can achieve extremely efficient information retrieval in the real data set. Although the proposed scheme only considers the static data set at present, it opens up a research line for the more advanced ABSE such as dynamic constructions.

REFERENCES

[1] I. C. Reinhardt, J. C. Oliveira, and D. T. Ring, "Current perspectives on the development of industry 4.0 in the pharmaceutical sector," *J. Ind. Inf. Integr.*, vol. 18, Jun. 2020, Art. no. 100131.

[2] X. Zhang, C. Xu, H. Wang, Y. Zhang, and 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.*, vol. 18, no. 3, pp. 1019–1032, May/Jun. 2021.

[3] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 69–73, Jan./Feb. 2012.

[4] S. Kamara and K. Lauter, "Cryptographic cloud storage," in *Proc. Int. Conf. Financ. Cryptogr. Data Security*, 2010, pp. 136–149.

[5] C. Wang, N. Cao, K. Ren, and W. lou, "Enabling secure and efficient ranked keyword search over outsourced cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 8, pp. 1467–1479, Aug. 2012.

[6] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Security Privacy*, 2000, pp. 44–55.

[7] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved deinitions and efficient constructions," in *Proc. ACM Conf. Comput. Commun. Security*, vol. 19, 2006, pp. 79–88.

[8] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmeteric encryption," in *Proc. ACM Conf. Comput. Commun. Security*, 2012, pp. 965–976.

[9] S. Kamara and C. Papamanthou, "Parallel and dynamic searchable symmetric encryption," in *Financial Cryptography Data Security*. Heidelberg, Germany: Springer, 2013, pp. 258–274.

[10] F. Hahn and F. Kerschbaum, "Searchable encryption with secure and efficient updates," in *Proc. ACM Conf. Comput. Commun. Security*, 2014, pp. 310–320.

[11] G. Amjad, S. Kamara, and T. Moataz, "Breach-resistant structured encryption," in *Proc. Privacy Enhanc. Technol. Symp.*, 2019, pp. 1–46.

[12] S. Lai et al., "Result pattern hiding searchable encryption for conjunctive queries," in *Proc. ACM Conf. Comput. Commun. Security*, 2018, pp. 745–762.

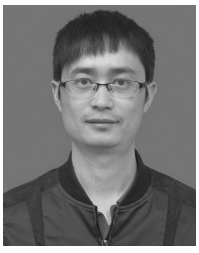
[13] M. Chase and S. Kamara, "Structured encryption and controlled disclosure," in *Proc. Adv. Cryptol.*, 2010, pp. 577–594.

[14] A. Sahai and B. Waters, "Fuzzy identity-base encryption," in *Proc. EUROCRYPT*, 2005, pp. 457–473.

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

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

[17] H. Wang, X. Dong, and Z. Cao, "Multi-value-independent ciphertext-policy attribute based encryption with fast keyword search," *IEEE Trans. Services Comput.*, vol. 13, no. 6, pp. 1142–1151, Nov./Dec. 2020.
[18] Y. Miao, J. Ma, X. Liu, X. Li, Q. Jiang, and J. Zhang, "Attribute-based keyword search over hierarchical data in cloud computing," *IEEE Trans. Services Comput.*, vol. 13, no. 6, pp. 985–998, Nov./Dec. 2020.
[19] H. Yin, Z. Qin, J. Zhang, H. Deng, F. Li, and K. Li, "A fine-grained authorized keyword secure search scheme with efficient search permission update in cloud computing," *J. Parallel Distrib. Comput.*, vol. 135, pp. 56–69, Jan. 2020.
[20] K. He, J. Chen, Q. Zhou, R. Du, and Y. Xiang, "Secure dynamic searchable symmetric encryption with constant client storage cost," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 1538–1549, 2020.
[21] I. Demertzis, J. G. Chamani, D. Papadopoulos, and C. Papamanthou, "Dynamic searchable encryption withsmall client storage," in *Proc. Netw. Distrib. Syst. Security (NDSS) Symp.*, 2020, pp. 1–18.
[22] Y. Watanabe, K. Ohara, M. Iwamoto, and K. Ohta, "Efficient dynamic searchable encryption with forward privacy under the decent leakage," in *Proc. ACM Conf. Data Appl. Security Privacy*, 2022, pp. 312–323.
[23] X. Song, C. Dong, D. Yuan, Q. Xu, and M. Zhao, "Forward private searchable symmetric encryption with Optimized I/O efficiency," *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 5, pp. 912–927, Sep./Oct. 2020.
[24] J. G. Chamani, D. Papadopoulos, C. Papamanthou, and R. Jalili, "New constructions for forward and backward private symmetric searchable encryption," in *Proc. ACM Conf. Comput. Commun. Security*, 2018, pp. 1038–1055.
[25] S. Sun et al., "Practical non-interactive searchable encryption with forward and backward privacy," in *Proc. Netw. Distrib. Syst. Security (NDSS) Symp.*, 2021, pp. 1–18.
[26] Y. Zhang, J. Katz, and C. Papamanthou, "All your queries are belong to us: The power of file-injection attacks on searchable encryption," in *Proc. USENIX Security Symp.*, 2016, pp. 707–720.
[27] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encryption data," in *Proc. ACM Conf. Comput. Commun. Security*, 2006, pp. 89–98.
[28] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Security Privacy*, 2007, pp. 321–334.
[29] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography* (Lecture Notes in Computer Science 6571). Heidelberg, Germany: Springer, 2011, pp. 53–70.
[30] G. Wang, Q. Liu, J. Wu, and M. Guo, "Hierarchical attribute-based encryption and scalable user revocation for sharing data in cloud servers," *Comput. Security*, vol. 30, no. 5, pp. 320–331, 2011.
[31] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Proc. EUROCRYPT*, 2011, pp. 547–567.
[32] J. Lai, R. H. Deng, and Y. Li, "Expressive CP-ABE with partially hidden access structures," in *Proc. ACM Symp. Inf. Comput. Commun. Security*, 2012, pp. 18–19.



Hui Yin received the M.S. degree in computer software and theory from Central South University, Changsha, China, in 2008, and the Ph.D. degree from the College of Information Science and Engineering, Hunan University, Changsha, in 2018.

He is currently an Associate Professor with the College of Computer Science and Engineering, Changsha University, Changsha. His interests are data security, privacy protection, and applied cryptography.

Wei Zhang received the M.S. and Ph.D. degrees in software engineering from the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China, in 2013 and 2020, respectively.

He is currently an Assistant Professor with the College of Computer Science and Engineering, Changsha University, Changsha. His research interests include privacy protection, wireless networks, signal processing, and machine learning.

Hua Deng received the M.S. degree in cryptography from Southwest Jiaotong University, Chengdu, China, in 2010, and the Ph.D. degree in information security from Wuhan University, Wuhan, China, in 2015.

He is currently an Associate Professor with the College of Computer Science and Engineering, Changsha University, Changsha, China. His research interests include applied cryptography, data security and privacy, and cloud security.

Zheng Qin received the Ph.D. degree in computer software and theory from Chongqing University, Chongqing, China, in 2001.

From 2010 to 2011, he served as a Visiting Scholar with the Department of Computer Science, University of Michigan at Ann Arbor, Ann Arbor, MI, USA. He is currently a Professor with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China. His main interests are network and data security, privacy, machine learning, and applied cryptography.

Keqin Li (Fellow, IEEE) received the B.S. degree in computer science from Tsinghua University, Beijing, China, in 1985, and the Ph.D. degree in computer science from the University of Houston, Houston, TX, USA, in 1990.

He is a SUNY Distinguished Professor of Computer Science with the State University of New York at New Paltz, New Paltz, NY, USA. He is also a National Distinguished Professor with Hunan University, Changsha, China. He holds over 70 patents announced or authorized by the Chinese National Intellectual Property Administration. He has authored or coauthored over 850 journal articles, book chapters, and refereed conference papers. 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.

Dr. Li has received several best paper awards. He is among the world's top 5 most influential scientists in parallel and distributed computing in terms of both single-year impact and career-long impact based on a composite indicator of Scopus citation database. 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 AAIA Fellow. He is also a member of Academia Europaea (Academician of the Academy of Europe).