An Attribute-Based Keyword Search Scheme for Multiple Data Owners in Cloud-Assisted Industrial Internet of Things

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

***Abstract*—The cloud-assisted industrial Internet of Things (IIoT) architecture can sustain highly available computation and massive storage services for modern industrial systems. When data owners store IIoT data to remote cloud platforms, the data security will face tough challenges. Cryptographic technologies endow an ability to guarantee data confidentiality. However, traditional encryption techniques make data access control and data searching malfunctioning. Recently emerging attribute-based keyword search (ABKS) primitive achieves fine-grained access control and effective data searching over ciphertexts. However, existing ABKS schemes only consider single data owner scenarios and may be an inappropriate choice for IIoT applications, where there exists multiple data owners for an integrated industrial system. Directly extending state-of-the-art single owner schemes to ones for multiowner environment will impose a complicated key management issue. We present an ABKS scheme for multiowners in the cloud-assisted IIoT architecture. By designing a novel master key generation and private key aggregation mechanism with desired communication overheads, our scheme eliminates the complex key management issue in the multiowner model. Formal security proof demonstrates that our scheme is secure against the cloud server. Experimental evaluations also demonstrate its correctness and practicality.**

***Index Terms*—Attribute-based encryption (ABE), attribute-based keyword search (ABKS), cloud computing, Industrial Internet of Things (IIoT), multiple data owners.**

Manuscript received 3 March 2022; revised 16 June 2022; accepted 13 July 2022. Date of publication 19 July 2022; date of current version 22 March 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 61972058 and Grant 61902123, in part by The Natural Science Foundation of Hunan Province, China under Grant 2021JJ30760 and Grant 2021JJ40636, in part by the Key Research Projects of Provincial Education Department of Hunan under Grant 20A041, and in part by the Outstanding Youth Research Project of Provincial Education Department of Hunan under Grant 20B064 and Grant 21B0775. Paper no. TII-22-0925. *(Corresponding author: Yangfan Li.)*

Hui Yin, Hua Deng, and Wei Zhang are with the College of Computer Engineering and Applied Mathematics, Changsha University, Changsha 410022, China (e-mail: yhui@ccsu.edu.cn; hdeng0804@163.com; zhangw@ccsu.edu.cn).

Yangfan Li is with the College of Computer Science and Engineering, Central South University, Changsha 410083, China (e-mail: yangfanli@hnu.edu.cn).

Zheng Qin is with the College of Information Science and Engineering, Hunan University, Changsha 410082, China (e-mail: zqin@hnu.edu.cn).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2022.3192304.

Digital Object Identifier 10.1109/TII.2022.3192304

I. Introduction

NOWADAYS, all kinds of industrial Internet of Things (IIoT) applications have been widely deployed in modern industry systems such as the smart supply chain, smart grids, and 5G-enabled unmanned aerial vehicles. The operations of IIoT systems result in the rapid increase of data. How to store and deal with the high-volume data generated from the resource-constrained IIoT devices every day is an inevitable problem. These data is the valuable wealth to achieve industrial intellectualization and needs to be periodically stored and analyzed over a more powerful platform. Naturally, the cloud computing paradigm can significantly alleviate the storage and computation burden for local IIoT applications. As a result, the cloud-assisted IIoT has become a widely adopted and popular IIoT architecture in modern industrial systems [1], [2].

While the cloud-assisted IIoT architecture can sustain highly available computation and massive storage services for modern industrial systems, data centralization on public and semitrusted cloud server incurs new challenges for data security [3]. Once privacy-sensitive IIoT data is aggregated into the cloud platform, the data would encounter various attacks from either outer or inner attackers. Cryptographic technologies endow an ability to guarantee data confidentiality against malicious attackers [4]. However, traditional encryption techniques make data access control and data searching malfunctioning. In many potential applications, especially in the cloud computing era, data searching is an indispensable function to quickly locate targets from large-scale cloud data, and the access control can prevent the outsourced data from being unauthorizedly accessed. Motivated by the practical requirements, the new cryptographic technique attribute-based keyword search (ABKS) [5]–[9], is proposed based on searchable encryption (SE) [10] and attribute-based encryption (ABE) [11], which enables fine-grained data access control and effective keyword searching over ciphertexts simultaneously.

Currently, existing ABKS schemes only aim at the single data owner scenario, where the sole data owner encrypts outsourced data, as well as plays a role of authority to establish system master key and issue private key for data users. Such schemes may be

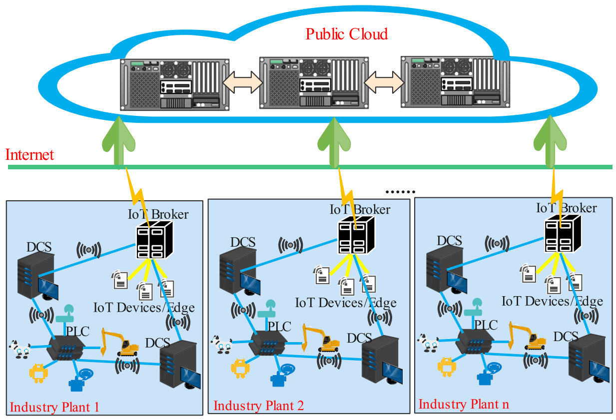

Fig. 1. Architecture for multiple data owners in the cloud-assisted IIoT.

an inappropriate to deploy in the IIoT environments, where there exist multiple data owners for an integrated industrial system. Fig. 1 describes a typical network architecture for multiple data owners (industry plants) in cloud-assisted IIoT. Naively, we can apply the state-of-the-art single owner schemes in the multiple data owners scenario as follows. One solution is to regard all data owners as one and let them share the same master key. However, in real applications, none of the data owners would be willing to share the master keys with others for privacy and data security. On the contrary, they would prefer to use their own master key to encrypt data and generate private key for data users [12]. Nevertheless, the master key independence will bring about heavy private key management burden for each data user in the system, as the data user has to maintain multiple private keys from different data owners. Moreover, the data user needs to submit multiple trapdoors using different private keys to retrieve data. This will cause as direct consequence, high communication and search costs. Directly extending those schemes to the solutions supporting multiple data owners is not trivial, since they are missing a necessary mechanism that allows all data owners to collaboratively generate an aggregated private key for data users in a privacy-preserving manner.

In this article, we construct an ABKS scheme for multiowner in cloud-assisted IIoT based on our proposed master key generation and private key aggregation mechanism. Compared with schemes that directly employ existing ABKS schemes for the multiowner scenario, the advantages can be summarized as: 1) all data owners can use their own master keys to generate an aggregated private key for the data user without sharing master keys each other; 2) a data user needs to maintain only a private key instead of holding multiple ones from different data owners; 3) thanks to the aggregate private key, the data user can use the single trapdoor for each search without submitting multiple ones to cloud server, and the communication and search overheads can be significantly reduced.

Contributions: We design a novel master key generation and private key aggregation mechanism, which allows all data owners to collaboratively aggregate a private key for the data user by only using their own master key. Based on the proposed mechanism, we construct an ABKS scheme for multiowner in cloud-assisted IIoT, by leveraging the expressive and efficient ABE scheme proposed in [13]. To the best of our knowledge, this construction is the first ABKS scheme for multiowner environment in cloud-assisted IIoT. Also, we provide the correctness and formal security proofs for our scheme, and show that it is secure against the cloud server. We implement our scheme in a Java platform, and experimental evaluations demonstrate its correctness and practicality.

II. Related Work

A. Searchable Encryption

SE is an attractive and promising cryptographic primitive, which allows an untrusted server to perform data searching over encrypted data via a specially encoded token. Song *et al.* [10] designed the first practical SE construction in the private-key setting; however, it suffers from the linearly increased search complexity with the size of document set. Curtmola *et al.* [14] significantly improved the search efficiency by employing encrypted inverted index structure so as to achieve optimal sublinear search. In order to support secure data addition and deletion with low communication and computation overheads, the authors of [15] and [16] realized dynamical constructions. Zhang *et al.* [17] declared that the dynamical constructions need forward privacy to resist file injection attacks. The forward privacy schemes [18], [19] can guarantee that prior search tokens cannot be used to search over newly added data in a time period. Due to possible misbehavior of the search server, verifiable SE constructions were proposed in [20]. Those schemes allow the data user to verify the completeness and correctness of query results [21]. The backward privacy is another stronger security notion. It requires that a search token does not reveal information from data that were deleted [22], [23]. We refer to the above schemes as searchable symmetric encryption (SSE), since they were all built in the private-key environment. Generally, the security of SSE schemes are formalized by a group of leakage functions. Mainly relying on the security of pseudorandom functions, they were proved to be secure against the cloud server.

The first public-key encryption with keyword search (PEKS) scheme was proposed in [24]. Compared with SSE, while PEKS schemes sustain more expensive search overhead, they can obtain stronger security and richer functionalities [25]–[27]. Recently, with increasing development of the cloud-assisted IIoT, researchers proposed advanced PEKS schemes in the cloud-assisted IIoT applications such as [1], [28]–[32]. While those schemes did not take data access control into consideration, they provide promising technological approaches to achieve secure data searching in the IIoT applications.

B. ABE and ABKS

ABE is a public-key cryptographic primitive, which possesses an instinctive ability of data sharing by allowing the encryptor to encrypt data only once for a set of users according to an access policy. Any user with credentials satisfying the access policy can recover information from ciphertexts. Sahai and Waters [11] designed the first ABE scheme. To enhance the expressivity

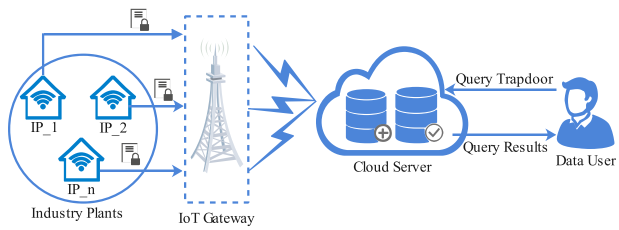

Fig. 2. System model.

of access policy, ABE has been further refined to key-policy ABE (KP-ABE) [33], [34] and ciphertext-policy ABE (CP-ABE) [13], [35]. CP-ABE (KP-ABE, respectively) signifies that the ciphertext (the private key, respectively) associates with the policy, while the user's attribute set (the policy, respectively) is bound to the private key. Whichever scheme is used, the common feature is that the private key is allowed to decrypt the ciphertext, if and only if the attribute set satisfies the access policy.

With the wide deployment of cloud computing applications, how to simultaneously carry out data access control and data searching over encrypted outsourced data has become a hot research point. Motivated by this practical requirement, researchers present a new cryptographic primitive ABKS [5]–[9]. Technically, these schemes achieve data searching and access control over ciphertexts by encrypting an index with an access policy, which specifies who have permission to search the encrypted index. The search process between secure index and query trapdoor is equivalent to a decryption test in original ABE. Moreover, the cloud server is not allowed to obtain any plaintext information of keywords. Since ABKS schemes are the public-key cryptosystem, they were proved to be secure against the cloud server based on some acknowledged difficult problems such as discrete logarithm problem. However, those above schemes only consider the single data owner scenario and are not suitable to be deployed in IIoT applications. Moreover, directly extending those schemes to the solutions supporting multiple data owners is not trivial, since they are missing a necessary mechanism that allows all data owners to collaboratively generate an aggregated private key for data users in a privacy-preserving manner.

III. Problem Formulations

A. System Model

The system model of our scheme involves the following four types of entities, i.e.,

1) many industry plants [36];
2) IoT gateway;
3) cloud server;
4) the data user, as shown in Fig. 2.

There exist multiple industry plants (data owners) IP_1, IP_2, ..., IP_n, who generate a large amount of data from production lines and/or machines. To guarantee data confidentiality, each data owner employs traditional symmetric encryption to encrypt their data, and builds encrypted searchable index as well. Encrypted data along with secure searchable index are uploaded via IoT gateways to the cloud server. A data user submits a query trapdoor (encrypted search query) to the cloud server, who performs the ABKS over secure searchable index. In ABKS schemes, each index keyword is associated with an access policy to specify its search permissions and the query trapdoor is embedded in the data user's attributes. Also, we assume that there exist private communication channels, by which the data owner can send secure parameters to the authorized data user.

B. Security Model

A secure ABKS scheme needs to guarantee that the "honest-but-curious" cloud server cannot obtain any underlying plaintext information from secure index and query trapdoor. To provide a formal security proof for our scheme, we describe the widely used selective security model, which is formalized via the following selective-set game between a probability polynomial-time (PPT) adversary $\mathcal{A}$ and a challenger $\mathcal{B}$.

Setup: $\mathcal{B}$ sets up the public parameter **Para** and sends it to $\mathcal{A}$. $\mathcal{A}$ sends a challenge linear secret sharing scheme (LSSS) [13] access policy (M^*, ρ^*) to $\mathcal{B}$.

Phase 1: $\mathcal{A}$ requests from $\mathcal{B}$ the query trapdoor of keyword $\mathcal{Q}_i (1 \leq i \leq n)$ for polynomially bounded times, where each keyword $\mathcal{Q}_i$ corresponds to a set S_i of attributes. For every request $\mathcal{Q}_i$, $\mathcal{B}$ responds to $\mathcal{A}$ by generating a private key K_i with respect to attribute set S_i and using K_i to establish the query trapdoor $\mathbf{Trap}_{\mathcal{Q}_i}$ of $\mathcal{Q}_i$. The only restriction is that none of the queried sets $\{S_i\}$ satisfy (M^*, ρ^*).

Challenge: $\mathcal{A}$ sends two keywords w_0 and w_1 to $\mathcal{B}$, and $\mathcal{B}$ chooses a random bit $b \in \{0, 1\}$ and encrypts w_b using **Para** and (M^*, ρ^*). The ciphertext $\mathbf{ind}_{w_b}$ is sent to $\mathcal{A}$.

Phase 2: $\mathcal{A}$ continues to request from $\mathcal{B}$ the query trapdoor $\mathbf{Trap}_{\mathcal{Q}_x}$ for any keyword $\mathcal{Q}_x$ corresponding to S_x for polynomially bounded times, as **Phase 1.** The only restriction is that S_x does not satisfy (M^*, ρ^*).

Guess: $\mathcal{A}$ outputs b's guess b'.

For any PPT adversary $\mathcal{A}$, the advantage of $\mathcal{A}$ winning the selective-set game is defined as $Adv = \mathbf{Pr}[b = b'] - \frac{1}{2}$.

IV. Scheme Construction

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic multiplicative groups of prime order p and g denote a generator of $\mathbb{G}_1$. We define a map $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the bilinearity property [13] and two hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$ and $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_p^*$.

A. Public Parameter and Master Key Generation

Public parameter generation: Each data owner $\mathcal{D}_i$ chooses random exponents α_i and a_i, and sends $E_i = e(g,g)^{\alpha_i}, A_i = g^{a_i}$ to all other data owners. On the other hand, when each data owner $\mathcal{D}_i$ receives all E_j's and A_j's, $j \in \{1, .., n\} \setminus \{i\}$, $\mathcal{D}_i$ individually computes

$$E = \prod_{i \in \mathcal{D}} E_i = e(g,g)^{\sum_{i \in \mathcal{D}} \alpha_i}, A = \prod_{i \in \mathcal{D}} A_i = g^{\sum_{i \in \mathcal{D}} a_i}. \quad (1)$$

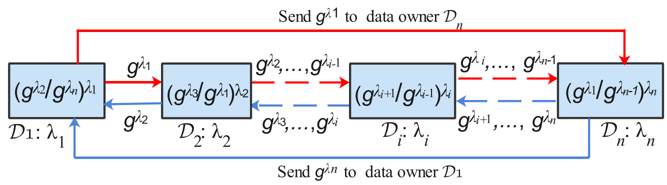

Fig. 3. Master key generation for each data owner.

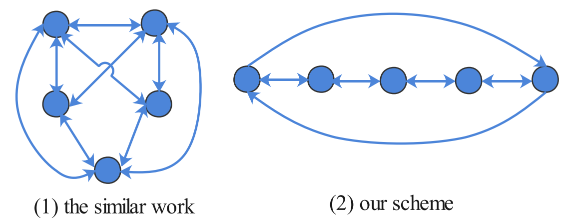

Fig. 4. Comparison between the similar scheme and ours.

Master key generation: Then, each data owner $\mathcal{D}_i, i \in \{2,\ldots,n-1\}$, chooses a random $\lambda_i \in \mathbb{Z}_p^*$ and calculates g^{λ_i}, which is sent to $\mathcal{D}_{i-1}$ and $\mathcal{D}_{i+1}$, respectively. Especially, $\mathcal{D}_1$ sends g^{λ_1} to $\mathcal{D}_n$ and $\mathcal{D}_2$, and $\mathcal{D}_n$ sends g^{λ_n} to $\mathcal{D}_{n-1}$ and $\mathcal{D}_1$.

Finally, each data owner $\mathcal{D}_i, i \in \{2,\ldots,n-1\}$ generates his own master key as

$$\mathbf{MK}_i = \left(a_i, \alpha_i, \beta_i = \left(\frac{g^{\lambda_{i+1}}}{g^{\lambda_{i-1}}}\right)^{\lambda_i}\right). \tag{2}$$

Especially, $\beta_1 = (g^{\lambda_2}/g^{\lambda_n})^{\lambda_1}$ and $\beta_n = (g^{\lambda_1}/g^{\lambda_{n-1}})^{\lambda_n}$. Fig. 3 shows the process of master key component β generation for each data owner. We can see that the process just likes constructing a double circular linked list such that our scheme supports dynamical addition of the new data owner, which is equivalent to inserting a new node after the last node of the linked list. However, when a new data owner joins in the system, the system needs to recalculate the public parameter and aggregated private key for the data user, which needs to pay for extra computation and communication costs.

In this phase, the system public parameter $\mathbf{PK} = \{E, A\}$ is published and data owner $\mathcal{D}_i$ keeps his master key $\mathbf{MK}_i = \{\alpha_i, \beta_i\}$ secret.

B. Private Key Generation

In order to perform an authorized keyword search over encrypted data, a data user $\mathcal{U}$ with attribute set S needs to obtain an aggregated private key from all data owners, by which $\mathcal{U}$ can encrypt a query keyword to generate the corresponding query trapdoor. The key generation process is composed of the following steps.

1) Each data owner $\mathcal{D}_i$ uses his master key to compute $\beta_i(1+\alpha_i \cdot p)$ (here, each data owner uses β_i to guarantee the confidentiality of α_i) and sends it to a specified data owner. After that, the data owner uses these values (including his own) to compute

$$\begin{aligned}
A &= \prod_{i=1}^{n} \beta_i \cdot (1+\alpha_i \cdot p) \\
&= \beta_1 \cdot \beta_n \prod_{i=2}^{n-1} \beta_i \cdot \prod_{i=1}^{n} (1+\alpha_i \cdot p) \\
&= \left(\frac{g^{\lambda_2}}{g^{\lambda_n}}\right)^{\lambda_1} \left(\frac{g^{\lambda_1}}{g^{\lambda_{n-1}}}\right)^{\lambda_n} \prod_{i=2}^{n-1} \left(\frac{g^{\lambda_{i+1}}}{g^{\lambda_{i-1}}}\right)^{\lambda_i} \\
&\quad \cdot \prod_{i=1}^{n} (1+\alpha_i \cdot p) \\
&= g^{\sum_{i=2}^{n-1}(\lambda_{i+1}\lambda_i - \lambda_{i-1}\lambda_i)} \prod_{i=1}^{n} (1+\alpha_i \cdot p) \\
&= \prod_{i=1}^{n} (1+\alpha_i \cdot p), \\
A' &= \prod_{i=1}^{n} (1+\alpha_i \cdot p) \bmod p^2 = 1 + p\sum_{i=1}^{n} \alpha_i, \\
A'' &= (A'-1)/p = \left(1 + p\sum_{i=1}^{n} \alpha_i - 1\right)/p = \sum_{i=1}^{n} \alpha_i
\end{aligned} \tag{3}$$

and compute the private-key component K_1 as

$$K_1 = g^{A''} = g^{\sum_{i=1}^{n} \alpha_i}. \tag{4}$$

2) Each data owner $\mathcal{D}_i$ randomly chooses an exponent $t_i \in \mathbb{Z}_p^*$ and computes $a_i t_i$ and $\beta_i \cdot (1+t_i \cdot p)$. $a_i t_i$ and $\beta_i \cdot (1+t_i \cdot p)$ are sent to a specified data owner. After that, the data owner uses these values (including his own) to compute

$$\begin{aligned}
&C = \prod_{i=1}^{n} a_i t_i = \sum_{i=1}^{n} a_i t_i, K_2 = g^C = g^{\sum_{i=1}^{n} a_i t_i}; \\
&D = \prod_{i=1}^{n} \beta_i \cdot (1+t_i \cdot p) \bmod p^2, K_3 \\
&\quad = g^{(D-1)/p} = g^{\sum_{i=1}^{n} t_i}; \\
&\forall s \in S \;\; K_s = H_1(s)^{(D-1)/p} = H_1(s)^{\sum_{i=1}^{n} t_i}.
\end{aligned} \tag{5}$$

3) The data owner sends $\{K_1, K_2, K_3, \{K_s\}\}$ to $\mathcal{U}$ via secure communication channels.

Remarks: We found a similar approach to generate the master key and aggregate the private key in multiauthority circumstances [37]. However, in that scheme, every authority must have one interaction with all other authorities when generating his own master key. Thus, the total number of communications is $n(n-1)$. Our scheme requires a data owner (authority) to interact with only his *neighbors* one time, and the total number of communications are reduced to $2n$. Fig. 4 shows a comparison of interaction processes between the similar work in [37] and our

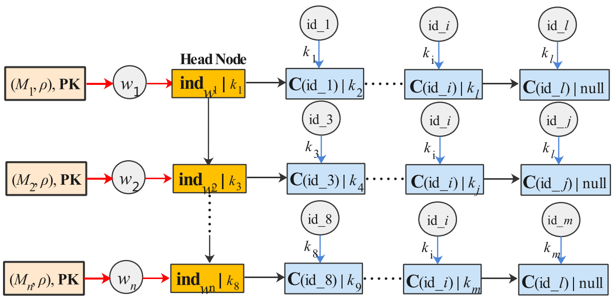

Fig. 5. Secure searchable index construction.

scheme. Our design is meaningful, because it can not only reduce communication costs, but also decrease the probability to incur errors during communications, especially when the number n of data owners is large.

C. Index Keyword Encryption

To conduct secure and authorized keyword search over ciphertexts, a data owner $\mathcal{D}$ uses system public parameter **PK** to encrypt an index keyword w by specifying an LSSS access policy [13], which determines who have access to data files containing the index keyword. The encryption process is as follows.

1) $\mathcal{D}$ defines an LSSS access policy (M, ρ) for index keyword w, where M is a matrix with l rows and n columns and each row $1 \le i \le l$ denotes an attribute in the access policy.
2) $\mathcal{D}$ chooses a secret value $\theta \in \mathbb{Z}_p^*$ and $n-1$ random values $y_2, \ldots, y_n$, which make up a random vector $\vec{v} = (\theta, y_2, \ldots, y_n)$. For each row M_i of matrix M, we calculate the inner product $\gamma_x = \vec{v} \cdot M_x$.
3) $\mathcal{D}$ uses **PK** to encrypt w as

$$\begin{aligned}\mathbf{ind}_w &= (\mathcal{I}_1 = e(g,g)^{\sum_{i\in\mathcal{D}}\alpha_i H_2(w)\theta}, \mathcal{I}_2 = g^{\theta},\\ &\forall x \in [1,l]) : \mathcal{I}_x = g^{\sum_{i\in\mathcal{D}} a_i\gamma_x} H_1(\rho(x))^{-\theta}).\end{aligned} \quad (6)$$

We use inverted index construction to realize our searchable secure index, as shown in Fig. 5. The whole secure index consists of n posted lists and each node contains two fields, i.e., 1) the ciphertext field; and 2) the symmetric key field, where each head node involves an encrypted index keyword, and other nodes include encrypted identifiers of data containing the index keyword. The arrows in red denote to encrypt the index keywords and the arrows in blue denote symmetric encryptions of file identifiers. Having encrypted index keywords (head nodes), it is easy to implement this attribute-based inverted index by adapting the algorithm framework proposed in [14].

D. Trapdoor Generation

When a data user $\mathcal{U}$ with attribute set S receives the joint key $\{K_1, K_2, K_3, \{K_s\}_{s\in S}\}$, $\mathcal{U}$ is allowed to encrypt a query keyword $\mathcal{Q}$ to obtain a trapdoor $\mathbf{Trap}_{\mathcal{Q}}$ by the following steps.

1) $\mathcal{U}$ chooses random $\delta \in \mathbb{Z}_p^*$ and calculates $A = K_1(K_2)^{\delta}$, $B = (K_3)^{\delta}$, and $\{C_s = (K_s)^{\delta}\}_{s\in S}$.
2) $\mathcal{U}$ computes $H_2(\mathcal{Q})$ and encrypts the hash value as

$$\begin{aligned}T_1 &= A^{H_2(\mathcal{Q})} = g^{\sum\alpha_i H_2(\mathcal{Q})} g^{\sum a_i t_i \delta H_2(\mathcal{Q})};\\ T_2 &= B^{H_2(\mathcal{Q})} = g^{\sum t_i\delta H_2(\mathcal{Q})};\\ \forall s \in S\ \ T_s &= C_s^{H_2(\mathcal{Q})} = H_1(s)^{\sum t_i\delta H_2(\mathcal{Q})}.\end{aligned} \quad (7)$$

The query trapdoor of query keyword $\mathcal{Q}$ is denoted as $\mathbf{Trap}_{\mathcal{Q}} = (T_1, T_2, \{T_s\}_{s\in S})$.

By introducing the random exponent δ, we can achieve trapdoor unlinkability property (identical query keywords have different ciphertexts), which are not indispensable to generate the query trapdoor.

E. Secure Search

When a data user $\mathcal{U}$ submits a query trapdoor $\mathbf{Trap}_{\mathcal{Q}} = (T_1, T_2, \{T_s\}_{s\in S})$, the cloud server will use it to perform a content-unaware linear search over the secure searchable index. Therefore, the search has to go through the whole secure index in the worst case or returns the encrypted goal data when encountering a successful match. In the whole search process, except the search results, the cloud server cannot gain anything else. Specifically, given a trapdoor $\mathbf{Trap}_{\mathcal{Q}}$ and a current encrypted index keyword $\mathbf{ind}_w$, the cloud server will conduct the following match between $\mathbf{Trap}_{\mathcal{Q}}$ and $\mathbf{ind}_w$.

1) *Permission check:* If S in $\mathbf{Trap}_{\mathcal{Q}}$ does not satisfy (M, ρ) associated with $\mathbf{ind}_w$, the search process will proceed as step 1) with next encrypted index keyword. If S satisfies (M, ρ), the algorithm performs step 2).
2) *Keyword match:* Define a subset of $\{1, 2, \ldots, l\}$ as $X = \{x : \rho(x) \in S\}$. Based on X and M, the search algorithm calculates a set of constants $\{\chi_i\}_{i\in X}$ such that $\sum_{i\in X}\chi_i M_i = (1, 0, \ldots, 0)$, where M_i is the ith row of LSSS matrix M. Next, it uses $\mathbf{Trap}_{\mathcal{Q}}$ and $\mathbf{ind}_w$ to test whether the following equation is true or not:

$$\mathcal{I}_1 \stackrel{?}{=} e(\mathcal{I}_2, T_1) \Big/ \left(\prod_{x\in X} (e(\mathcal{I}_x, T_2) e(\mathcal{I}_2, T_{\rho(x)}))^{\chi_x} \right). \quad (8)$$

If the above equation is true, the algorithm returns the data files containing keyword w to $\mathcal{U}$; otherwise, the search algorithm skips to step 1).

F. Correctness Proof

Theorem 1: If attribute set S in $\mathbf{Trap}_{\mathcal{Q}}$ satisfies LSSS matrix (M, ρ) associated with $\mathbf{Ind}_w$ and $w = \mathcal{Q}$, then (5) holds and our proposed search algorithm can return correct query results.

Proof: According to (5), the search algorithm first calculates

$$\begin{aligned}e((\mathcal{I}_2, T_1) &= e(g^{\theta}, g^{\sum\alpha_i H_2(\mathcal{Q})} g^{\sum a_i t_i\delta H_2(\mathcal{Q})})\\ &= e(g^{\theta}, g^{\sum\alpha_i H_2(\mathcal{Q})}) e(g^{\theta}, g^{\sum a_i t_i \delta H_2(\mathcal{Q})})\\ &= e(g,g)^{\theta\sum\alpha_i H_2(\mathcal{Q})} e(g,g)^{\theta\sum a_i t_i\delta H_2(\mathcal{Q})}.\end{aligned} \quad (9)$$

On the other hand, having the knowledge of LSSS, we know that if S satisfies (M, ρ), the search algorithm can find a subset $X = \{x : \rho(x) \in S\} \subset \{1, 2, \ldots, l\}$ and a group of constants

$\{\chi_x\}_{x\in X}$ such that $\Sigma_{x\in X}\chi_x M_x = (1, 0, \ldots, 0)$. Next, for each $x \in X$, the algorithm calculates

$$\begin{aligned} e(\mathcal{I}_x, T_2) &= e\left(g^{\sum a_i \gamma_x} H_1(\rho(x))^{-\theta}, g^{\sum t_i \delta H_2(\mathcal{Q})}\right) \\ &= e\left(g^{\sum a_i \gamma_x}, g^{\sum t_i \delta H_2(\mathcal{Q})}\right) e \\ &\quad \times \left(H_1(\rho(x))^{-\theta}, g^{\sum_{i\in\mathcal{D}} t_i \delta H_2(\mathcal{Q})}\right) \\ &= e\left(g^{\sum a_i \gamma_x}, g^{\sum t_i \delta H_2(\mathcal{Q})}\right) e \\ &\quad \times (H_1(\rho(x)), g)^{-\theta \sum t_i \delta H_2(\mathcal{Q})} \end{aligned} \tag{10}$$

and

$$\begin{aligned} e(\mathcal{I}_2, T_{\rho(x)}) &= e\left(g^{\theta}, H_1(\rho(x))^{\sum t_i \delta H_2(\mathcal{Q})}\right) \\ &= e\left(H_1(\rho(x)), g\right)^{\theta \sum t_i \delta H_2(\mathcal{Q})}. \end{aligned} \tag{11}$$

Therefore, we have

$$\begin{aligned} &\prod_{x\in X}\left(e\left(\mathcal{I}_x, T_2\right) e\left(\mathcal{I}_2, T_{\rho(x)}\right)\right)^{\chi_x} \\ &= \prod_{x\in X}\left(e\left(g^{\sum a_j \gamma_x}, g^{\sum t_i \delta H_2(\mathcal{Q})}\right)\cdot\right. \\ &\left. e\left(H_1(\rho(x)), g\right)^{-\theta \sum t_i \delta H_2(\mathcal{Q})} e\left(H_1(\rho(x)), g\right)^{\theta \sum t_i \delta H_2(\mathcal{Q})}\right)^{\chi_x} \\ &= \prod_{i\in I}\left(e\left(g^{\sum a_j \gamma_x}, g^{\sum t_i \delta H_2(\mathcal{Q})}\right)\right)^{\chi_x} \\ &= e\left(g^{\sum a_i}, g^{\sum t_i \delta H_2(\mathcal{Q})}\right)^{\sum_{x\in X} \gamma_x \chi_x} \\ &= e\left(g^{\sum a_i}, g^{\sum t_i \delta H_2(\mathcal{Q})}\right)^{\sum_{x\in X} \vec{v} M_x \chi_x} \\ &= e(g, g)^{\theta \sum a_i t_i \delta H_2(\mathcal{Q})}. \end{aligned} \tag{12}$$

The search algorithm further calculates

$$\begin{aligned} &e(\mathcal{I}_2, T_1) \Big/ \left(\prod_{x\in X}\left(e(\mathcal{I}_x, T_2) e(\mathcal{I}_2, T_{\rho(x)})\right)^{\chi_x}\right) \\ &= \frac{e(g,g)^{\theta \sum \alpha_i H_2(\mathcal{Q})} e(g,g)^{\theta \sum a_i t_i \delta H_2(\mathcal{Q})}}{e(g,g)^{\theta \sum a_i t_i \delta H_2(\mathcal{Q})}} \\ &= e(g,g)^{\theta \sum \alpha_i H_2(\mathcal{Q})}. \end{aligned} \tag{13}$$

If the underlying index keyword w in $\mathbf{Ind}_w$ is identical to the query keyword $\mathcal{Q}$ in $\mathbf{Trap}_{\mathcal{Q}}$, then

$$\mathcal{I}_1 = e(g,g)^{\theta \sum \alpha_i H_2(w)} = e(g,g)^{\theta \sum \alpha_i H_2(\mathcal{Q})}. \tag{14}$$

We complete the correctness proof.

G. Security Proof

We will adopt the reduction idea to prove the security of our scheme. Loosely speaking, proving our scheme's security will be reduced to try to solve the decisional q-bilinear Diffie-Hellman exponent (BDHE) problem [13]. However, the decisional q-BDHE problem is acknowledgedly intractable in the polynomial time. Here, we review the decisional q-BDHE problem/assumption as follows.

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic groups of prime order p, and g denote a generator of $\mathbb{G}_1$. Randomly choose two elements a and θ from $\mathbb{Z}_p^*$. The decisional q-BDHE problem is to distinguish element $R = e(g,g)^{a^{q+1}\theta}$ from a random element $\widehat{R}$ when given $t = (g, g^{\theta}, g^{a}, \ldots, g^{a^q}, g^{a^{q+2}}, \ldots, g^{a^{2q}})$, where e denotes the bilinear pairing map and $R, \widehat{R} \in \mathbb{Z}_p^*$.

We define that an adversary has advantage ϵ in solving decisional q-BDHE problem if

$$\begin{aligned} &\left|\Pr\left[\mathcal{A}(t, R = e(g,g)^{a^{q+1}\theta}) = 0\right]\right. \\ &\left. - \Pr\left[\mathcal{A}(t, Q = \widehat{R}) = 0\right]\right| \geq \epsilon. \end{aligned} \tag{15}$$

Decisional q-BDHE problem difficulty assumption: We say the decisional q-BDHE problem is intractable since no a polynomial time algorithm is able to solve it with a nonnegligible advantage.

Theorem 2: If the decisional *q-BDHE* assumption [13] holds, our scheme is secure and no polynomial-time adversary can selectively break our scheme with a challenge access policy (M^*, ρ^*), where M^* is an $l^* \times n^*$ matrix and $n^* \leq q$. In our scheme, H_1 is modeled as a random oracle and H_2 is instantiated as a cryptology hash function.

Proof: Intuitively, the adversary $\mathcal{A}$ wishes to distinguish $e(g,g)^{\Sigma_{i\in\mathcal{D}}\alpha_i H_2(w_0)s}$ from $e(g,g)^{\Sigma_{i\in\mathcal{D}}\alpha_i H_2(w_1)s}$ with an advantage ε to break through our scheme. We will prove that if the advantage ε is nonnegligible, then there exists a simulator $\mathcal{S}$ that can solve the decisional q-BDHE problem with the nonnegligible advantage $\varepsilon/2$.

Setup: $\mathcal{S}$ first chooses two random $\alpha_i^*, a_i^* \in \mathbb{Z}_p^*$ and calculates the public parameters $E = e(g,g)^{\sum_{i=1}^n \alpha_i^*}$ and $A = g^{\sum_{i=1}^n a_i^*}$. Then, $\mathcal{S}$ sets $\alpha' = \sum_{i=1}^n \alpha_i^* - (\sum_{i=1}^n a_i^*)^{q+1}$. Since α_i and a_i are random, α' is also a random element from $\mathcal{A}$'s point of view. For simplicity, we write $\sum_{i=1}^n \alpha_i^*$ and $\sum_{i=1}^n a_i^*$ as α and a in the following description, respectively, and then $\alpha' = \alpha - a^{q+1}$. Next, $\mathcal{S}$ chooses a random $\theta \in \mathbb{Z}_p^*$ and generates the q-BDHE challenge $R = e(g,g)^{a^{q+1}\theta}$ and $t = (g, g^{\theta}, g^{a}, \ldots, g^{a^q}, g^{a^{q+2}}, \ldots, g^{a^2 q})$. Finally, $\mathcal{A}$ sends a challenge policy (M^*, ρ^*) to $\mathcal{S}$, where M^* is an $l^* \times n^*$ matrix and $n^* \leq q$.

In our proof, H_1 is modeled as the random oracle, and we describe how $\mathcal{S}$ responds $\mathcal{A}$'s query to $H_1(s)$ by maintaining a response table, where s denotes an attribute $\mathcal{A}$ wants to query, as follows.

1) If s has never been queried before and we can find an i in (M^*, ρ^*) such that $\rho^*(i) = s$, then $\mathcal{S}$ sets

$$H_1(s) = g^{r_s} \times g^{a M_{i,1}^*} \times g^{a^2 M_{i,2}^*} \times \cdots \times g^{a^{n^*} M_{i,n^*}^*}$$

where r_s is randomly chosen from $\mathbb{Z}_p^*$; if, thus, i does not exist in (M^*, ρ^*), $H_1(s)$ is simply set to be $H_1(s) = g^{r_s}$. Finally, that value $H_1(s)$ is added into a global response table.

2) If s has been queried before, then, $\mathcal{S}$ obtains $H(s)$ from the global response table and sends it to $\mathcal{A}$.

Phase 1: In this phase, $\mathcal{A}$ from the simulator $\mathcal{S}$ queries trapdoors of keywords for polynomially bounded times. We describe the process $\mathcal{A}$ responds a query by an example as follows.

We use $\mathcal{Q}$ to denote a query keyword. To generate the trapdoor of $\mathcal{Q}$, $\mathcal{S}$ has to first simulate the private key with respect to an attribute set S. We describe this simulation process as follows.

As S does not satisfy (M^*, ρ^*), $\mathcal{S}$ must be able to find a vector $\vec{h} = (h_1, \ldots h_{n^*})$ such that $h_1 = -1$ and $\vec{h} \cdot M_i^* = 0$, where $\rho^*(i) \in S$ and M_i^* is the ith row of M^*.

Simulating K_1^*: Since both $\sum_{i=1}^{n} \alpha_i$ in K_1 and $\alpha' = \alpha - a^{q+1}$ are random from the adversary's point of view, in this simulation $\mathcal{S}$ simply sets

$$K_1^* = g^{\alpha'} = g^{\alpha - a^{q+1}} \tag{16}$$

where $\alpha = \sum_{i=1}^{n} \alpha_i^*$ and $a = \sum_{i=1}^{n} a_i^*$.

Since simulating K_2^* needs to a simulation of $\sum_{i=1}^{n} t_i$ produced in K_3 simulation process, here we have to first simulate K_3^*, followed by K_2^*.

Simulating K_3^*: $\mathcal{S}$ first chooses random $r \in \mathbb{Z}_p^*$ and calculates

$$g^r \prod_{i=1}^{n^*} (g^{a^{q+1-i}})^{h_i} = g^{r + \sum_{i=1}^{n^*} (a^{q+1-i}) h_i}. \tag{17}$$

Since both $\sum_{i=1}^{n} t_i$ in K_3 and r are random from the adversary's point of view, $\mathcal{S}$ can implicitly set

$$\sum_{i=1}^{n} t_i = r + \sum_{i=1}^{n^*} (a^{q+1-i}) h_i. \tag{18}$$

Thus, $\mathcal{S}$ simulates the private component K_3 as

$$K_3^* = g^{\sum_{i=1}^{n} t_i} = g^{r + \sum_{i=1}^{n^*} (a^{q+1-i}) h_i}, \tag{19}$$

where $a = \sum_{i=1}^{n} a_i^*$.

Simulating K_2^*: Since both $\sum_{i=1}^{n} a_i t_i$ in K_2 and $\sum_{i=1}^{n} a_i^*, \sum_{i=1}^{n} t_i$ are random from the adversary's point of view, $\mathcal{S}$ can use the value $\sum_{i=1}^{n} a_i^* \sum_{i=1}^{n} t_i$ to simulate K_2. For ease of writing, let $t = \sum_{i=1}^{n} t_i = r + \sum_{i=1}^{n^*} (a^{q+1-i}) h_i$. That is, $\mathcal{S}$ employs values $a = \sum_{i=1}^{n} a_i^*$ and t to simulate K_2 as

$$\begin{aligned} K_2^* &= g^{a\left(r + \sum_{i=2}^{n^*} (a^{q+1-i}) h_i\right)} \\ &= g^{ar} \prod_{i=2}^{n^*} \left(g^{a^{q+2-i}}\right)^{h_i}. \end{aligned} \tag{20}$$

Obviously, K_2^* does not contain the term $g^{ar + aa^{(q+1-i)} h_i} = g^{ar} g^{-a^{q+1}}$, where $i = 1$ and $h_1 = -1$. This is because that when generating the trapdoor of a query keyword, our trapdoor generation algorithm requires a combination of K_1^* and K_2^*, and $K_1^* = g^{\alpha'} = g^{\alpha - a^{q+1}}$ exactly contains term $g^{-a^{q+1}}$.

Simulating K_s^* $\forall s \in S$: There are two cases when simulating private component K_s^* for attribute $s \in S$. The first case is that if there is no i in (M^*, ρ^*) such that $\rho(i)^* = s$, the simulator $\mathcal{S}$ simply sets

$$K_s^* = (K_3^*)^{r_s} = g^{r r_s + \sum_{i=1}^{n^*} (a^{q+1-i}) h_i r_s} \tag{21}$$

where r_s is randomly chosen when responding the query $H_1(s)$ in **Setup** phase. The second case is to create K_s^* for which there exists an i in (M^*, ρ^*) such that $\rho(i)^* = s$. This case is slightly complex as we must guarantee that K_s^* does not contain the term of the form $g^{a^{q+1}}$ that we cannot simulate. To achieve this, $\mathcal{S}$ creates K_s^* as

$$K_s^* = (K_3^*)^{r_s} \prod_{j=1}^{n^*} \left(g^{a^j r} \prod_{k=1, k \neq j}^{n^*} (g^{a^{q+1+j-k}})^{h_k} \right)^{M_{i,j}^*}. \tag{22}$$

The fact is that everything with a^{q+1} in the exponent cancels when combined since $\vec{h} \cdot M_i^* = 0$.

Simulating trapdoor $\mathbf{Trap}_{\mathcal{Q}}$: With K_1^*, K_2^*, K_3^*, and $\{K_s^*\}_{s \in S}$, $\mathcal{S}$ returns the query trapdoor of $\mathcal{Q}$ as

$$\begin{aligned} &(T_1^* = (K_1^* K_2^*)^{H_2(\mathcal{Q})}, T_2^* = (K_3^*)^{H_2(\mathcal{Q})}, \\ &\forall s \in S \;\; T_s^* = (K_s^*)^{H_2(\mathcal{Q})}). \end{aligned} \tag{23}$$

Here, we do not introduce the random exponent δ in K_2^*, K_3^*, K_s^* like we do in the original scheme, to construct the correct exponent $ar - a^{q+1} + a^q h_2 +, \ldots, a^{q+2-n^*} h_{n^*}$ in $K_1^* K_2^*$. However, this does not affect the randomization of the simulated trapdoor since K_2^*, K_3^*, and K_s^* all contain the random exponent r.

Challenge: $\mathcal{A}$ sends two keywords w_0 and w_1 to $\mathcal{B}$, and $\mathcal{B}$ chooses a random bit $b \in \{0, 1\}$. $\mathcal{B}$ first simulates the ciphertext components $\mathcal{I}_1$ and $\mathcal{I}_2$ as

$$\mathcal{I}_1 = R^{H_2(w_b)} e(g^\theta, g^{\alpha'})^{H_2(w_b)}, \mathcal{I}_2 = g^\theta. \tag{24}$$

Then, $\mathcal{S}$ needs to build the ciphertext components $\{\mathcal{I}_x\}_{x \in [1, l^*]}$. Recall that, for a query to random oracle H_1 on input $\rho^*(x)$, $\mathcal{B}$ responds $H_1(\rho^*(x))$ as $H_1(\rho^*(x)) = g^{r_{\rho^*(x)}} \times g^{a M_{i,1}^*} \times g^{a^2 M_{i,2}^*} \times \cdots \times g^{a^{n^*} M_{i,n^*}^*}$. Obviously, $H_1(\rho^*(x))^\theta$ contains terms of the form $g^{a^j \theta}$ that we cannot simulate. In order to cancel out these terms, $\mathcal{B}$ uses the secret splitting vector as

$$\vec{v}^* = (\theta, \theta a + y_2,' \theta a^2 + y_3,' \ldots, \theta a^{n^*-1} + y_{n^*}'), \tag{25}$$

and generates the ciphertext components as

$$\forall x \in [1, l^*]) : \mathcal{I}_x = \left(\prod_{j=2}^{n^*} (g^a)^{M_{i,j}^* y_j'} \right) (g^\theta)^{-r_{\rho^*(x)}}. \tag{26}$$

Phase 2: $\mathcal{A}$ continues to request from $\mathcal{S}$ query trapdoors for polynomially bounded times, as **Phase 1.**

Guess: $\mathcal{A}$ outputs a guess b' of b. $\mathcal{S}$ then outputs 0 to guess that $R = e(g,g)^{a^{q+1}\theta}$, if $b = b'$. According to the simulated ciphertext

$$\begin{aligned} \mathcal{I}_1 &= R^{H_2(w_b)} e(g^\theta, g^{\alpha'})^{H_2(w_b)} \\ &= e(g,g)^{a^{q+1}\theta H_2(w_b)} e(g^\theta, g^{\alpha - a^{q+1}})^{H_2(w_b)} \\ &= e(g,g)^{\alpha\theta H_2(w_b)}, \end{aligned} \tag{27}$$

this is a legal ciphertext of w_b. Since $\mathcal{A}$ can break through our scheme with the nonnegligible advantage ε, the advantage that $\mathcal{S}$ solves the decisional q-BDHE problem is

$$\Pr\left[\mathcal{S}(t, R = e(g,g)^{a^{q+1}\theta}) = 0\right] = \frac{1}{2} + \epsilon. \tag{28}$$

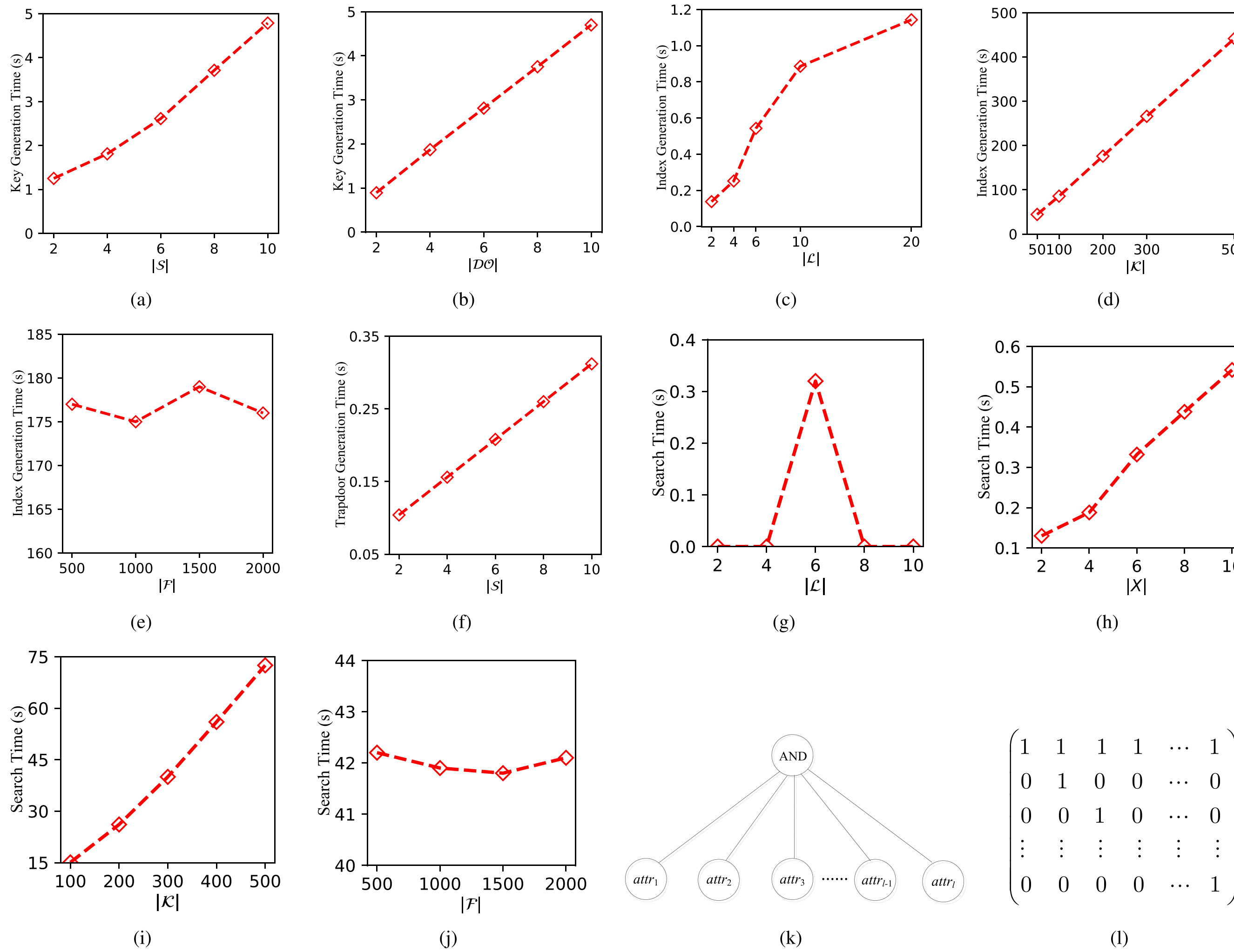

Fig. 6. Performance evaluations of our scheme.

TABLE I
NOTATIONS FOR EXPERIMENTAL EVALUATION

Notation	Description
$\mathcal{DO}$	Set of data owners
$\mathcal{F}$	Set of data files
$\mathcal{K}$	Set of index keywords
$\mathcal{L}$	Set of attributes in the access policy
$\mathcal{S}$	Data user's attribute set
X	Least attribute set satisfying the access policy

On the other hand, if $\mathcal{S}$ outputs 1 to indicate R is a random element, then $\mathcal{I}_1$ is also a random element from $\mathcal{A}$'s point of view and contains no information about w_b, we have

$$\Pr\left[\mathcal{S}(t, R = \widehat{R}) = 0\right] = \frac{1}{2} \tag{29}$$

where $\widehat{R}$ denotes a random value in $\mathbb{Z}_p^*$.

Therefore, $\mathcal{S}$ is able to solve the decisional q-BDHE problem with the nonnegligible advantage $\frac{1}{2} \cdot (\frac{1}{2} + \epsilon) + \frac{1}{2} \cdot \frac{1}{2} = \frac{\epsilon}{2}$, which contradicts with decisional q-BDHE assumption. ■

V. Experimental Evaluation

We implement our scheme in Java platform by employing Java pairing-based cryptography library. In the experimental implementation, we choose *Type A* pairing, which is constructed on the bilinear curve $y^2 = x^3 + x$. We choose 2000 data files from the real-world Enron email dataset and conduct all experimental evaluations in a Windows 7 system with 3.60 GHZ inter Core i7-7700 CPU, 16 GB memory. Table I defines several notations used to describe the data set in our experiments.

Fig. 6(a) and (b) demonstrate the running time of private-key generation in different experimental parameters. We can observe that the private-key generation time is affected by the size of the data user's attribute set and the number of data owners in the system. In Fig. 6(a), when fixing the number of data owners to be 10 ($|\mathcal{DO}| = 10$), the private key generation time linearly increases with the number of attributes. For example, when setting $|\mathcal{S}| = 10$, it needs to spend about 4.8 s generating the private key for a specified data user. On the other hand, in Fig. 6(b), we set the size of attribute set to be 10 and vary the number of data owners. The experimental results show that the more data owners are in the system, the more time costs are required to generate a private key. When setting $|\mathcal{S}| = 10$ and $|\mathcal{DO}| = 10$, the time cost spent on private-key generation is about 4.6 s.

Fig. 6(c)–(e) illustrate the index generation overhead with different experimental parameters. In Fig. 6(c), when varying the number of attributes in the access policy, we evaluate the time cost of encrypting one index keyword. The experimental results demonstrate that the index keyword encryption time linearly grows with the number of attributes, and when setting $|\mathcal{L}| = 20$, encrypting one index keyword needs to consume about 1.1 s in our machine. We extract 500 index keywords from 2000 Enron email data and use them to construct secure searchable index as shown in Fig. 5. In this experiment, we employ an access policy of form "$attr_1$ AND $attr_2$... AND $attr_l$" [this Boolean formula form can be presented as an access tree as shown in Fig. 6(k) and utilize the standard technique proposed in [38] to convert it to the corresponding LSSS matrix as shown in Fig. 6(l)] to encrypt each index keywords, where $attr_i$ denotes an attribute value and l is set to be 10. Fig. 6(d) shows the time cost of building secure index when fixing the size of data file set to be 2000 ($|\mathcal{F}| = 2000$) and varying the size of index keyword set. When the number of index keywords achieves 500, constructing the searchable secure index needs to expend about 442 s. Fig. 6(e) shows that the size of data file set has no influence on index generation time. This is a desirable feature as the size of data file set is usually much larger than the size of keyword set in a practical application.

Fig. 6(f) demonstrates query trapdoor generation time in our scheme when varying the number of attributes. We can observe that with a linearly increasing number of attributes, the trapdoor generation time also linearly increases. When setting $|\mathcal{S}| = 10$, generating a query trapdoor needs to consume about 0.32 s, which is efficient for a data user to encrypt a query keyword.

In order to illustrate the influence of the number of attributes on search time cost, specially, we conduct an equality match experiment over a specified index keyword and a query keyword (in this test, let the index keyword be equal to the query keyword and the attributes in query keyword satisfy the access policy in index keyword). Fig. 6(g) and (h) show the time cost of the equality match when changing the size of attribute sets $\mathcal{L}$ and X. In Fig. 6(g), an access policy "$attr_1$ AND $attr_2$... AND $attr_l$" is used to encrypted the index keyword and an attribute set "$(attr_1, attr_2, \ldots, attr_6)$" is used to generate the query trapdoor, and we vary $l = 2, 4, 6, 8, 10$ and fix the query trapdoor to conduct five groups of search experiment. On the other hand, in Fig. 6(h), we vary access policy "$attr_1$ AND $attr_2$... AND $attr_l$" in the index keyword and the attribute set "$(attr_1, attr_2, \ldots, attr_l)$" in query trapdoor to be $l = 2, 4, 6, 8, 10$ (due to the usage of only "AND" policy, this attribute set is also the least attribute subset X satisfying the access policy). The experimental results show that the search time is not affected by the number of attributes in access policy, but grows linearly with the size of set X. In the real system, the practical search time cost would be less than our experimental results since access policies usually involve other gates such as "OR" (i.e., $|X| < |\mathcal{L}|$). Further, we run the secure search algorithm on our real dataset to verify the average of the search overhead of our scheme. For the ease of evaluation, we let all index keywords ($|\mathcal{K}| = 500$) be encrypted under the identical access policy "$attr_1$ AND $attr_2$... AND $attr_{10}$" [the corresponding LSSS matrix is shown in Fig. 6(l) and the number of rows of this matrix is set to be 10] and all used query keywords to be encrypted under the same attribute set ["$(attr_1, attr_2, \ldots, attr_{10})$"], i.e., $|\mathcal{L}| = |X| = 10$. From the experimental results shown in Fig. 6(i) and (j), we can observe that the search time cost of our scheme is linear to the number of index keywords and is not affected by the size of data file set [in Fig. 6(i), the number of data files is set as 2000 ($|\mathcal{F}| = 2000$) and in Fig. 6(j), we fix the number of index keywords to be 300 ($|\mathcal{K}|$=300)]. Due to $|\mathcal{K}| \ll |\mathcal{F}|$, our scheme has practical search efficiency in the real system.

VI. Conclusion

In this article, we investigated ABKS scheme for multiple data owners in the cloud-assisted IIoT. On the one hand, since extending existing single owner ABKS schemes to multiowner ones will lead to the complicated key management issue, we designed a novel master key generation mechanism that allowed multiple data owners to collaboratively aggregate the private key for an authorized data user with the desired communication costs. On the other hand, we instantiated an ABKS scheme for multiple data owners in cloud-assisted IIoT systems, by utilizing the efficient ABE construction proposed by Waters. We provided the formal security proof and showed our scheme was semantically secure against the cloud server. Also, we conducted a group of experimental evaluations, and the results demonstrated its correctness and practicality. In addition, like all existing ABKS schemes, our scheme only considered the static dataset, and we will explore the dynamical ABKS scheme for multiple data owners in our future work.

Acknowledgment

The authors would like to express their gratitude to the anonymous reviewers whose constructive comments have helped to improve the quality of the manuscript.

References

[1] W. Wang, P. Xu, D. Liu, L. Yang, W. Wang, and Z. Yan, "Lightweighted secure searching over public-key ciphertexts for edge-cloud-assisted industrial IoT devices," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4221–4230, Jun. 2020.

[2] S. Qi, Y. Lu, W. Wei, and X. Chen, "Efficient data access control with fine-grained data protection in cloud-assisted IIoT," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2886–2899, Feb. 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. Financial Cryptogr. Data Secur.*, 2010, pp. 136–149.

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

[6] Y. Miao, J. Ma, X. Liu, X. Li, Z. Liu, and H. Li, "Practical attribute-based multi-keyword search scheme in mobile crowdsourcing," *IEEE Internet Things J.*, vol. 5, no. 4, pp. 3008–3018, 2018.

[7] 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, 2020.
[8] Y. Miao et al., "Attribute-based keyword search over hierarchical data in cloud computing," *IEEE Trans. Serv. Comput.*, vol. 13, no. 6, pp. 985–998, 2020.
[9] H. Wang, X. Dong, and Z. Cao, "Multi-value-independent ciphertext-policy attribute based encryption with fast keyword search," *IEEE Trans. Serv. Comput.*, vol. 13, pp. 1142–1151, 2020.
[10] D. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proc. IEEE Symp. Secur. Privacy*, 2000, pp. 44–55.
[11] A. Sahai and B. Waters, "Fuzzy identity-base encryption," in *Proc. EUROCRYPT*, 2005, pp. 457–473.
[12] W. Zhang, Y. Lin, S. Xiao, J. Wu, and S. Zhou, "Privacy preserving ranked multi-keyword search for multiple data owners in cloud computing," *IEEE Trans. Comput.*, vol. 65, no. 5, pp. 1566–1577, 2016.
[13] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Public Key Cryptography*, 2011, pp. 53–70.
[14] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, "Searchable symmetric encryption: Improved definitions and efficient constructions," in *Proc. ACM Conf. Comput. Commun. Secur.*, vol. 19, 2006, pp. 79–88.
[15] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmeteric encryption," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 965–976.
[16] E. Stefanov, C. Papamanthou, and E. Shi, "Practical dynamic searchable encryption with small leakage," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2014, pp. 1–15.
[17] 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 Secur. Symp.*, 2016, pp. 707–720.
[18] R. Bost, "$\sigma o \varphi o \varsigma$: Forward secure searchable encryption," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2016, pp. 1143–1154.
[19] K. S. Kim, M. Kim, D. Lee, J. H. Park, and W. Kim, "Forward secure dynamic searchable symmetric encryption with efficient updates," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2017, pp. 1449–1463.
[20] M. S. Mohamad and G. S. Poh, "Verifiable structured encryption," in *Proc. Int. Conf. Inf. Secur. Cryptol.*, Berlin, Germany: Springer, 2013, pp. 137–156.
[21] H. Yin, Z. Qin, J. Zhang, L. Ou, and K. Li, "Achieving secure, universal, and fine-grained query results verification for secure search scheme over encrypted cloud data," *IEEE Trans. Cloud Comput.*, vol. 9, no. 1, pp. 27–39, 2021.
[22] S. Sun, X. Yuan, J. K. Liu, and R. Steinfeld, "Practical backward-secure searchable encryption from symmetric puncturable encryption," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2018, pp. 763–780.
[23] S. Sun et al., "Practical non-interactive searchable encryption with forward and backward privacy," in *Proc. Netw. Distrib. Syst. Secur. (NDSS) Symp.*, 2021, pp. 1–18.
[24] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public-key encryption with keyword search," in *Proc. EUROCRYPR*, 2004, pp. 506–522.
[25] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. Theory Cryptogr. Conf.*, 2007, pp. 535–554.
[26] H. Yin, Z. Qin, L. Ou, and K. Li, "A privacy-enhanced and efficient search method for encrypted data in cloud computing," *J. Comput. Syst. Sci.*, vol. 90, pp. 14–27, 2017.
[27] H. Yin, Z. Qin, J. Zhang, L. Ou, F. Li, and K. Li, "Secure conjunctive multi-keyword ranked search over encrypted cloud data for multiple data owners," *Future Gener. Comput. Syst.*, vol. 100, pp. 689–700, 2019.
[28] 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, 2021.
[29] Y. Lu, J. Li, and Y. Zhang, "Privacy-preserving and pairing-free multirecipient certificateless encryption with keyword search for cloud-assisted IIoT," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 2553–2562, 2020.
[30] Y. Lu, J. Li, and F. Wang, "Pairing-free certificate-based searchable encryption supporting privacy-preserving keyword search function for IIoTs," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2696–2706, 2021.
[31] D. He, M. Ma, S. Zeadally, N. Kumar, and K. Liang, "Certificateless public key authenticated encryption with keyword search for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3618–3627, 2018.
[32] Y. Miao et al., "Hybrid keyword-field search with efficient key management for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, vol. 15, no. 6, pp. 3206–3217, 2019.
[33] 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. Secur.*, 2006, pp. 89–98.
[34] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2007, pp. 195–203.
[35] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy*, 2007, pp. 321–334.
[36] P. Bellini, D. Cenni, N. Mitolo, P. Nesi, G. Pantaleo, and M. Soderi, "High level control of chemical plant by industry 4.0 solutions," *J. Ind. Inf. Integration*, vol. 26, 2022, Art. no. 100276.
[37] T. Jung, X. Y. Li, Z. Wan, and M. Wan, "Privacy preserving cloud data access with multi-authorities," in *Proc. IEEE INFOCOM*, 2013, pp. 2626–2633.
[38] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Proc. EUROCRYPT*, 2011, pp. 547–567.

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 in software engineering from the College of Information Science and Engineering, Hunan University, Changsha, in 2018.

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

Yangfan Li received the bachelor's degree of engineering from the School of Automation, Huazhong University of Science and Technology, Wuhan, China, in 2015. He is currently working toward the Ph.D. degree in computer science with Hunan University, Changsha, China.

His research interests include information security, parallel and distributed computing, machine learning, and deep 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 Engineering and Applied Mathematics, Changsha University, Changsha, China. His research interests include applied cryptography, data security and privacy, and cloud security.

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 Engineering and Applied Mathematics, Changsha University, Changsha. His research interests include privacy protection, wireless networks, signal processing, and machine learning.

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

From 2010 to 2011, he was a Visiting Scholar with the Department of Computer Science, University of Michigan, Ann Arbor, MI, USA. He is currently a Professor with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, China, where he is also the Vice Dean. His research interests include network and data security, privacy, data analytics and applications, 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, Texas, USA, in 1990. He is currently a SUNY Distinguished Professor of Computer Science with the State University of New York, Albany, NY, USA. He is also a National Distinguished Professor with Hunan University, Changsha, China. He has authored or coauthored more than 850 journal articles, book chapters, and refereed conference papers. He holds over 70 patents announced or authorized by the Chinese National Intellectual Property Administration, Beijing, China. He is among the world's top five 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. 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, and intelligent and soft computing.

Mr. Li was the recipient of several best paper awards. He is currently an Associate Editor for the *ACM Computing Surveys* and the *CCF Transactions on High Performance Computing*. He was on the Editorial Board 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 a Fellow of the AAIA and a member of the Academia Europaea (Academician of the Academy of Europe).