Hindawi
Security and Communication Networks
Volume 2021, Article ID 9929663, 13 pages
https://doi.org/10.1155/2021/9929663

Research Article

Privacy-Preserving Attribute-Based Keyword Search with Traceability and Revocation for Cloud-Assisted IoT

Kai Zhang, Yanping Li, and Laifeng Lu

School of Mathematics and Statistics, Shaanxi Normal University, Xi'an 710119, China

Correspondence should be addressed to Laifeng Lu; lulaifeng@snnu.edu.cn

Received 19 March 2021; Accepted 23 May 2021; Published 30 May 2021

Academic Editor: Qing Yang

With the rapid development of cloud computing and Internet of Things (IoT) technology, it is becoming increasingly popular for source-limited devices to outsource the massive IoT data to the cloud. How to protect data security and user privacy is an important challenge in the cloud-assisted IoT environment. Attribute-based keyword search (ABKS) has been regarded as a promising solution to ensure data confidentiality and fine-grained search control for cloud-assisted IoT. However, due to the fact that multiple users may have the same retrieval permission in ABKS, malicious users may sell their private keys on the Internet without fear of being caught. In addition, most of existing ABKS schemes do not protect the access policy which may contain privacy information. Towards this end, we present a privacy-preserving ABKS that simultaneously supports policy hiding, malicious user traceability, and revocation. Formal security analysis shows that our scheme can not only guarantee the confidentiality of keywords and access policies but also realize the traceability of malicious users. Furthermore, we provide another more efficient construction for public tracing.

1. Introduction

As a prevalent Internet technology, Internet of Things (IoT) [1] has been widely used in various industries, such as smart healthcare, transportation, and city [2–5]. Due to the limited computing and storage capacity of many IoT devices, users often need to store IoT data in the cloud. The cloud-assisted IoT [6] technology can be used to collect and store massive medical data, so it is expected to greatly improve the efficiency of medical institutions and promote the development of smart healthcare. Apart from the efficiency concern, security issue is an important concern hindering the widespread application of IoT technology [7–10]. Especially for the smart healthcare system based on cloud-assisted IoT, the data security issue has become a key challenge, due to the fact that the sensitive personal health record (PHR) outsourced in the cloud is vulnerable to hacker attacks.

Although the traditional encryption technology [11] can protect the data security, it makes the ciphertext data unable to retrieve, thus greatly reducing the availability of IoT data. An inefficient solution is that data users download ciphertext data from the cloud, decrypt it, and then search on plaintext data. However, ordinary users do not have enough storage and computing power to retrieve the huge amount of cloud data locally. Public key encryption with keyword search (PEKS) [12, 13] is a more efficient solution, which can realize the retrieval of ciphertext by a cloud server without decryption. In a PEKS scheme, a data user can delegate the cloud server to retrieve all cloud ciphertexts by sending a search token to it. However, in order to avoid the abuse of retrieval ability, data owners usually want to control the retrieval permission.

As an efficient and flexible solution to meet the above requirements, attribute-based keyword search (ABKS) [14, 15] can realize data confidentiality, ciphertext retrieval, and fine-grained access control simultaneously. In a ciphertext-policy ABKS (CP-ABKS) system, a data owner encrypts the file keyword by an access policy and only users whose attributes satisfy the access policy can retrieve the ciphertext file. However, the public access policy in CP-ABKS may disclose privacy information in the smart medical cloud system. For example, a medical institution

wants to share PHR with users whose attributes meet the policy "(Institution: hospital A AND Patient ID: 202007953) OR (Institution: hospital B AND Position: oncologist)"; then it encrypts the PHR keyword by this policy and generates the corresponding ciphertext. Note that the access policy is exposed together with the ciphertext in the traditional CP-ABKS, so anyone can infer that the patient with the identity "202007953" is likely to have a tumor. Moreover, multiple users with the same attributes have the same retrieval ability and the user identity cannot be determined by the user private key in CP-ABKS, so malicious users may sell their private keys without worrying about being caught. As in the above example, if one of the multiple oncologists in hospital B sells his private key online, it is difficult to accurately identify and revoke the malicious user who sells his private key.

1.1. Our Contributions. Up till now, there is no secure ABKS scheme that simultaneously supports hidden policy, traceability, and revocation. To address these issues, we propose a traceable and revocable hidden ABKS (TR-HABKS) scheme and an enhanced TR-HABKS (eTR-HABKS) scheme, which support the above three properties at once. Moreover, the eTR-HABKS scheme achieves two other remarkable properties: (1) no identity table for tracing: the scheme only needs to maintain an identity table for revocation but does not require any identity table for tracing; (2) public traceability: besides the trusted authority, anyone without additional secret information can also run the tracing algorithm to capture malicious users. Specifically, our TR-HABKS and eTR-HABKS schemes provide the following properties:

(i) *Fine-Grained Search Control*. In our schemes, a data user's search token is corresponding to his attributes and can be used to retrieve ciphertext only when the attributes satisfy the ciphertext policy. To control the user search permission, our schemes allow the data owner to encrypt the keyword by a specified access policy, which can be expressed as an AND-gates on multivalued attributes.

(ii) *Hidden Policy*. Our schemes not only guarantee the confidentiality of the keyword but also protect the privacy of the policy. Different from those ABKS schemes which only prove the keyword security, we also prove that access policies are also indistinguishable in the selective security model. Moreover, our schemes require the data owner to encrypt the keyword by his private key, so that the adversary cannot launch the keyword guessing attacks (KGA) by generating the ciphertext himself.

(iii) *Traceability*. Both the TR-HABKS and eTR-HABKS schemes achieve the user traceability in ABKS. When a malicious user leaks his private key in our TR-HABKS scheme, then the trusted authority can determine the identity of the malicious user by a tracing identity table. In our eTR-HABKS scheme, everyone can trace the malicious user's identity without the help of any identity table.

(iv) *Revocation*. When the malicious user's identity is determined, our schemes can effectively revoke the user by managing a registration table. In our schemes, the trusted authority adds each legitimate user to a registered identity table in the key generation stage and can easily revoke the malicious user by deleting his identity from the identity table.

The properties comparison between our schemes with other related works can be seen in Table 1. The symbol "—" means not applicable.

1.2. Related Work

1.2.1. Attributed-Based Encryption. Attribute-based encryption (ABE) [19] is a practical method for fine-grained access control and can be divided into key-policy ABE (KP-ABE) [20] and ciphertext-policy ABE (CP-ABE) [21, 22]. Based on KP-ABE and CP-ABE, dual-policy ABE (DP-ABE) [23, 24] was also introduced for achieving content-based and role-based access control simultaneously. However, in traditional CP-ABE and DP-ABE, the access policy corresponding to the ciphertext may disclose the user's privacy. To address this problem, Nishide et al. [25] proposed the first CP-ABE in which access policies can be hidden by the encryptor. Later, Lai et al. [26] presented a high expressive CP-ABE with partially hidden access structure that can be expressed as a linear secret-sharing scheme (LSSS) [27]. Yang et al. [28] proposed a privacy-preserving CP-ABE to hide both the attribute names and the attribute values in the access policy. Based on an optimized vector transformation approach, Sun et al. [29] proposed a lightweight hiding CP-ABE scheme for IoT-oriented smart health. Their scheme can not only support policy hiding but also support offline encryption and outsourcing decryption. In order to prevent key abuse, Hinek et al. [30] first considered the trace problem in ABE and constructed a traceable ABE scheme. Liu et al. [31] proposed a high expressive white-box traceable ABE that supports traceability of the malicious user who sold his decryption key on the Internet. For a decryption black-box in ABE, Liu et al. [32] later proposed a black-box traceable CP-ABE that can trace the malicious user whose private key was used to construct the decryption device. To support more flexible attributes, Ning et al. [33] presented a traceable CP-ABE that simultaneously supports white-box traceability and large universe. Ying et al. [34] presented a black-box traceable CP-ABE with hidden policy in e-healthcare cloud. Recently, several novel ABE schemes [35–37] were proposed for stronger security and user revocation in cloud storage system. Unfortunately, the above ABE schemes cannot search the ciphertext data in the cloud.

1.2.2. Attribute-Based Keyword Search. Boneh et al. [12] first introduced the concept of PEKS and constructed the first concrete PEKS scheme. In the scheme, the user authorizes a third party to search the ciphertext by giving him a search token that is associated with a keyword; the third party

Table 1: Properties comparison.

Scheme	[12]	[16]	[14]	[15]	[17]	[18]	TR-HABKS	eTR-HABKS
Search	✓	✓	✓	✓	✓	✓	✓	✓
Fine-grained search control	×	×	✓	✓	✓	✓	✓	✓
Hidden policy	—	—	×	×	×	✓	✓	✓
Resist KGA	×	✓	×	×	×	✓	✓	✓
Traceability	×	×	×	×	×	×	Private traceability	Public traceability
Revocation	×	×	×	✓	✓	×	✓	✓

returns the search results to the user but without learning the keyword information. However, Byun et al. [38] pointed out that the PEKS scheme [12] cannot resist KGA. Specifically, anyone can generate a ciphertext by encrypting a keyword in PEKS scheme, so the third party can use the search token to continuously retrieve the ciphertexts corresponding to different keywords to guess the keyword corresponding to the search token. To resist the above attack in PEKS, Huang and Li [16] presented a public key authenticated encryption with keyword search, in which the keyword needs to be authenticated by the data owner during the encryption phase. Miao et al. [39] proposed a verifiable searchable encryption, which can achieve verifiable searchability and resist KGA. In order to support fine-grained search authorization, Zheng et al. [14] proposed a CP-ABKS scheme based on PEKS and CP-ABE [21]. In the CP-ABKS scheme, a data owner encrypts a keyword by an access policy and only users whose attributes meet the access policy can retrieve the ciphertext. With the help of proxy reencryption and lazy reencryption techniques, Sun et al. [15] presented a revocable ABKS scheme that can delegate the search and update workload to the cloud server. Liu et al. [17] proposed a searchable ABE with efficient revocation and outsourced decryption for cloud IoT. Based on online/offline encryption and outsourced decryption techniques, Miao et al. [40] presented an efficient ABKS scheme in the cloud-assisted healthcare IoT system. To protect access policies, Qiu et al. [18] presented a hidden policy CP-ABKS against KGA. Later, Miao et al. [41] presented a privacy-preserving CP-ABKS in multiowner setting. However, Sun et al. [42] pointed out that four types of KGA exist in this scheme. To achieve hidden policy and traceability simultaneously, Liu et al. [43] presented a privacy-preserving ABKS with user tracing. However, the security proof cannot ensure the policy hiding property due to its flawed security model. Unlike with the formal security model in hidden policy CP-ABKS [18, 41], the security model in [43] only shows the indistinguishability of keywords and does not consider the indistinguishability of access policies.

1.3. Organization. The rest of this paper is organized as follows. Section 2 introduces the necessary background information of the paper. Section 3 defines the algorithm and model for TR-HABKS. Section 4 presents the TR-HABKS construction and proves its correctness and security. Section 5 presents the eTR-HABKS construction and compares the efficiencies of the TR-HABKS and eTR-HABKS schemes. Section 6 concludes the paper.

2. Background

For a set S, let $s \longleftarrow_R S$ denote that an element s is chosen uniformly at random from S. Let $\mathbb{Z}_p$ denote the set $\{0, 1, 2, \ldots, p-1\}$, where p is a prime, let $[n]$ denote the set $\{1, 2, \ldots, n\}$, where n is a natural number, and let PPT denote probabilistic polynomial time.

2.1. Access Policy. In our system, the total number of attributes is n, and the access policy is represented by an AND-gates on multivalued attributes [25]. For each $i \in [n]$, let A_i be the attribute index, and let $S_i = \{v_{i,t}\}_{t \in n_i}$ be the possible values of A_i, where n_i is the number of possible values for A_i. Let $L = \{L_i\}$ be a user attributes set, where $L_i \in S_i$, and let $P = \{P_i\}_{i \in [n]}$ be an access policy, where $P_i \subseteq S_i$. If $L_i \in P_i$ for $i \in [n]$, we say that the attributes set L satisfies the access policy P, written as $L \vDash P$; otherwise, we say that the attributes set L does not satisfy the access policy P, written as $L \nvDash P$. For ease of description, we use i instead of A_i to represent attribute index in our schemes.

2.2. Bilinear Map. An asymmetric bilinear group generator $\mathcal{G}$ takes as input a security parameter λ and outputs a tuple $\mathbb{G} = (p, G_1, G_2, G_T, g_1, g_2, e)$, where p is a prime, G_1, G_2, and G_T are multiplicative cyclic groups of order p, g_1 (resp., g_2) is a generator of G_1 (resp., G_2), and e: $G_1 \times G_2 \longrightarrow G_T$ is an efficiently computable bilinear map with the following properties:

(1) Bilinear: $\forall g \in G_1, h \in G_2, a, b \in \mathbb{Z}_p, e \quad (g^a, h^b) = e(g, h)^{ab}$

(2) Nondegenerate: $e(g_1, g_2) \neq 1$

2.3. Signature. A signature scheme consists of the following algorithms:

$(\mathrm{PK}, \mathrm{SK}) \longleftarrow$ KeyGen (λ): The key generation algorithm gets the security parameter λ as input. It outputs a random key pair $(\mathrm{PK}, \mathrm{SK})$.

$(\sigma) \longleftarrow$ Sign (SK, M): The signing algorithm gets a private key SK and a message M as input. It outputs a signature σ.

$(0/1) \longleftarrow$ Verify (PK, M, σ): The verifying algorithm gets a public key PK, a message M, and a signature σ as input. It outputs 1 if the signature is valid, and outputs 0 otherwise.

The existential unforgeability under a weak chosen message attack [44] is defined by the following game:

Query: the adversary sends messages $\{M_j\}_{j\in[q_s]}$ to the challenger, where q_s is the maximum number of signatures that the adversary can query.

Response: the challenger runs the key generation algorithm and generates the signatures $\{\sigma_j\}_{j\in[q_s]}$ on the messages $\{M_j\}_{j\in[q_s]}$. Then, the challenger gives the public key PK and the signatures $\{\sigma_j\}_{j\in[q_s]}$ to the adversary.

Output: the adversary outputs a pair (M, σ).

The adversary wins this game if verify $(\mathrm{PK}, M, \sigma) = 1$ and $(M, \sigma) \notin \{(M_j, \sigma_j)\}_{j\in[q_s]}$. The adversary's advantage is defined as the probability that he wins this game.

Definition 1. A signature scheme is said to be existentially unforgeable under a weak chosen message attack if all PPT adversaries have only a negligible advantage in this game.

3. Problem Formulation

In this section, we describe the algorithm definition, system model, and security model of TR-HABKS.

3.1. Algorithm Definition. A TR-HABKS scheme is formally defined as follows:

$(\mathrm{MK}, \mathrm{PK}) \longleftarrow$ Setup (λ): the setup algorithm gets the security parameter as input. It outputs the master key MK and the public parameter PK. In addition, it also generates two empty identity tables T_1 and T_2.

$(\mathrm{SK}_{\mathrm{id},L}, \mathrm{SK}_o) \longleftarrow$ KeyGen $(\mathrm{MK}, \mathrm{PK}, \mathrm{id}, L)$: the key generation algorithm gets an attributes set L, an identity id, the master key MK, and the public parameter PK as input. It outputs a private key $\mathrm{SK}_{\mathrm{id},L}$ for a data user and a private key SK_o for the data owner. In addition, it adds id to T_1 and T_2.

$\mathrm{CT} \longleftarrow$ Enc $(\omega, P, \mathrm{SK}_o, \mathrm{PK})$: the encryption algorithm gets a keyword ω, an access policy P, the data owner's private key SK_o, and the public parameter PK as input. It outputs a ciphertext CT.

$\mathrm{TK}_{\mathrm{id},L} \longleftarrow$ TokenGen $(\mathrm{SK}_{\mathrm{id},L}, \mathrm{PK}, \omega')$: the token generation algorithm gets a data user's private key $\mathrm{SK}_{\mathrm{id},L}$, the public parameter PK, and a keyword ω' as input. It outputs a search token $\mathrm{TK}_{\mathrm{id},L}$.

$(0/1) \longleftarrow$ Search $(\mathrm{TK}_{\mathrm{id},L}, \mathrm{CT}, T_1)$: the searching algorithm gets a ciphertext CT, a search token $\mathrm{TK}_{\mathrm{id},L}$, and an identity table T_1 as input. It outputs 1 if (1) $L \vDash P$, (2) $\mathrm{id} \in T_1$, and (3) $\omega = \omega'$ and outputs 0 otherwise.

$(\mathrm{id}/\top) \longleftarrow$ Trace $(\mathrm{SK}_{\mathrm{id},L}, \mathrm{PK}, T_2)$: the tracing algorithm gets a secret key $\mathrm{SK}_{\mathrm{id},L}$, the public parameter PK, and an identity table T_2 as input. It outputs a user identity id if $\mathrm{SK}_{\mathrm{id},L}$ passes the key sanity check and outputs symbol $\top$ otherwise. Key sanity check is a deterministic algorithm to test whether $\mathrm{SK}_{\mathrm{id},L}$ needs to be traced.

$T_1 \longleftarrow$ Revoke (id, T_1): the revocation algorithm gets a revocation user identity id and an identity table T_1 as input. It outputs an updated table T_1.

3.1.1. Correctness. A TR-HABKS scheme is correct if the following condition holds: Given $(\mathrm{MK}, \mathrm{PK}) \longleftarrow$ Setup (λ), $(\mathrm{SK}_{\mathrm{id},L}, \mathrm{SK}_o) \longleftarrow$ KeyGen $(\mathrm{MK}, \mathrm{PK}, \mathrm{id}, L), \mathrm{CT} \longleftarrow$ Enc $(\omega, P, \mathrm{SK}_o, \mathrm{PK})$, $\mathrm{TK}_{\mathrm{id},L} \longleftarrow$ TokenGen $(\mathrm{SK}_{\mathrm{id},L}, \mathrm{PK}, \omega')$, where $L \vDash P$ and $\mathrm{id} \in T_1$; then Search $(\mathrm{TK}_{\mathrm{id},L}, \mathrm{CT}, T_1)$ outputs 1 when $\omega = \omega'$.

3.2. System Model. As depicted in Figure 1, our TR-HABKS system includes four entities: a trusted authority (TA), a data owner (DO), a cloud sever (CS), and multiple data users (DUs). Specifically, the role of each entity in our system model is described below.

TA: TA first runs the setup algorithm, keeps the master key MK secretly, and publishes the public parameter PK. Then, he uses his master key to generate private keys for DO and DUs. In addition, he creates an identity table T_1 for user revocation and another identity table T_2 for the malicious user tracing. When a malicious user sells his private key on the Internet, TA runs the tracing algorithm and then obtains the malicious user identity id from T_2. Finally, TA deletes id from table T_1 and sends T_1 to CS to revoke the malicious user's search ability.

DO: when DO wants to encrypt a keyword ω under an access policy P, he runs the encryption algorithm with his private key SK_o and then generates a ciphertext CT corresponding to (P, ω). Finally, he outsources the corresponding ciphertext CT to the cloud.

DU: when DU wants to search the data files with the keyword ω', he runs the token generation algorithm with his private key $\mathrm{SK}_{\mathrm{id},L}$ and then generates a search token $\mathrm{TK}_{\mathrm{id},L}$ corresponding to $(\mathrm{id}, L, \omega')$. Finally, he sends $\mathrm{TK}_{\mathrm{id},L}$ to CS to query documents containing the keyword ω'.

CS: when CS receives the search token $\mathrm{TK}_{\mathrm{id},L}$ from DU, he first searches id in the table T_1. If $\mathrm{id} \notin T_1$, CS returns 0 and aborts; otherwise, CS runs the searching algorithm and returns the search result to DU.

In our threat model, TA and DO are assumed to be fully trusted; that is, they execute the above algorithm honestly and will not attack the system. CS is assumed to be an honest-but-curious adversary who honestly executes the searching algorithm but tries to infer the privacy of keywords. Note that the generation of ciphertext needs to use the private key of DO, so CS cannot generate ciphertext by itself and carry out keyword guessing attack. DUs in our system may be malicious adversaries who not only try to retrieve the ciphertext beyond their retrieval permission but also leak their private keys to others.

3.3. Security Model. In order to realize the confidentiality of keywords and access policies simultaneously, the security

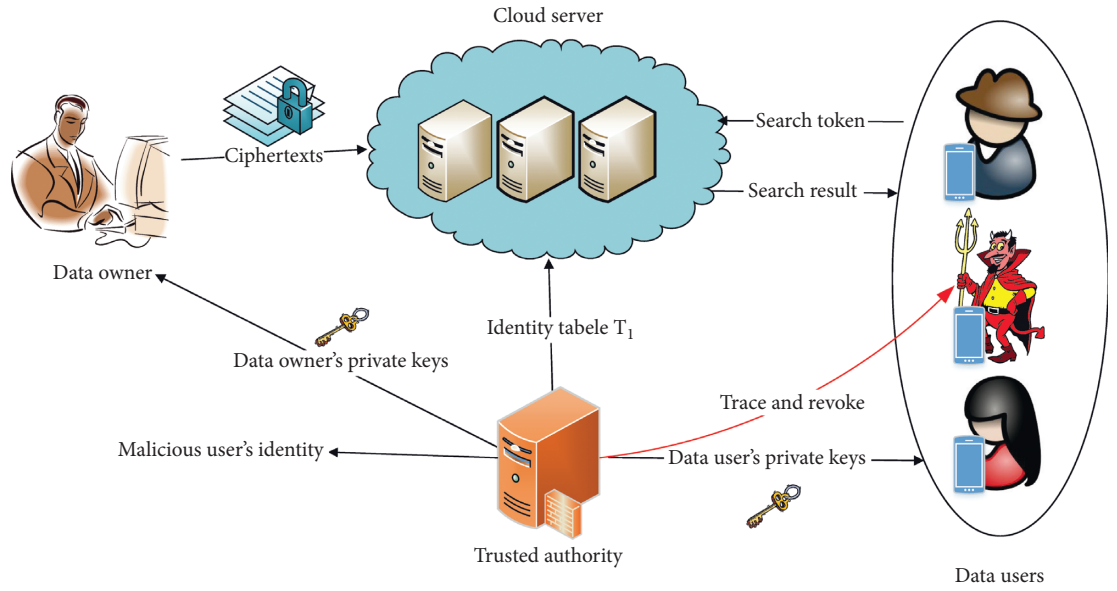

Figure 1: System model of TR-HABKS.

model of our TR-HABKS scheme requires that the adversary cannot distinguish between the encryption of a keyword ω_0 under an access policy P_0 and the encryption of a keyword ω_1 under an access policy P_1. In the selective security model, the adversary needs to submit two challenge access policies P_0 and P_1 before the Setup phase. The selective security game includes the following phases:

Inti: the adversary declares two challenge access policies P_0 and P_1 that he tries to attack and gives them to the challenger.

Setup: the challenger calls the setup algorithm $(\mathrm{MK}, \mathrm{PK}) \longleftarrow \mathrm{Setup}(\lambda)$ and gives the public parameter PK to the adversary.

Query Phase1: the adversary can repeatedly ask for private keys and search tokens as follows:

(1) *Private Key Query* $\mathcal{O}_{\mathrm{KeyGen}}(\mathrm{id}, L)$: the adversary submits an identity id and an attributes set L to the challenger. If $(L \vDash P_0 \wedge L \nvDash P_1)$ or $(L \nvDash P_0 \wedge L \vDash P_1)$, then abort; otherwise, the challenger returns the corresponding private key $\mathrm{SK}_{\mathrm{id},L}$.

(2) *Search Token Query* $\mathcal{O}_{\mathrm{TokenGen}}(\mathrm{id}, L, w)$: the adversary submits an identity id, an attributes set L, and a keyword ω to the challenger. The challenger returns the corresponding search token $\mathrm{TK}_{\mathrm{id},L}$.

Challenge: the adversary submits two keywords ω_0 and ω_1 that satisfy the following constraint. If the adversary has queried the private key or search token for the attributes set L that satisfies both access policies P_0 and P_1, then we require that $\omega_0 = \omega_1$. The challenger flips a random coin $\gamma \in \{0, 1\}$ and returns the challenge ciphertext $\mathrm{CT}^* \longleftarrow \mathrm{Enc}(\omega_\gamma, P_\gamma, \mathrm{SK}_o, \mathrm{PK})$ to the adversary.

Query Phase2: phase 1 is repeated with the restriction that the adversary cannot query the private key or search token for the attributes set L when $(L \vDash P_0 \wedge L \vDash P_1)$ and $\omega_0 \neq \omega_1$.

Guess: the adversary outputs a guess $\gamma' \in \{0, 1\}$.

The adversary wins this game if $\gamma = \gamma'$, and his advantage is defined as $\Pr[\gamma = \gamma'] - (1/2)$.

Definition 2. A TR-HABKS scheme is said to be selectively secure if all PPT adversaries have only a negligible advantage in the above security game.

The traceability game of TR-HABKS is described as follows:

Setup: the challenger runs the setup algorithm $(\mathrm{MK}, \mathrm{PK}) \longleftarrow \mathrm{Setup}(\lambda)$ and gives the public parameter PK to the adversary.

Key query: the adversary queries the private keys corresponding to pairs $\{(\mathrm{id}_j, L_j)\}_{j \in [q_s]}$, where id_j is an identity, L_j is an attributes set, and q_s is the maximum number of private keys that the adversary can query. The challenger returns the corresponding user private keys $\left\{\mathrm{SK}_{\mathrm{id}_j, L_j}\right\}_{j \in [q_s]}$.

Key forgery: the adversary outputs a user private key SK^*.

In this game, the adversary's advantage is defined as $\Pr[\mathrm{Trace}(\mathrm{SK}^*, \mathrm{PK}, T_2) \notin \{\mathrm{id}_1, \mathrm{id}_2, \ldots, \mathrm{id}_{q_s}, \top\}]$.

Definition 3. A TR-HABKS scheme is said to be fully traceable if all PPT adversaries have only a negligible advantage in this traceability game.

4. Our TR-HABKS Scheme

In this section, we propose the construction of our TR-HABKS scheme and prove that it is selectively secure and fully traceable in the generic bilinear group model. We first adopt the technique from [18, 25] to realize hidden policy. The access policy P is embedded in the ciphertext CT as follows: if $v_{i,t} \in P_i$, we set $C_{i,t,1} = A_{i,t,1}^{\alpha_i}, C_{i,t,2} = A_{i,t,2}^{\alpha_i}$; otherwise, we set $C_{i,t,1}$ and $C_{i,t,2}$ as two random elements in G_1. That is, if $v_{i,t} \in P_i$, these ciphertext components $C_{i,t,1}, C_{i,t,2}$ are well formed and can be used for successful search; otherwise, the ciphertext components $C_{i,t,1}, C_{i,t,2}$ are malformed. As it is hard to distinguish the well-formed ciphertext components from the malformed ciphertext components, the user cannot get the access policy from the corresponding ciphertext. Then, we exploit the signature technique in [31, 44] to realize the user traceability. On one hand, we inject the message y_{id} and its signature into the user private key; then DU cannot rerandomize the private key component y_{id}. On the other hand, we add the message y_{id} and the corresponding user identity id in the identity table T_2; then TA can identify the malicious user by the private key and the table T_2. Finally, we add the legitimate user to the system by storing the user identity id and its corresponding element C_{id} in the registered identity table T_1 and revoke the malicious user by deleting the corresponding pair $(\mathrm{id}, C_{\mathrm{id}})$ from the table T_1.

4.1. Construction. Setup (λ): TA first runs $\mathcal{G}(\lambda)$ to obtain $(p, G_1, G_2, G_T, g_1, g_2, e)$, where G_1, G_2, and G_T are cyclic groups with prime order p, and $e: G_1 \times G_2 \longrightarrow G_T$ is a bilinear map. Then, TA picks $a, b, c \longleftarrow_R \mathbb{Z}_p$ and a one-way hash function $H: \{0,1\}^* \longrightarrow \mathbb{Z}_p$. For each $i \in [n]$, TA chooses random exponents $\{a_{i,t} \in \mathbb{Z}_p\}_{t\in[n_i]}$ and computes $\{A_{i,t,1} = g_1^{a_{i,t}}, A_{i,t,2} = g_1^{ca_{i,t}}\}_{t\in[n_i]}$. Next, TA sets $\mathrm{MK} = (a, b, c, \{\{a_{i,t}\}_{t\in[n_i]}\}_{i\in[n]})$ as his master key and publishes the public parameter $\mathrm{PK} = (p, G_1, G_2, G_T, g_1, g_2, e, e(g_1, g_2)^a, g_1^b, g_1^c, g_1^{bc}, H, \{\{A_{i,t,1}, A_{i,t,2}\}_{t\in[n_i]}\}_{i\in[n]})$. Finally, TA creates two empty identity tables T_1 and T_2.

KeyGen $(\mathrm{MK}, \mathrm{PK}, \mathrm{id}, L)$: DU submits his identity id and attributes set $L = \{v_{i,t_i}\}_{i\in[n]}$ to TA in order to apply for the user private key. TA first picks $x_{\mathrm{id}}, y_{\mathrm{id}}, \beta \longleftarrow_R \mathbb{Z}_p$ and sets $K = x_{\mathrm{id}}, K_0 = g_2^{a/b(c+y_{\mathrm{id}})} g_2^{\beta/b}, K_1 = y_{\mathrm{id}}$. For each $i \in [n]$, TA picks $\lambda_i \longleftarrow_R \mathbb{Z}_p$ and computes $K_{i,1} = g_2^{\beta+\lambda_i a_{i,t_i}}, K_{i,2} = g_2^{\lambda_i}$. Then, TA sets $\mathrm{SK}_{\mathrm{id},L} = (K, K_0, K_1, \{K_{i,1}, K_{i,2}\}_{i\in[n]})$ as DU private key and sends it to the DU with identity id. Next, TA picks $\alpha \longleftarrow_R \mathbb{Z}_p$, sets $\mathrm{SK}_o = \alpha$ as DO private key, and sends it to DO. After that, TA computes $C_{\mathrm{id}} = e(g_1, g_2)^{-a\alpha x_{\mathrm{id}}}$, stores $(\mathrm{id}, C_{\mathrm{id}})$ in the registered identity table T_1, and sends T_1 to CS for search permission revocation. Finally, TA adds $(\mathrm{id}, y_{\mathrm{id}})$ in the identity table T_2 and secretly stores T_2 for user tracing.

Enc $(\omega, P, \mathrm{SK}_o, \mathrm{PK})$: to encrypt a keyword under an access policy $P = \{P_i\}_{i\in[n]}$, DO computes $C = e(g_1, g_2)^{a\alpha}, C_0 = g_1^{b\alpha/H(\omega)}, C_1 = g_1^{bc\alpha/H(\omega)}$. For each $i \in [n]$, DO chooses $\alpha_i \longleftarrow_R \mathbb{Z}_p$ such that $\sum_{i\in[n]} \alpha_i = \alpha$, computes $C_{i,1} = g_1^{\alpha_i}, C_{i,2} = g_1^{c\alpha_i}$, and sets $C_{i,t,1}$ for each $t \in [n_i]$ as follows: if $v_{i,t} \in P_i$, it sets $C_{i,t,1} = A_{i,t,1}^{\alpha_i}, C_{i,t,2} = A_{i,t,2}^{\alpha_i}$; otherwise, it sets $C_{i,t,1}$ and $C_{i,t,2}$ as two random elements in G_1. Finally, DO uploads the ciphertext $\mathrm{CT} = (C, C_0, C_1, \{C_{i,1}, C_{i,2}, \{C_{i,t,1}, C_{i,t,2}\}_{t\in[n_i]}\}_{i\in[n]})$ into the cloud.

TokenGen $(\mathrm{SK}_{\mathrm{id},L}, \mathrm{PK}, \omega')$: to generate a search token for a keyword $\omega' \in \{0,1\}^*$, DU picks $s \longleftarrow_R \mathbb{Z}_p$ and computes $\mathrm{tok}_0 = K_0^{H(\omega')s}, \mathrm{tok} = K + s, \mathrm{tok}_1 = K_1$. For each $i \in [n]$, DU computes $T_{i,1} = K_{i,1}^s, T_{i,2} = K_{i,2}^s$. Finally, DU sends the search token $\mathrm{TK}_{\mathrm{id},L} = (\mathrm{tok}_0, \mathrm{tok}, \mathrm{tok}_1, \{T_{i,1}, T_{i,2}\}_{i\in[n]})$ to CS.

Search $(\mathrm{TK}_{\mathrm{id},L}, \mathrm{CT}, T_1)$: when CS receives the search token $\mathrm{TK}_{\mathrm{id},L} = (\mathrm{tok}_0, \mathrm{tok}, \mathrm{tok}_1, \{T_{i,1}, T_{i,2}\}_{i\in[n]})$ from the DU with identity id, it first searches the entry $(\mathrm{id}, C_{\mathrm{id}})$ in the table T_1. If no such entry exists, CS returns 0 and aborts; otherwise, CS obtains C_{id} from T_1 and runs the following search algorithm. *If Algorithm.* If $L = \{v_{i,t_i}\}_{i\in[n]}$, it computes $E = \prod_{i\in[n]} (e(C_{i,1}^{\mathrm{tok}_1} C_{i,2}, T_{i,1}) / e(C_{i,t_i,1}^{\mathrm{tok}_1} C_{i,t_i,2}, T_{i,2})) = e(g_1, g_2)^{\alpha\beta s(y_{\mathrm{id}}+c)}$. Finally, CS returns 1 if $\mathrm{EC}^{\mathrm{tok}} C_{\mathrm{id}} = e(C_0^{\mathrm{tok}_1} C_1, \mathrm{tok}_0)$ and 0 otherwise.

Trace $(\mathrm{SK}_{\mathrm{id},L}, \mathrm{PK}, T_2)$: if the private key is not in the form of $\mathrm{SK}_{\mathrm{id},L} = (K, K_0, K_1, \{K_{i,1}, K_{i,2}\}_{i\in[n]})$, TA returns $\top$ and aborts; otherwise, TA runs the following key sanity check algorithm. $K, K_1 \in \mathbb{Z}_p, K_0, K_{i,1}, K_{i,2} \in G_2, \exists i \in [n]$, s.t.

$$e\left(g_1^{bc} g_1^{bK_1}, K_0\right) e\left(A_{i,t_i,1}^{K_1} A_{i,t_i,2}, K_{i,2}\right) = e(g_1, g_2)^a e\left(g_1^c g_1^{K_1}, K_{i,1}\right). \tag{1}$$

If the private key $\mathrm{SK}_{\mathrm{id},L}$ does not pass the above check, TA returns $\top$ and aborts; otherwise, TA searches the entry (id, K_1) in table T_2 and returns the corresponding id.

Revoke (id, T_1): to revoke the search permission of the malicious user with identity id, TA updates table T_1 by deleting the entry $(\mathrm{id}, C_{\mathrm{id}})$ and sends new table T_1 to CS.

4.2. Correctness Proof. We now prove the correctness of our TR-HABKS scheme. *If Scheme.* If the user attributes $L = \{v_{i,t_i}\}_{i\in[n]}$ satisfy the access policy $P = \{P_i\}_{i\in[n]}$, we have $v_{i,t_i} \in P_i$ and $C_{i,t_i,1} = A_{i,t_i,1}^{\alpha_i}, C_{i,t_i,2} = A_{i,t_i,2}^{\alpha_i}$ for each $i \in [n]$. Then,

$$
\begin{aligned}
&\prod_{i\in[n]}\frac{e\left(C_{i,1}^{\mathrm{tok}_1}C_{i,2},T_{i,1}\right)}{e\left(C_{i,t_i,1}^{\mathrm{tok}_1}C_{i,t_i,2},T_{i,2}\right)}\\
&=\prod_{i\in[n]}\frac{e\left(g_1^{\alpha_i y_{\mathrm{id}}}g_1^{c\alpha_i},K_{i,1}^{s}\right)}{e\left(A_{i,t_i,1}^{\alpha_i y_{\mathrm{id}}}A_{i,t_i,2}^{\alpha_i},K_{i,2}^{s}\right)}\\
&=\prod_{i\in[n]}\frac{e\left(g_1^{y_{\mathrm{id}}+c},g_2^{\beta+\lambda_i a_{i,t_i}}\right)^{s\alpha_i}}{e\left(g_1^{(y_{\mathrm{id}}+c)a_{i,t_i}},g_2^{\lambda_i}\right)^{s\alpha_i}}\\
&=\prod_{i\in[n]}\frac{e\left(g_1^{y_{\mathrm{id}}+c},g_2^{\beta}\right)^{s\alpha_i}e\left(g_1^{y_{\mathrm{id}}+c},g_2^{\lambda_i a_{i,t_i}}\right)^{s\alpha_i}}{e\left(g_1^{(y_{\mathrm{id}}+c)a_{i,t_i}},g_2^{\lambda_i}\right)^{s\alpha_i}}\\
&=\prod_{i\in[n]}e\left(g_1,g_2\right)^{\beta s\left(y_{\mathrm{id}}+c\right)\alpha_i}\\
&=e\left(g_1,g_2\right)^{\beta s\left(y_{\mathrm{id}}+c\right)\sum_{i\in[n]}\alpha_i}\\
&=e\left(g_1,g_2\right)^{\alpha\beta s\left(y_{\mathrm{id}}+c\right)}.
\end{aligned}
\tag{2}
$$

If the user id is in table T_1, then CS can obtain the corresponding $C_{\mathrm{id}}=e(g_1,g_2)^{-a\alpha x_{\mathrm{id}}}$. Therefore,

$$
\begin{aligned}
&\mathrm{EC}^{\mathrm{tok}}C_{\mathrm{id}}\\
&=e\left(g_1,g_2\right)^{\alpha\beta s\left(y_{\mathrm{id}}+c\right)}e\left(g_1,g_2\right)^{a\alpha\left(x_{\mathrm{id}}+s\right)}e\left(g_1,g_2\right)^{-a\alpha x_{\mathrm{id}}}\\
&=e\left(g_1,g_2\right)^{\alpha\beta s\left(y_{\mathrm{id}}+c\right)}e\left(g_1,g_2\right)^{a\alpha s}.
\end{aligned}
\tag{3}
$$

In this case, if $\omega=\omega'$, we have

$$
\begin{aligned}
&e\left(C_0^{\mathrm{tok}_1}C_1,\mathrm{tok}_0\right)\\
&=e\left(g_1^{b\alpha y_{\mathrm{id}}/H(\omega)}g_1^{bc\alpha/H(\omega)},K_0^{H(\omega')s}\right)\\
&=e\left(g_1^{b\alpha\left(c+y_{\mathrm{id}}\right)},g_2^{\left(a/b\left(c+y_{\mathrm{id}}\right)\right)}g_2^{\beta/b}\right)^{s}\\
&=e\left(g_1^{b\alpha\left(c+y_{\mathrm{id}}\right)},g_2^{a/b\left(c+y_{\mathrm{id}}\right)}\right)^{s}e\left(g_1^{b\alpha\left(c+y_{\mathrm{id}}\right)},g_2^{\beta/b}\right)^{s}\\
&=e\left(g_1,g_2\right)^{a\alpha s}e\left(g_1,g_2\right)^{\alpha\beta s\left(y_{\mathrm{id}}+c\right)}\\
&=\mathrm{EC}^{\mathrm{tok}}C_{\mathrm{id}}.
\end{aligned}
\tag{4}
$$

4.3. Proof of Selective Security. In this part, we prove the confidentiality of keywords and access policies in our scheme by a security reduction to the QLSZ scheme [18]. More specifically, if there are any attacks in our TR-HABKS scheme, then we can use these attacks to break the QLSZ scheme in the generic bilinear group model [18, 45]. Followed by the definition in [45], we consider three random encodings $\varphi_1,\varphi_2,\varphi_T\colon F_p\longrightarrow\{0,1\}^m$, where F_p is an additive group and $m>3\log(p)$. For $i=1,2,T$, let $G_i=\{\varphi_i(x)\colon x\in F_p\}$. Therefore, there are three oracles to compute the group action on G_1,G_2,G_T and an oracle to compute the bilinear map e. We refer to G_1 as a generic bilinear group. In addition, our TR-HABKS scheme only allows DO to generate ciphertext by his private key, so the adversary cannot successfully carry out the keyword guessing attack.

Theorem 1. *If the QLSZ scheme is selectively secure in the generic bilinear group model, then our TR-HABKS scheme is selectively secure.*

Proof. Suppose that there exists a PPT adversary $\mathscr{A}$ that can break our TR-HABKS scheme with advantage ε in the selective security model. We will build a simulator $\mathscr{B}$ that can break the QLSZ scheme with advantage ε. Let $\mathscr{C}$ be the challenger corresponding to $\mathscr{B}$ in the security game of QLSZ scheme. For more information about the QLSZ scheme and its security, please refer to [18].

Inti: simulator $\mathscr{B}$ receives two challenge access policies P_0 and P_1 from adversary $\mathscr{A}$ and then sends these policies to challenger $\mathscr{C}$.

Setup: $\mathscr{C}$ sends the QLSZ public parameter $\overline{\mathrm{PK}}=(p,G_1,G_2,G_T,\ g_1,g_2,e,e(g_1,g_2)^a,\ g_1^b,H,\left\{\left\{A_{i,t}\right\}_{t\in[n_i]}\right\}_{i\in[n]})$ to $\mathscr{B}$. Then, $\mathscr{B}$ picks $c\longleftarrow_R\mathbb{Z}_p$, sets $\left\{A_{i,t,1}=A_{i,t},A_{i,t,2}=A_{i,t}^{c}\right\}_{t\in[n_i]}$, and sends the public parameter $\mathrm{PK}=(p,G_1,G_2,G_T,\ g_1,g_2,e,e(g_1,g_2)^a,\ g_1^b,g_1^c,g_1^{bc},H,\left\{\left\{A_{i,t,1},A_{i,t,2}\right\}_{t\in[n_i]}\right\}_{i\in[n]})$ to $\mathscr{A}$.

Query Phase1: $\mathcal{B}$ answers the adversary's private key and search token queries in our TR-HABKS scheme as follows:

(1) Private key query $\mathcal{O}_{\text{KeyGen}}(\text{id}, L)$: in order to create the user private key for a pair (id, L) requested by $\mathcal{A}$, $\mathcal{B}$ first sends (id, L) to $\mathcal{C}$ to query the corresponding user private key in QLSZ scheme. When $\mathcal{B}$ receives the QLSZ private key $\text{SK} = (x_{\text{id}}, K_0' = g_2^{(a+\beta')/b}, \{K_{i,1}' = g_2^{\beta'+\lambda_i' a_{i,t_i}}, K_{i,2}' = g_2^{\lambda_i'}\}_{i\in[n]})$, it picks $y_{\text{id}} \longleftarrow_R \mathbb{Z}_p$, implicitly sets $\beta = \beta'/(y_{\text{id}}+c)$, $\lambda_i = \lambda_i'/(y_{\text{id}}+c)$, and sets $K = x_{\text{id}}, K_1 = y_{\text{id}}$. Then, it computes $K_0 = (K_0')^{1/(y_{\text{id}}+c)} = g_2^{a/b(c+y_{\text{id}})} g_2^{\beta'/b(c+y_{\text{id}})} = g_2^{a/b(c+y_{\text{id}})} g_2^{\beta/b}$, $K_{i,1} = (K_{i,1}')^{1/(y_{\text{id}}+c)} = g_2^{(\beta'/(y_{\text{id}}+c))} + (\lambda_i'/(y_{\text{id}}+c))a_{i,t_i} = g_2^{\beta+\lambda_i a_{i,t_i}}$, $K_{i,2} = (K_{i,2}')^{1/(y_{\text{id}}+c)} = g_2^{\lambda_i'/(y_{\text{id}}+c)} = g_2^{\lambda_i}$. Finally, $\mathcal{B}$ sets the user private key $\text{SK}_{\text{id},L} = (K, K_0, K_1, \{K_{i,1}, K_{i,2}\}_{i\in[n]})$ and sends it to $\mathcal{A}$.

(2) Search token query $\mathcal{O}_{\text{TokenGen}}(\text{id}, L, w)$: in order to create the search token for a tuple (id, L, w) requested by $\mathcal{A}$, $\mathcal{B}$ first runs $\mathcal{O}_{\text{KeyGen}}(\text{id}, L)$ to obtain $\text{SK}_{\text{id},L} = (K, K_0, K_1, \{K_{i,1}, K_{i,2}\}_{i\in[n]})$. Then, $\mathcal{B}$ picks $s \longleftarrow_R \mathbb{Z}_p$ and computes $\text{tok}_0 = K_0^{H(\omega')s}, \text{tok} = K + s, \text{tok}_1 = K_1, T_{i,1} = K_{i,1}^s, T_{i,2} = K_{i,2}^s$. Finally, $\mathcal{B}$ sets the search token $\text{TK}_{\text{id},L} = (\text{tok}_0, \text{tok}, \text{tok}_1, \{T_{i,1}, T_{i,2}\}_{i\in[n]})$ and sends it to $\mathcal{A}$.

Challenge: $\mathcal{A}$ submits two keywords ω_0 and ω_1 to $\mathcal{B}$, where the pair (ω_0, ω_1) satisfies the following constraint. If the adversary has queried the private key or search token for the attributes set L that satisfies both access policies P_0 and P_1, then we require that $\omega_0 = \omega_1$. Next, $\mathcal{B}$ sends ω_0 and ω_1 to $\mathcal{C}$ to obtain the QLSZ challenge ciphertext $\text{CT}' = (C', C_0', \{C_{i,1}', \{C_{i,t,2}'\}_{t\in[n_i]}\}_{i\in[n]})$. Finally, $\mathcal{B}$ sets $C = C', C_0 = C_0', C_1 = C_0^c, C_{i,1} = C_{i,1}', C_{i,2} = C_{i,1}^c, C_{i,t,1} = C_{i,t,2}', C_{i,t,2} = C_{i,t,1}^c$ and returns the challenge ciphertext $\text{CT}^* = (C, C_0, C_1, \{C_{i,1}, C_{i,2}, \{C_{i,t,1}, C_{i,t,2}\}_{t\in[n_i]}\}_{i\in[n]})$ to $\mathcal{A}$.

Query Phase2: same as Phase 1.

Guess: finally, $\mathcal{A}$ outputs a guess $\gamma' \in \{0, 1\}$, and then $\mathcal{B}$ sends γ' to $\mathcal{C}$.

According to the above simulation, if $\mathcal{A}$ has advantage ε in the selective security game against our scheme, then $\mathcal{B}$ can use $\mathcal{A}$ to break the QLSZ scheme with the same advantage ε. □

4.4. Proof of Traceability. In this part, we prove that our TR-HABKS scheme is fully traceable based on the BB basic signature scheme, which has been proved to be existentially unforgeable under a weak chosen message attack in [44]. In order to prove the traceability, we assume that there exists a PPT adversary $\mathcal{A}$ who can win the traceability game, and then we construct a simulator $\mathcal{B}$ that can break the BB basic signature scheme under a weak chosen message attack. Let $\mathcal{C}$ denote the BB challenger corresponding to $\mathcal{B}$ in the signature security game. For more information about the BB basic signature scheme and its security, please refer to [44].

Theorem 2. *If the BB basic signature scheme is existentially unforgeable under a weak chosen message attack, then our TR-HABKS scheme is fully traceable.*

Proof. Before the game starts, challenger $\mathcal{C}$ sets a bilinear group $(p, G_1, G_2, G_T, g_1, g_2, e)$. G_1, G_2, and G_T are the cyclic groups of prime order p, g_1 is a generator of G_1, g_2 is a generator of G_2, and $e: G_1 \times G_2 \longrightarrow G_T$ is a bilinear map. □

4.4.1. Setup. Simulator $\mathcal{B}$ first chooses random values $\{y_j \in \mathbb{Z}_p\}_{j\in[q_s]}$ and makes signature queries by submitting the values $\{y_j \in \mathbb{Z}_p\}_{j\in[q_s]}$ to challenger $\mathcal{C}$. Then, $\mathcal{C}$ sends the BB public parameter $\overline{\text{PK}} = (p, G_1, G_2, G_T, e, g_1, g_2, g_1^c)$ and the corresponding signatures $\{y_j, \sigma_j = g_2^{1/(y_j+c)}\}_{j\in[q_s]}$ to $\mathcal{B}$. For each $i \in [n]$, $\mathcal{B}$ chooses random exponents $\{a_{i,t} \in \mathbb{Z}_p\}_{t\in[n_i]}$ and computes $\{A_{i,t,1} = g_1^{a_{i,t}}, A_{i,t,2} = g_1^{ca_{i,t}}\}_{t\in[n_i]}$. Next, $\mathcal{B}$ picks $a, b \longleftarrow_R \mathbb{Z}_p$ and a one-way hash function $H: \{0,1\}^* \longrightarrow \mathbb{Z}_p$ and computes $e(g_1, g_2)^a, g_1^b, g_1^c, g_1^{bc}$. Finally, $\mathcal{B}$ sends the public parameter $\text{PK} = (p, G_1, G_2, G_T, g_1, g_2, e, e(g_1, g_2)^a, g_1^b, g_1^c, g_1^{bc}, H, \{\{A_{i,t,1}, A_{i,t,2}\}_{t\in[n_i]}\}_{i\in[n]})$ to adversary $\mathcal{A}$ and initializes an empty identity table T_2.

4.4.2. Key query. When $\mathcal{A}$ makes user private key query for (id_j, L_j), $\mathcal{B}$ first picks $x_j, \beta \longleftarrow_R \mathbb{Z}_p$ and sets $K_j = x_j$, $K_{0,j} = \sigma_j^{a/b} g_2^{\beta/b} = g_2^{a/b(c+y_j)} g_2^{\beta/b}, K_{1,j} = y_j$. For each $i \in [n]$, $\mathcal{B}$ picks $\lambda_{i,j} \longleftarrow_R \mathbb{Z}_p$ and computes $K_{i,1,j} = g_2^{\beta+\lambda_{i,j}a_{i,t_i}}, K_{i,2,j} = g_2^{\lambda_{i,j}}$. Then, $\mathcal{B}$ sets $\text{SK}_{\text{id}_j,L_j} = (K_j, K_{0,j}, K_{1,j}, \{K_{i,1,j}, K_{i,2,j}\}_{i\in[n]})$ as the corresponding private key and returns it to $\mathcal{A}$. Finally, $\mathcal{B}$ adds (id_j, y_j) to T_2.

4.4.3. Key forgery. Adversary $\mathcal{A}$ outputs a user private key SK^*.

If $\mathcal{A}$ wins the above traceability game, then $\text{Trace}(\text{SK}^*, \text{PK}, T_2) \notin \{\text{id}_1, \text{id}_2, \ldots, \text{id}_{q_s}, \top\}$. Hence, the key $\text{SK}^* = (K, K_0, K_1, \{K_{i,1}, K_{i,2}, K_{i,3}\}_{i\in[n]})$ satisfies the following conditions: $K_1 \notin \{y_1, y_2, \ldots, y_{q_s}\}$, $K, K_1 \in \mathbb{Z}_p, K_0$, $K_{i,1}$, $K_{i,2} \in G_2$, and $\exists i \in [n]$, s.t. $e(g_1^{bc} g_1^{bK_1}, K_0) \cdot e(A_{i,t_i,1}^{K_1} A_{i,t_i,2}, K_{i,2}) = e(g_1, g_2)^a e(g_1^c g_1^{K_1}, K_{i,1})$.

Assuming that $K_{i,1} = g_2^{\beta_i}, K_{i,2} = g_2^{\lambda_i}$, where $\beta_i \in \mathbb{Z}_p$ and $\lambda_i \in \mathbb{Z}_p$ are unknown, we have $e(g_1^{b(K_1+c)}, K_0) e(g_1^{(K_1+c)a_{i,t_i}}, g_2^{\lambda_i}) = e(g_1, g_2)^a e(g_1^{(K_1+c)}, g_2^{\beta_i})$, and then $e(g$

$_1, K_0^{(K_1+c)}) = e(g_1, g_2^{a+(K_1+c)\beta_i-\lambda_i(K_1+c)a_{i,t_i}})$. Therefore, $K_0^b = g_2^{(a+(K_1+c)\beta_i-\lambda_i(K_1+c)a_{i,t_i})/(K_1+c)} = g_2^{a/(K_1+c)} g_2^{\beta_i} g_2^{-\lambda_i a_{i,t_i}} = g_2^{a/(K_1+c)} K_{i,1} K_{i,2}^{-a_{i,t_i}}$.

Finally, $\mathcal{B}$ computes $\sigma = [K_0^b K_{i,2}^{a_{i,t_i}}/K_{i,1}]^{1/a} = g_2^{1/(K_1+c)}$ and then obtains a valid signature σ on message K_1, where $K_1 \notin \{y_1, y_2, \ldots, y_{q_s}\}$. Hence, if $\mathcal{A}$ has advantage ε in the traceability game, then $\mathcal{B}$ can forge a valid BB basic signature scheme with advantage ε under a weak chosen message attack.

5. Our eTR-HABKS System

In this section, we describe our enhanced TR-HABKS system based on our TR-HABKS scheme in Section 4. Different from the TR-HABKS scheme, the tracing algorithm in this system is public traceable and does not require any identity table. In addition, the efficiency comparison shows that the storage overhead of the eTR-HABKS system is much smaller than that of the TR-HABKS scheme.

5.1. Concrete System

5.1.1. System Initialization. In this phase, TA generates the system parameter, the master key for himself, and an identity table for revocation.

TA first runs $\mathcal{G}(\lambda)$ to obtain $(p, G_1, G_2, G_T, g_1, g_2, e)$. For each $i \in [n]$, TA chooses random exponents $\{a_{i,t} \in \mathbb{Z}_p\}_{t\in[n_i]}$ and computes $\{A_{i,t} = g_1^{a_{i,t}}\}_{t\in[n_i]}$. Then, TA picks $a, b, c, d \longleftarrow_R \mathbb{Z}_p$ and a one-way hash function $H: \{0,1\}^* \longrightarrow \mathbb{Z}_p$. Next, TA sets $\mathrm{MK} = (a, b, c, d, \{\{a_{i,t}\}_{t\in[n_i]}\}_{i\in[n]})$ as his master key and publishes the public parameter $\mathrm{PK} = (p, G_1, G_2, G_T, g_1, g_2, e, e(g_1,g_2)^a, g_1^b, g_1^c, g_1^d, g_1^{bc}, g_1^{bd}, H, \{\{A_{i,t}\}_{t\in[n_i]}\}_{i\in[n]})$. Finally, TA creates an empty identity table T_1.

5.1.2. User Registration. In this phase, TA uses his master key to generate the private keys for the registered DUs and DO.

When DU wants to join the system, he submits his identity $\mathrm{id} \in \mathbb{Z}_p$ and attributes set $L = \{v_{i,t_i}\}_{i\in[n]}$ to TA to apply for his private key. TA first picks $x_{\mathrm{id}}, r, \beta \longleftarrow_R \mathbb{Z}_p$ and sets $K = x_{\mathrm{id}}, K_0 = g_2^{a/b(c+\mathrm{id}+dr)} g_2^{\beta/b}, K_1 = \mathrm{id}, K_2 = r$. For each $i \in [n]$, TA picks $\lambda_i \longleftarrow_R \mathbb{Z}_p$ and computes $K_{i,1} = g_2^{\beta+\lambda_i a_{i,t_i}}, K_{i,2} = g_2^{\lambda_i}, K_{i,3} = g_2^{(c+dr)\lambda_i}$. TA sets $\mathrm{SK}_{\mathrm{id},L} = (K, K_0, K_1, K_2, \{K_{i,1}, K_{i,2}, K_{i,3}\}_{i\in[n]})$ as the user private key and sends it to the corresponding DU. Then, TA picks $\alpha \longleftarrow_R \mathbb{Z}_p$ and sets $\mathrm{SK}_o = \alpha$ as the data owner private key and sends it to DO. Finally, TA computes $C_{\mathrm{id}} = e(g_1, g_2)^{-a\alpha x_{\mathrm{id}}}$, stores $(\mathrm{id}, C_{\mathrm{id}})$ in the identity table T_1, and sends T_1 to CS.

5.1.3. Secure Index Generation. In this phase, DO uses his private key to generate a secure index for each file and outsources all the files and indexes in the cloud.

When DO wants to share a file with the specific data users, he extracts a keyword $\omega \in \{0,1\}^*$ from the file and encrypts the keyword ω under an access policy $P = \{P_i\}_{i\in[n]}$. DO first computes $C = e(g_1, g_2)^{a\alpha}, C_0 = g_1^{b\alpha/H(\omega)}, C_1 = g_1^{bc\alpha/H(\omega)}, C_2 = g_1^{bd\alpha/H(\omega)}$. For each $i \in [n]$, DO chooses $\alpha_i \longleftarrow_R \mathbb{Z}_p$ such that $\sum_{i\in[n]}\alpha_i = \alpha$, computes $C_{i,1} = g_1^{\alpha_i}$, $C_{i,2} = g_1^{c\alpha_i}, C_{i,3} = g_1^{d\alpha_i}$, and sets $C_{i,t,2}$ for each $t \in [n_i]$ as follows: if $v_{i,t} \in P_i$, it sets $C_{i,t,2} = A_{i,t}^{\alpha_i}$; otherwise, it sets $C_{i,t,2}$ as a random element in G_1. Finally, DO stores the encrypted index $\mathrm{CT} = \left(C, C_0, C_1, C_2, \left\{C_{i,1}, C_{i,2}, C_{i,3}, \{C_{i,t,2}\}_{t\in[n_i]}\right\}_{i\in[n]}\right)$ in the cloud.

5.1.4. Search Token Generation. In this phase, DU generates a search token for a keyword $\omega' \in \{0,1\}^*$, and sends the search token to CS for the data retrieval request.

DU first picks $s \longleftarrow_R \mathbb{Z}_p$, computes $\mathrm{tok}_0 = K_0^{H(\omega')s}, \mathrm{tok} = K + s, \mathrm{tok}_1 = K_1, \mathrm{tok}_2 = K_2$. For each $i \in [n]$, DU computes $T_{i,1} = K_{i,1}^s, T_{i,2} = K_{i,2}^s, T_{i,3} = K_{i,3}^s$. Finally, DU sets the search token $\mathrm{TK}_{\mathrm{id},L} = (\mathrm{tok}_0, \mathrm{tok}, \mathrm{tok}_1, \mathrm{tok}_2, \{T_{i,1}, T_{i,2}, T_{i,3}\}_{i\in[n]})$.

5.1.5. Data Retrieval. In this phase, CS uses the token to search the data in the cloud and responds the search results to DU.

When CS receives the retrieval request and the search token $\mathrm{TK}_{\mathrm{id},L} = (\mathrm{tok}_0, \mathrm{tok}, \mathrm{tok}_1, \mathrm{tok}_2, \{T_{i,1}, T_{i,2}, T_{i,3}\}_{i\in[n]})$ from DU, it first searches the entry $(\mathrm{id}, C_{\mathrm{id}})$ in T_1. If no such entry exists, CS returns error symbol $\perp$ and aborts; otherwise, CS obtains C_{id} from T_1 and then runs the following search algorithm. *If Algorithm.* If $L = \{v_{i,t_i}\}_{i\in[n]}$, it computes $E = \prod_{i\in[n]}(e(C_{i,1}^{\mathrm{tok}_1} C_{i,2} C_{i,3}^{\mathrm{tok}_2}, T_{i,1})/e(C_{i,t_i,2}, T_{i,2}\mathrm{tok}_1 T_{i,3})) = (g_1, g_2)^{\alpha\beta s(c+\mathrm{id}+dr)}$. Finally, CS returns 1 if $\mathrm{EC}^{\mathrm{tok}} C_{\mathrm{id}} = e(C_0^{\mathrm{tok}_1} C_1 C_2^{\mathrm{tok}_2}, \mathrm{tok}_0)$ and 0 otherwise.

5.1.6. User Tracing. In this phase, TA traces the malicious user who sales his private key $\mathrm{SK}_{\mathrm{id},L}$ on the Internet and outputs the malicious user's identity.

TA first checks whether $\mathrm{SK}_{\mathrm{id},L}$ is a well-formed key. If the private key is not in the form of $\mathrm{SK}_{\mathrm{id},L} = (K, K_0, K_1, K_2\{K_{i,1}, K_{i,2}, K_{i,3}\}_{i\in[n]})$, it returns $\top$ and aborts;

otherwise, it runs the following key sanity check algorithm. $K, K_1, K_2 \in \mathbb{Z}_p, K_0, K_{i,1}, K_{i,2}, K_{i,3} \in G_2$, $\exists i \in [n]$, s.t.

$$\begin{aligned} e\left(g_1^c g_1^{dK_2}, K_{i,2}\right) &= e\left(g_1, K_{i,3}\right), \\ e\left(g_1^{bc} g_1^{bK_1} g_1^{b\,dK_2}, K_0\right) e\left(A_{i,t_i}, K_{i,2}^{K_1} K_{i,3}\right) &= e\left(g_1, g_2\right)^a e\left(g_1^c g_1^{K_1} g_1^{dK_2}, K_{i,1}\right). \end{aligned} \tag{5}$$

If $\mathrm{SK}_{\mathrm{id},L}$ does not pass the above check, it returns $\top$ and aborts; otherwise, it returns K_1 as the corresponding user identity.

5.1.7. User Revocation. In this phase, TA revokes the search permissions of the malicious users. When TA obtains the malicious user identity id, he updates table T_1 by deleting the entry $(\mathrm{id}, C_{\mathrm{id}})$ and sends the new table T_1 to CS.

5.2. Correctness Proof. The correctness of our TR-HABKS scheme is proved as follows. If the user attributes $L = \{v_{i,t_i}\}_{i\in[n]}$ satisfy the access policy $P = \{P_i\}_{i\in[n]}$, we have $v_{i,t_i} \in P_i$ and $C_{i,t_i,2} = A_{i,t_i}^{\alpha_i}$ for each $i \in [n]$. Then,

$$\begin{aligned} &\prod_{i\in[n]} \frac{e\left(C_{i,1}^{\mathrm{tok}_1} C_{i,2} C_{i,3}^{\mathrm{tok}_2}, T_{i,1}\right)}{e\left(C_{i,t_i,2}, T_{i,2}^{\mathrm{tok}_1} T_{i,3}\right)} \\ &= \prod_{i\in[n]} \frac{e\left(g_1^{\mathrm{id}\cdot\alpha_i} g_1^{c\alpha_i} g_1^{d\cdot r\alpha_i}, K_{i,1}^s\right)}{e\left(A_{i,t_i}^{\alpha_i}, K_{i,2}^{\mathrm{id}\cdot s} K_{i,3}^s\right)} \\ &= \prod_{i\in[n]} \frac{e\left(g_1^{\mathrm{id}+c+dr}, g_2^{\beta+\lambda_i a_{i,t_i}}\right)^{s\alpha_i}}{e\left(g_1^{a_{i,t_i}}, g_2^{\lambda_i(\mathrm{id}+c+dr)}\right)^{s\alpha_i}} \\ &= \prod_{i\in[n]} \frac{e\left(g_1^{\mathrm{id}+c+dr}, g_2^{\beta}\right)^{s\alpha_i} e\left(g_1^{\mathrm{id}+c+dr}, g_2^{\lambda_i a_{i,t_i}}\right)^{s\alpha_i}}{e\left(g_1^{a_{i,t_i}}, g_2^{\lambda_i(\mathrm{id}+c+dr)}\right)^{s\alpha_i}} \\ &= \prod_{i\in[n]} e\left(g_1, g_2\right)^{\beta s(\mathrm{id}+c+dr)\alpha_i} \\ &= e\left(g_1, g_2\right)^{\beta s(\mathrm{id}+c+dr)\sum_{i\in[n]}\alpha_i} \\ &= e\left(g_1, g_2\right)^{\alpha\beta s(\mathrm{id}+c+dr)}. \end{aligned} \tag{6}$$

If the user id is in the table T_1, then CS has the corresponding $C_{\mathrm{id}} = e(g_1, g_2)^{-a\alpha x_{\mathrm{id}}}$. Therefore,

$$\begin{aligned} &\mathrm{EC}^{\mathrm{tok}} C_{\mathrm{id}} \\ &= e\left(g_1, g_2\right)^{\alpha\beta s(\mathrm{id}+c+dr)} e\left(g_1, g_2\right)^{a\alpha\left(x_{\mathrm{id}}+s\right)} e\left(g_1, g_2\right)^{-a\alpha x_{\mathrm{id}}} \\ &= e\left(g_1, g_2\right)^{\alpha\beta s(\mathrm{id}+c+dr)} e\left(g_1, g_2\right)^{a\alpha s}. \end{aligned} \tag{7}$$

In this case, if $\omega = \omega'$, we have

$$\begin{aligned} &e\left(C_0^{\mathrm{tok}_1} C_1 C_2^{\mathrm{tok}_2}, \mathrm{tok}_0\right) \\ &= e\left(g_1^{(\mathrm{id}\cdot b\alpha)/H(\omega)} g_1^{bc\alpha/H(\omega)} g_1^{r\cdot b\ d\alpha/H(\omega)}, K_0^{H(\omega')s}\right) \\ &= e\left(g_1^{b\alpha(\mathrm{id}+c+dr)}, g_2^{a/b(\mathrm{id}+c+dr)} g_2^{\beta/b}\right)^s \\ &= e\left(g_1^{b\alpha(\mathrm{id}+c+dr)}, g_2^{a/b(\mathrm{id}+c+dr)}\right)^s e\left(g_1^{b\alpha(i\ d+c+dr)}, g_2^{\beta/b}\right)^s \\ &= e\left(g_1, g_2\right)^{a\alpha s} e\left(g_1, g_2\right)^{\alpha\beta s(\mathrm{id}+c+dr)} \\ &= \mathrm{EC}^{\mathrm{tok}} C_{\mathrm{id}}. \end{aligned} \tag{8}$$

The security proofs of our eTR-HABKS scheme are almost the same as that in Section 4, so we omit the details here.

5.3. Comparison. Table 2 compares the storage costs of our schemes with that of QLSZ scheme [18]. The length of the public parameter/ciphertext of all three schemes increases linearly with $\sum_{i\in[n]} n_i$, where n is the total number of attributes in the system and n_i is the number of possible values for attribute index i. Compared with QLSZ scheme, the public key and ciphertext size of our TR-HABKS scheme have almost doubled, but the public key and ciphertext size of our eTR-HABKS scheme are only increased by 4 and $2 + n$ elements, respectively. The user private key/token size of all schemes grows linearly with the total number of attributes, and the user private key/token of the eTR-HABKS scheme is about 1.5 times as long as that of other schemes. Note that n is far less than $\sum_{i\in[n]} n_i$, so the system storage overhead of the eTR-HABKS scheme is much less than that of the TR-HABKS scheme, although the user storage overhead of the eTR-HABKS scheme is slightly greater. In addition, the eTR-HABKS scheme only needs to maintain an identity table T_1 for revocation but does not require any identity table for tracing, which makes our eTR-HABKS scheme more practical. Figure 2 illustrates the system storage overhead for tracing (including the public parameter and the storage for tracing) in our TR-HABKS and eTR-HABKS schemes. We set the group element size to 160 bits and the random number and identity size to 1024 bits, and $\sum_{i\in[n]} n_i = 100$. From Figure 2, it is easy to see that the system storage overhead for tracing in our eTR-HABKS scheme is constant and significantly smaller than that grows linearly with the number of users in TR-HABKS scheme.

Table 3 gives a computation cost comparison that ignores nondominant operations in the schemes. E_1, E_2, and E_T denote an exponentiation operation in groups G_1, G_2, and G_T, respectively. P is a bilinear pairing operation and $|P_i|$ ($|P_i| \le n_i$) is the number of attribute values in P_i. Let "Trace (max)" and "Trace (min)" denote the maximum and

Table 2: Storage cost comparison.

Scheme	[18]	TR-HABKS	eTR-HABKS
Public parameter size	$4 + \sum_{i\in[n]} n_i$	$6 + 2\sum_{i\in[n]} n_i$	$8 + \sum_{i\in[n]} n_i$
User private key size	$2 + 2n$	$3 + 2n$	$4 + 3n$
Ciphertext size	$2 + n + \sum_{i\in[n]} n_i$	$3 + 2n + 2\sum_{i\in[n]} n_i$	$4 + 3n + \sum_{i\in[n]} n_i$
Token size	$2 + 2n$	$3 + 2n$	$4 + 3n$
The storage for tracing	—	$\lvert T_2\rvert$	0
The storage for revocation	—	$\lvert T_1\rvert$	$\lvert T_1\rvert$

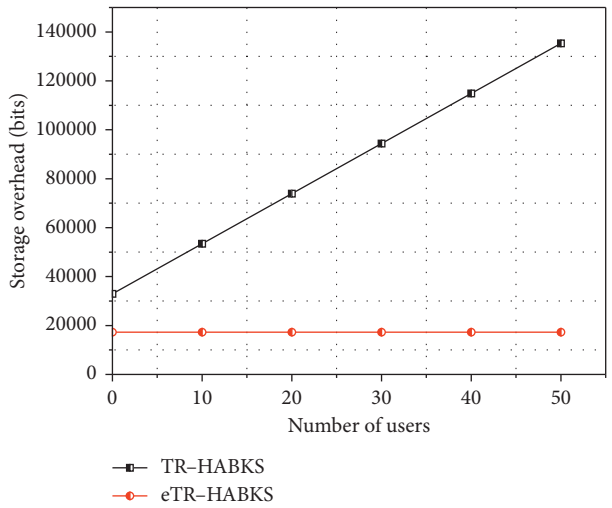

Figure 2: System storage overhead for tracing.

Table 3: Computation cost comparison.

Scheme	[18]	TR-HABKS	eTR-HABKS
Setup	$(1 + \sum_{i\in[n]} n_i)E_1 + E_T$	$(3 + 2\sum_{i\in[n]} n_i)E_1 + E_T$	$(5 + \sum_{i\in[n]} n_i)E_1 + E_T$
KeyGen	$(1 + 2n)E_2 + E_T$	$(1 + 2n)E_2 + E_T$	$(1 + 3n)E_2 + E_T$
Enc	$(1 + n + \sum_{i\in[n]} \lvert P_i\rvert)E_1 + E_T$	$(2 + 2n + 2\sum_{i\in[n]} \lvert P\rvert_i)E_1 + E_T$	$(3 + 3n + \sum_{i\in[n]} \lvert P_i\rvert)E_1 + E_T$
TokenGen	$(1 + 2n)E_2$	$(1 + 2n)E_2$	$(1 + 3n)E_2$
Search	$(1 + 2n)P + E_1$	$(1 + 2n)P + (2 + 2n)E_1$	$(1 + 2n)P + (3 + 2n)E_1 + nE_2$
Trace (max)	—	$(1 + 2n)P + (2 + n)E_1$	$(1 + 4n)P + 4E_1 + nE_2$
Trace (min)	—	$3P + 3E_1$	$5P + 4E_1 + E_2$
Revoke	—	0	0

minimum computation cost for successful tracing, respectively. From Table 3, we can see that the computation cost in the setup and encryption algorithms of the eTR-HABKS scheme is almost the same as that of QLSZ scheme, but it is obviously smaller than that of the TR-HABKS scheme. All three schemes have the same level of computation overhead in key generation and token generation algorithms. Compared with QLSZ scheme, our TR-HABKS and eTR-HABKS schemes do not add any computation overhead to achieve user revocation but add more computation overhead to realize user accountability. However, the increased computation burden has little effect on the performance of our eTR-HABKS system, because the search and tracing algorithms can be executed by the cloud with powerful computing capability.

6. Conclusion

In this paper, we first presented a new privacy-preserving ABKS construction for cloud-assisted IoT and then proved that it is selectively secure and fully traceable in the generic bilinear group model. We also proposed another ABKS construction with public traceability and showed that it is more efficient than the first construction. In short, our two constructions not only reduce privacy leakage by hiding access policies but also prevent private key abuse by tracing and revoking malicious users. As our schemes are designed for just one-owner setting, we aim to construct a traceable and revocable ABKS scheme with policy protection in multiowner setting in the future.

Data Availability

No data were used to support the findings of this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant nos. 61802243 and 11801345), the Natural Science Foundation of Shaanxi Province (Grant nos. 2019JQ-273 and 2020JM-288), and the Key Research and Development Program in Industry Field of Shaanxi Province (Grant no. 2019GY-013).

References

[1] L. Atzori, A. Iera, and G. Morabito, "The internet of things: a survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[2] Y. Zhang, D. Zheng, and R. H. Deng, "Security and privacy in smart health: efficient policy-hiding attribute-based access control," *IEEE Internet of Things Journal*, vol. 5, no. 3, pp. 2130–2145, 2018.

[3] L. Zhao and X. Dong, "An industrial internet of things feature selection method based on potential entropy evaluation criteria," *IEEE Access*, vol. 6, pp. 4608–4617, 2018.

[4] J. Xiong, R. Bi, M. Zhao, J. Guo, and Q. Yang, "Edge-assisted privacy-preserving raw data sharing framework for connected autonomous vehicles," *IEEE Wireless Communications*, vol. 27, no. 3, pp. 24–30, 2020.

[5] Y. Liu, X. Ma, L. Shu et al., "Internet of things for noise mapping in smart cities: state of the art and future directions," *IEEE Network*, vol. 34, no. 4, pp. 112–118, 2020.

[6] A. Sajid, H. Abbas, and K. Saleem, "Cloud-assisted iot-based scada systems security: a review of the state of the art and future challenges," *IEEE Access*, vol. 4, pp. 1375–1384, 2016.

[7] K. Tange, M. De Donno, X. Fafoutis, and N. Dragoni, "A systematic survey of industrial internet of things security: requirements and fog computing opportunities," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 4, pp. 2489–2520, 2020.

[8] J. Xiong, J. Ren, L. Chen et al., "Enhancing privacy and availability for data clustering in intelligent electrical service of iot," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 1530–1540, 2018.

[9] X. Han, L. Wang, S. Xu, D. Zhao, and G. Liu, "Recognizing roles of online illegal gambling participants: an ensemble learning approach," *Computers & Security*, vol. 87, Article ID 101588, 2019.

[10] Y. Tian, Z. Wang, J. Xiong, and J. Ma, "A blockchain-based secure key management scheme with trustworthiness in dwsns," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 9, pp. 6193–6202, 2020.

[11] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Proceedings of the 21st Annual International Cryptology Conference on Advances in Cryptology*, pp. 213–229, Springer, Santa Barbara, CA, USA, August 2001.

[12] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proceedings of the 2004 International conference on the theory and applications of cryptographic techniques*, pp. 506–522, Interlaken, Switzerland, May 2004.

[13] M. Abdalla, M. Bellare, D. Catalano et al., "Searchable encryption revisited: consistency properties, relation to anonymous ibe, and extensions," *Journal of Cryptology*, vol. 21, no. 3, pp. 350–391, 2008.

[14] Q. Zheng, S. Xu, and G. Ateniese, "Vabks: verifiable attribute-based keyword search over outsourced encrypted data," in *Proceedings of the IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pp. 522–530, IEEE, Toronto, ON, Canada, April 2014.

[15] 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 *Proceedings of the IEEE INFOCOM 2014-IEEE Conference on Computer Communications*, pp. 226–234, IEEE, Toronto, ON, Canada, April 2014.

[16] Q. Huang and H. Li, "An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks," *Information Sciences*, vol. 403-404, pp. 1–14, 2017.

[17] S. Liu, J. Yu, Y. Xiao, Z. Wan, S. Wang, and B. Yan, "Bc-sabe: blockchain-aided searchable attribute-based encryption for cloud-iot," *IEEE Internet of Things Journal*, vol. 7, no. 9, pp. 7851–7867, 2020.

[18] S. Qiu, J. Liu, Y. Shi, and R. Zhang, "Hidden policy ciphertext-policy attribute-based encryption with keyword search against keyword guessing attack," *Science China Information Sciences*, vol. 60, no. 5, Article ID 052105, 2017.

[19] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 457–473, Aarhus, Denmark, May 2005.

[20] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*, pp. 89–98, Alexandria, VA, USA, October 2006.

[21] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the 2007 IEEE symposium on security and privacy (SP'07)*, pp. 321–334, IEEE, 2007.

[22] B. Waters, "Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization," in *International Workshop on Public Key Cryptography*, pp. 53–70, Berkeley, CA, USA, May 2011.

[23] N. Attrapadung and H. Imai, "Dual-policy attribute based encryption," in *Proceedings of the 2009 International Conference on Applied Cryptography and Network Security*, pp. 168–185, Paris-Rocquencourt, France, June 2009.

[24] S. Xu, Y. Li, R. Deng, Y. Zhang, X. Luo, and X. Liu, "Lightweight and expressive fine-grained access control for healthcare internet-of-things," *IEEE Transactions on Cloud Computing*, 2019.

[25] T. Nishide, K. Yoneyama, and K. Ohta, "Attribute-based encryption with partially hidden encryptor-specified access structures," in *Proceedings of the 2008 International conference on applied cryptography and network security*, pp. 111–129, New York, NY, USA, June 2008.

[26] J. Lai, R. H. Deng, and Y. Li, "Expressive cp-abe with partially hidden access structures," in *Proceedings of the 7th ACM symposium on information, computer and communications security*, pp. 18-19, Seoul, South Korea, May 2012.

[27] A. Beimel, *Secure schemes for secret sharing and key distribution*, Ph.D. Dissertation, Technion-Israel Institute of Technology, Haifa, Israel, 1996.

[28] K. Yang, Q. Han, H. Li, K. Zheng, Z. Su, and X. Shen, "An efficient and fine-grained big data access control scheme with privacy-preserving policy," *IEEE Internet of Things Journal*, vol. 4, no. 2, pp. 563–571, 2016.

[29] J. Sun, H. Xiong, X. Liu, Y. Zhang, X. Nie, and R. H. Deng, "Lightweight and privacy-aware fine-grained access control for iot-oriented smart health," *IEEE Internet of Things Journal*, vol. 7, no. 7, pp. 6566–6575, 2020.

[30] M. J. Hinek, S. Jiang, R. S. Naini, and S. F. Shahandashti, "Attribute-based encryption without key cloning," *International Journal of Applied Cryptography*, vol. 2, no. 3, pp. 250–270, 2012.

[31] Z. Liu, Z. Cao, and D. S. Wong, "White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 1, pp. 76–88, 2012.

[32] Z. Liu, Z. Cao, and D. S. Wong, "Blackbox traceable cp-abe: How to catch people leaking their keys by selling decryption devices on ebay," in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pp. 475–486, Hangzhou China, May 2013.

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

[34] Z. Ying, Y. Si, J. Ma, X. Liu, and S. Xu, "Fhpt: fine-grained ehr sharing in e-healthcare cloud with hidden policy and traceability," in *Proceedings of the 2020 GLOBECOM 2020-2020 IEEE Global Communications Conference*, pp. 1–6, IEEE, Taipei, Taiwan, December 2020.

[35] S. Xu, J. Ning, Y. Li et al., "Match in my way: fine-grained bilateral access control for secure cloud-fog computing," *IEEE Transactions on Dependable and Secure Computing*, 2020.

[36] B. Qin, Q. Zhao, D. Zheng, and H. Cui, "(Dual) server-aided revocable attribute-based encryption with decryption key exposure resistance," *Information Sciences*, vol. 490, pp. 74–92, 2019.

[37] S. Xu, G. Yang, Y. Mu, and X. Liu, "A secure iot cloud storage system with fine-grained access control and decryption key exposure resistance," *Future Generation Computer Systems*, vol. 97, pp. 284–294, 2019.

[38] J. W. Byun, H. S. Rhee, H.-A. Park, and D. H. Lee, "Off-line keyword guessing attacks on recent keyword search schemes over encrypted data," in *Proceedings of the 2006 Workshop on secure data management*, pp. 75–83, Seoul, South Korea, September 2006.

[39] Y. Miao, Q. Tong, R. Deng, K.-K. R. Choo, X. Liu, and H. Li, "Verifiable searchable encryption framework against insider keyword-guessing attack in cloud storage," *IEEE Transactions on Cloud Computing*, 2020.

[40] Y. Miao, Q. Tong, K.-K. R. Choo, X. Liu, R. H. Deng, and H. Li, "Secure online/offline data sharing framework for cloud-assisted industrial internet of things," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8681–8691, 2019.

[41] Y. Miao, X. Liu, K.-K. R. Choo et al., "Privacy-preserving attribute-based keyword search in shared multi-owner setting," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 3, 2019.

[42] J. Sun, H. Xiong, X. Nie, Y. Zhang, and P. Wu, "On the security of privacy-preserving attribute-based keyword search in shared multi-owner setting," *IEEE Transactions on Dependable and Secure Computing*, 2019.

[43] Z. Liu, Y. Liu, J. Xu, and B. Wang, "Privacy-preserving attribute-based multi-keyword search encryption scheme with user tracing," in *Proceedings of the 2019 International Symposium on Cyberspace Safety and Security*, pp. 382–397, Guangzhou, China, December 2019.

[44] D. Boneh and X. Boyen, "Short signatures without random oracles and the sdh assumption in bilinear groups," *Journal of Cryptology*, vol. 21, no. 2, pp. 149–177, 2008.

[45] D. Boneh, X. Boyen, and E.-J. Goh, "Hierarchical identity based encryption with constant size ciphertext," in *Proceedings of the 2005 Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 440–456, Aarhus, Denmark, May 2005.