IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 28, 2019, accepted March 18, 2019, date of publication March 21, 2019, date of current version December 26, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2906726

Efficient Attribute-Based Access Control With Authorized Search in Cloud Storage

JIALU HAO[1], (Student Member, IEEE), JIAN LIU[1], HUIMEI WANG[1], LINGSHUANG LIU[2], MING XIAN[1], AND XUEMIN SHEN[3], (Fellow, IEEE)

[1]College of Electronic Science and Technology, National University of Defense Technology, Changsha 410073, China
[2]Science and Technology on Reactor System Design Technology Laboratory, Nuclear Power Institute of China, Chengdu 610213, China
[3]Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON N2L 3G1, Canada

Corresponding author: Huimei Wang (freshcdwhm@163.com)

This work was supported by the National Natural Science Foundation of China under Grant No. 61801489.

ABSTRACT Attribute-based encryption has been widely employed to achieve data confidentiality and fine-grained access control in cloud storage. To enable users to identify accessible data in numerous dataset, clear attributes should be appended to the ciphertext, which results in the exposure of attribute privacy. In this paper, we propose an efficient attribute-based access control with authorized search scheme (EACAS) in cloud storage by extending the anonymous key-policy attribute-based encryption (AKP-ABE) to support fine-grained data retrieval with attribute privacy preservation. Specifically, by integrating the key delegation technique into AKP-ABE, EACAS enables data users to customize search policies based on their access policies, and generate the corresponding trapdoor using the secret key granted by the data owner to retrieve their interesting data. In addition, a virtual attribute with no semantic meaning is utilized in data encryption and trapdoor generation to empower the cloud to perform an attribute-based search on the outsourced ciphertext without knowing the underlying attributes or outsourced data. The data owners can achieve fine-grained access control on their outsourced data, and the data users are flexible to search their interesting data based on protected attributes through customizing the search policies. Finally, we demonstrate that EACAS is more efficient than existing solutions on computation and storage overheads.

INDEX TERMS Access control, authorized search, cloud storage, data sharing, key-policy attribute-based encryption.

I. INTRODUCTION

Cloud storage as one of the most popular cloud-based applications supplies users with scalable and elastic storage resources for remote data sharing, which dramatically reduces the local cost on data management and maintenance [1]–[3]. However, once the data is outsourced to the cloud, the security and privacy threats become huge concerns for data owners as they lose the physical control over their data [4], [5]. Moreover, the frequently happened data leakage incidents undermine the trust on the cloud service provider, which significantly impedes the wide adoption of outsourced cloud storage [6], [7]. Traditional one-to-one encryption is able to protect data confidentiality [8], but it is quite incompetent for data owners sharing their data with authorized users efficiently and flexibly. As well known, attribute-based encryption (ABE) [9] can be used to achieve fine-grained access control and protect data confidentiality simultaneously, and key-policy attribute-based encryption (KP-ABE) [10] enables the data owner to label each ciphertext with a set of descriptive attributes, and generate the private key that is related to an access policy to specify which type of ciphertext can be decrypted. After acquiring this private key, the data user can decrypt the specified ciphertext shared by the data owner. With the property of designated data sharing, KP-ABE has been widely used in electronic medical record systems and remote cloud storage [11], [12].

To enable remote data access, the data owners have to explicitly append the attributes to the ciphertext and then upload the attributes and ciphertext to the cloud; otherwise, the data users cannot identify their accessible data. Although this simple approach is quite popular in conventional KP-ABE schemes [9], [10], the public attributes may cause the privacy leakage, indicating that anyone who obtains the ciphertext can infer some secret information

The associate editor coordinating the review of this manuscript and approving it for publication was Kuan Zhang.

about the data content. For example, the medical record of a patient is encrypted with the following attributes, {**Affiliation**: *Hospital A*, **Department**: *Cardiology*, **Gender**: *Male*} and uploaded to the cloud. In such situation, anyone who has the ciphertext is able to deduce from the public attributes that the patient may suffer from heart disease, even they cannot access the plaintext. Therefore, it is necessary to introduce anonymous KP-ABE to preserve attribute privacy. However, if the attribute information is hidden, the data users cannot identify and retrieve their accessible data shared by the data owners.

To achieve fine-gained access control and flexible data retrieval simultaneously, a general scheme [13] has been proposed based on the generic construction [14] that combines ABE with expressive searchable encryption (ESE), i.e., encrypt the data with ABE and encrypt the corresponding keywords with ESE. Unfortunately, simply combining them suffers from the following problems. Firstly, any user, even unauthorized one, is able to search the ciphertext, even if he cannot decrypt to obtain data. Secondly, the double encryption (ABE+ESE) brings about large computation and storage overheads. Thirdly, the trapdoor obtained from the data owner for ciphertext search is one-time, which means that the data user has to request a new trapdoor for every data retrieval. Recently, some authorized keyword search (AKS) schemes [15], [16] were proposed to allow only authorized data user to perform ciphertext search, but they are either restricted with less expressive search policy or inefficient for practical applications.

In this paper, we propose an efficient attribute-based access control with authorized search scheme (EACAS) in cloud storage by extending the anonymous key-policy attribute-based encryption (AKP-ABE). Our EACAS is characterized by employing the key delegation to empower the data users to independently generate the trapdoor for ciphertext retrieving, and a virtual attribute is introduced in both ciphertext and trapdoor to protect data confidentiality against the semi-honest cloud server while performing data search. The main contributions of this paper can be summarized as threefold.

1) Firstly, we propose AKP-ABE with partially hidden attributes based on the expressive keyword search (EKS) scheme proposed by Cui *et al.* [17]. In AKP-ABE, the attribute values in the attribute set are protected for preventing attribute privacy leakage, and the linear splitting technique [18] is utilized to protect the attribute values against offline guessing attack. The AKP-ABE is proved secure under the q-2 Decisional Bilinear Diffie-Hellman (DBDH) assumption [9] and the Decisional Linear (D-Linear) assumption [19].
2) Secondly, we propose EACAS by extending AKP-ABE to support fine-grained access control with authorized search over the cloud data. Specifically, we use the key delegation technique to empower the data user to generate trapdoor associated with the search policy only based on his secret key. Additionally, we bind a virtual attribute in the data encryption and trapdoor generation process to prevent the cloud server decrypting the ciphertext while performing ciphertext search on behalf of the data users.
3) Thirdly, we analyze the property, security and efficiency of EACAS, and implement it by using the rapidly prototyping tool called Charm [20]. The extensive experiments demonstrate that EACAS is more efficient than existing solution on computation and storage overheads and is practical to be implemented in cloud storage.

The remainder of this paper is organized as follows. We first review some related work and preliminaries in Section II and III. In Section IV, we present the AKP-ABE primitive. The detailed construction of EACAS can be found in Section V, followed by the property and security discussions in Section VI. The performance evaluation is given in Section VII. Finally, we conclude the paper in Section VIII.

II. RELATED WORK

We discuss the existing works related to our proposed scheme, including ABE, PEKS and AKS.

A. ATTRIBUTE-BASED ENCRYPTION (ABE)

ABE mainly includes two forms: ciphertext-policy ABE (CP-ABE) [21] and KP-ABE [10]. In CP-ABE, the data is encrypted under a specified access policy, and only the users possessing attributes that satisfy the access policy are able to decrypt the ciphertext. While in KP-ABE, the data is encrypted under several attributes, and the user is assigned with an access policy [22]. With the above properties, ABE soon became popular in the outsourced data access control systems [23], [24]. However, most of the existing schemes expose attribute information in the ciphertext, which may incur data or user privacy leakage, thus the research on anonymity of ABE is also necessary [25]. Some anonymous CP-ABE schemes [26] have been proposed to prevent the unauthorized user presuming attribute information from the access policy attached to the ciphertext. Nishide *et al.* [19] proposed the first construction of anonymous CP-ABE by partially hiding the attributes in access policy, in which the attribute is split into an attribute name and multiple attribute values, and only the values are concealed. Based on it, some works [27], [28] improved the construction in terms of efficiency and security, but only AND gates are supported in the access policy. Later, Lai *et al.* [29] constructed an anonymous CP-ABE primitive based on the composite-order groups, which supports more expressive access policy. With the same form of the access policy, Cui *et al.* [30] combined the large universe CP-ABE [9] with the linear splitting technique [18] to give a more efficient construction in the prime-order groups. Obviously, the ideas in anonymous CP-ABE can somehow be used to achieve anonymity in KP-ABE.

B. PUBLIC-KEY ENCRYPTION WITH KEYWORD SEARCH (PEKS)

The concept of PEKS was introduced by Boneh *et al.* [31], and two constructions were given in their scheme to support equality queries. Later, many other PEKS schemes with better security or new functionalities are proposed to make it more practical [32], in which the following two directions are mainly included: (1) how to support more expressive and flexible search policy on keywords; and (2) how to resist the offline guessing attack. With regard to the first one, Park *et al.* [33] presented a PEKS construction to support searching with conjunctive keywords. Boneh and Waters [34] proposed a public-key based scheme supporting arbitrary conjunctive queries over the encrypted data. In addition, with the idea of viewing attribute in KP-ABE as keyword, some search schemes [35], [36] supporting arbitrary monotone boolean search policy on the keywords are proposed, but they are constructed in composite-order groups which make them impractical. Recently, Cui *et al.* [17] utilized the large universe KP-ABE [9] to achieve expressive keyword search (EKS) in prime-order groups. The offline guessing attack was introduced by Byun *et al.* [37], and it was demonstrated that the popular PEKS construction proposed in [31] is vulnerable to this kind of attack, since the keywords can be guessed from the trapdoor. As discussed in [38], the vulnerability against the offline guessing attack is an essential feature for searchable encryption in public-key situation. Rhee *et al.* [39] considered a tradeoff solution, where a designated tester is introduced to perform search such that any adversary without the private key assigned to the designated tester cannot launch offline guessing attack.

C. AUTHORIZED KEYWORD SEARCH (AKS)

Sun *et al.* [40] first considered fine-grained search authorization in their attribute-based keyword search scheme, but only single keyword is supported. Shi *et al.* [15] put forward an authorized keyword search (AKS) scheme supporting expressive authorization policies and query predicates, but too much cost is incurred due to the composite-order groups and the data owner need generate trapdoor for each search policy. Jiang *et al.* [16] presented a public key based scheme supporting authorized ciphertext search which only supports single keyword. However, these schemes mainly focus on the authorization of keyword search, but do not consider the ability of decryption. Lately, Cui *et al.* [13] designed a generic attribute-based encryption with expressive and authorized keyword search (ABE-EAKS) scheme, in which ABE is applied to encrypt the data, and EKS is used to encrypt the corresponding keywords. Their scheme can better meet the real demand for data sharing in the cloud storage environment, but the double encryption incurs large computation and storage overheads. Compared with [13], only AKP-ABE primitive is utilized to realize fine-grained access control with authorized search in EACAS, thus the cost is significantly smaller than the cumulative cost of ABE and EKS.

III. PRELIMINARIES

In this section, we briefly review the technical preliminaries closely related to our work.

A. ACCESS STRUCTURE

An access structure [10] on an attribute universe U is a collection $\mathbb{C}$, which includes non-empty sets of attributes. The sets in $\mathbb{C}$ are defined as the authorized sets. In addition, an access structure which satisfies the following requirement is called monotone: if $X \in \mathbb{C}$ and $X \subseteq Y$, then $Y \in \mathbb{C}$. Note that, only monotone access structures are considered in this paper.

B. ACCESS TREE

The access tree structure can be used to represent the access policy [10]. In an access tree $\mathcal{T}$, each non-leaf node represents a threshold gate, and each leaf node is associated with an attribute. For a non-leaf node x, let k_x, num_x denote the threshold value and the number of its children respectively, where $k_x = 1$ means the OR gate and $k_x = num_x$ means the AND gate. Let $par(x)$ be x's parent node, $id(x)$ be the index of x ordered by its parent. $atts(\mathcal{T})$ means the set of attributes (leaves) in $\mathcal{T}$, and $\mathcal{T}_x$ means the subtree with a root node x.

To share a secret α based on an access tree $\mathcal{T}$, a random polynomial p_x is defined for each node x in the top-down manner, where the degree $d_x = k_x - 1$. For the root node r, $p_r(0) = \alpha$. For each non-root node, $p_x(0) = p_{par(x)}(id(x))$. For an attribute (leaf) node i, $p_i(0) = p_{par(i)}(id(i))$ can be seen as the secret share assigned to it.

If an attribute set S satisfies the access tree $\mathcal{T}_x$, we denote that $\mathcal{T}_x(S) = 1$. Specifically, if x is an attribute node, $\mathcal{T}_x(S)$ returns 1 only if $x \in S$. If x is a non-leaf node, $\mathcal{T}_x(S) = 1$ only if at least k_x children nodes of x return 1. In addition, for an access tree with root node r, if $\mathcal{T}_r(S) = 1$, the secret α associated with r can be recovered by combining the shares assigned to the attributes belonging to S in the bottom-up manner using the Lagrange polynomial interpolation technique recursively.

C. KEY-POLICY ATTRIBUTE-BASED ENCRYPTION

Four algorithms are included in the KP-ABE primitive [10].

- $\texttt{Setup}(\xi) \rightarrow (PK, MSK)$. This algorithm takes as input a security parameter ξ, and generates the master secret key MSK as well as the public key PK.
- $\texttt{Encrypt}(PK, M, S) \rightarrow CT$. This algorithm takes as input PK, a message M, and an attribute set S. It generates the ciphertext CT.
- $\texttt{KeyGen}(PK, MSK, \mathcal{AP}) \rightarrow SK$. This algorithm takes as input PK, MSK, and the access policy $\mathcal{AP}$. It generates the secret key SK associated with $\mathcal{AP}$.
- $\texttt{Decrypt}(CT, SK) \rightarrow M/\perp$. This algorithm takes as input CT and SK. It outputs the message M only if the attribute set S of CT satisfies the access policy related to SK, otherwise it outputs $\perp$.

The AKP-ABE primitive consists of the same algorithms, except that the attribute information is hidden in the ciphertext.

IV. ANONYMOUS KP-ABE WITH PARTIALLY HIDDEN ATTRIBUTES

In this section, we propose an anonymous key-policy attribute-based encryption (AKP-ABE) with partially hidden attributes as the main building block of our EACAS based on the expressive keyword search (EKS) scheme proposed by Cui *et al.* [17].

A. DEFINITIONS

An attribute in AKP-ABE is split into two parts: an attribute name and multiple attribute values. More specifically, an attribute in the attribute set S is represented as $[n_x : s_x]$, where n_x means the generic attribute name, and s_x is the corresponding attribute value. We denote N_S as the attribute name set related to S. In addition, an access policy $\mathcal{AP}$ is represented as $(\mathcal{T}, \{[n_x : t_x]\}_{n_x \in atts(\mathcal{T})})$, where $\mathcal{T}$ is an access tree structure taking the attribute name as leaf node, and each attribute name n_x is bound with an attribute value t_x. Furthermore, for the access tree structure $\mathcal{T}$ in the access policy $\mathcal{AP}$, we define $\mathcal{N}$ as a set of minimum subsets of $atts(\mathcal{T})$, where for each attribute name set $N_I \in \mathcal{N}$, $\mathcal{T}(N_I) = 1$ and N_I cannot be smaller.

An attribute set S satisfies an access policy $\mathcal{AP} = (\mathcal{T}, \{[n_x : t_x]\}_{n_x \in atts(\mathcal{T})})$, if there exists an attribute name set N_I in $\mathcal{N}$ such that

$$N_I \subset N_S, \quad \text{and } \forall n_x \in N_I,\ s_x = t_x,$$

where $N_I \subset N_S$ guarantees that the attribute name set N_S related to S satisfies the access tree structure $\mathcal{T}$, and $s_x = t_x$ means the attribute values of the same attribute name n_x are identical in the access policy and attribute set.

Note that, in our scheme, the attribute value s_x in an attribute set S is hidden in the ciphertext to protect the attribute privacy, which means it is embedded into the ciphertext component implicitly, while the corresponding attribute name n_x is public to simplify the matching process in decryption.

B. SECURITY MODEL

The security model with partially hidden attributes for AKP-ABE is described as the following game between a challenger and an adversary.

- **Init.** The adversary commits to the challenger two attribute sets S_0 and S_1.
  Note that the attribute name set is the same for the two attribute sets (e.g. $N_{S_0} = N_{S_1}$*), and there exists at least one different attribute value. Otherwise, one can distinguish the ciphertext from the different attribute name sets, since it is explicitly attached in the ciphertext.*
- **Setup.** The challenger runs the `Setup` algorithm to generate PK and MSK. Then, it sends PK to the adversary and keeps MSK.
- **Phase 1.** The adversary queries the secret key related to an access policy $\mathcal{AP}$ from the challenger. The only restriction is that $\mathcal{AP}$ cannot be satisfied by either S_0 or S_1. Then, the challenger calls the `KeyGen` algorithm to generate SK for the adversary. Multiple queries can be requested by the adversary.
- **Challenge.** The adversary submits two messages M_0 and M_1 with the same size to the challenger. The challenger picks a random bit β in $\{0, 1\}$, and runs the `Encrypt`(PK, M_β, S_β) algorithm to generate the ciphertext CT for the adversary.
- **Phase 2.** Phase 1 is repeated.
- **Guess.** If the adversary outputs a guess β' that equals to β, it wins the game.

In the game, we use $|Pr[\beta' = \beta] - \frac{1}{2}|$ to define the advantage of an adversary.

Definition 1: The AKP-ABE with partially hidden attributes is selectively secure under the chosen plaintext attack if no probabilistic polynomial-time (PPT) adversary has non-negligible advantage in the above security game.

C. AKP-ABE SCHEME

We utilize the linear splitting technique [18] on the ciphertext components to protect the attribute values against offline guessing attack. The concrete construction of the AKP-ABE is as follows.

- `Setup`$(\xi) \rightarrow (PK, MSK)$

With the input of a security parameter ξ, the algorithm first generates a bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{G}_T$ are multiplicative cyclic groups of prime order p, and g is a generator of $\mathbb{G}$. It computes $g_1 = g^{\tau_1}, g_2 = g^{\tau_2}, g_3 = g^{\tau_3}, g_4 = g^{\tau_4}$, where $\alpha, \tau_1, \tau_2, \tau_3, \tau_4 \in Z_p^*$ are random values. Then, it randomly picks three group elements u, h, w from $\mathbb{G}$. The public key PK is produced as

$$PK = \langle g, u, h, w, e(g, g)^\alpha, g_1, g_2, g_3, g_4 \rangle.$$

The master secret key is $MSK = \langle \alpha, \tau_1, \tau_2, \tau_3, \tau_4 \rangle$.

- `Encrypt`$(PK, M, S) \rightarrow CT$

This algorithm takes as input the message M, the public key PK, and the attribute set S. As noted, each attribute in S is represented as $[n_x : s_x]$, where n_x means the generic attribute name, and $s_x \in Z_p^*$ is the corresponding attribute value. It first randomly picks $s \in Z_p^*$, and computes $\widetilde{E} = M \cdot e(g, g)^{\alpha s}, E = g^s$. Then, it picks three random exponents $s_{x,1}, s_{x,2}, z_x \in Z_p^*$ for every attribute in S, and computes

$$E_{x,0} = w^{-s}(u^{s_x}h)^{z_x}, \quad E_{x,1} = g_1^{z_x - s_{x,1}},\ E_{x,2} = g_2^{s_{x,1}},$$
$$E_{x,3} = g_3^{z_x - s_{x,2}}, \quad E_{x,4} = g_4^{s_{x,2}}.$$

Finally, the algorithm outputs the ciphertext as $CT = \langle N_S, \widetilde{E}, E, \{E_{x,0}, E_{x,1}, E_{x,2}, E_{x,3}, E_{x,4}\}_{x \in S} \rangle$. Note that, only the attribute name set N_S is included in the ciphertext to protect the attribute privacy.

- `KeyGen`$(PK, MSK, \mathcal{AP}) \rightarrow SK$

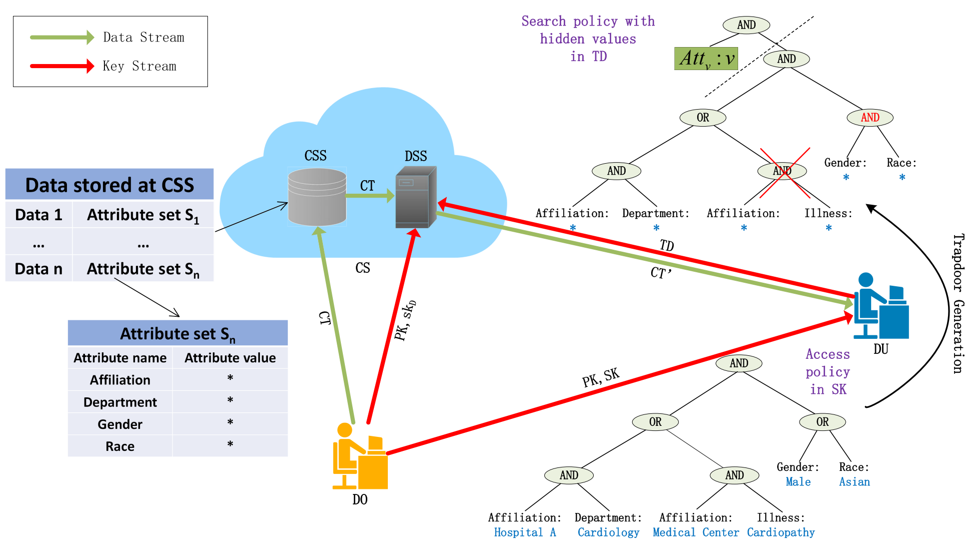


FIGURE 1. System model.

This algorithm takes as input the public key PK, the master secret key MSK, and the access policy $\mathcal{AP}$. The access policy $\mathcal{AP}$ is represented as $(\mathcal{T}, \{[n_x : t_x]\}_{n_x \in atts(\mathcal{T})})$, where $\mathcal{T}$ is an access tree structure taking the attribute name as the leaf node, and each attribute name n_x is bound with an attribute value t_x. It first splits the secret α in MSK based on the access tree $\mathcal{T}$, such that the secret share for the leaf node n_x in $\mathcal{T}$ is $p_x(0)$. Then, for each leaf node, it randomly selects $t_{x,1}, t_{x,2} \in Z_p^*$, and computes

$$D_x = g^{p_x(0)} w^{\tau_1\tau_2 t_{x,1} + \tau_3\tau_4 t_{x,2}},$$
$$D_{x,0} = g^{\tau_1\tau_2\, t_{x,1} + \tau_3\tau_4\, t_{x,2}},$$
$$D_{x,1} = (u^{t_x} h)^{-\tau_2 t_{x,1}}, \quad D_{x,2} = (u^{t_x} h)^{-\tau_1 t_{x,1}},$$
$$D_{x,3} = (u^{t_x} h)^{-\tau_4 t_{x,2}}, \quad D_{x,4} = (u^{t_x} h)^{-\tau_3 t_{x,2}}.$$

Finally, the secret key associated with the access policy $\mathcal{AP}$ is produced as

$$SK = \langle \mathcal{AP}, \quad D_x, D_{x,0}, D_{x,1}, \ D_{x,2}, D_{x,3}, D_{x,4}\}_{n_x \in atts(\mathcal{T})}\rangle.$$

- `Decrypt`$(CT, SK) \rightarrow M/\perp$

This algorithm first checks whether the attribute name set N_S in CT satisfies the access tree structure $\mathcal{T}$ in the access policy $\mathcal{AP}$ of the secret key SK. If not, it terminates with $\perp$. Otherwise, it computes $\mathcal{N}$ from the access tree structure $\mathcal{T}$, where $\mathcal{N}$ means a set of minimum subsets of the attribute names in $atts(\mathcal{T})$ that satisfy $\mathcal{T}$.

Then, it tests whether there exists an attribute name set $N_I \in \mathcal{N}$ can decrypt successfully. Concretely, if $N_I \subset N_S$, for each attribute name $n_x \in N_I$, it computes

$$\begin{aligned} P_x &= e(E, D_x)e(E_{x,0}, D_{x,0})e(E_{x,1}, D_{x,1}) \\ &\quad e(E_{x,2}, D_{x,2})e(E_{x,3}, D_{x,3})e(E_{x,4}, D_{x,4}) \\ &= e(g, g)^{p_x(0)s} e(u, g)^{z_x(s_x - t_x)(\tau_1\tau_2\, t_{x,1} + \tau_3\tau_4\, t_{x,2})}. \end{aligned}$$

If the attribute values in the access policy and the attribute set are matching for the same attribute name n_x (e.g. $s_x = t_x$), then $P_x = e(g, g)^{p_x(0)s}$. Furthermore, if the attribute values are matching for all the attribute names in N_I, $e(g, g)^{\alpha s}$ can be calculated through combining $\{P_x\}_{n_x \in N_I}$ in the bottom-up manner recursively with the Lagrange polynomial interpolation technique. Thus, the message M can be calculated with $\widetilde{E}/e(g, g)^{\alpha s}$. Finally, if no such $N_I \in \mathcal{N}$ exists, the algorithm outputs $\perp$.

D. SECURITY PROOF

Theorem 1: If the q-2 DBDH assumption [9] and the D-Linear assumption [19] hold, AKP-ABE with partially hidden attributes is selectively secure under chosen plaintext attack.

Proof: The proof can be completed by proving that any PPT adversary has negligible advantage in the security game. The detailed proof is presented in Appendix.

V. ATTRIBUTE-BASED ACCESS CONTROL WITH AUTHORIZED SEARCH

In this section, we define the system model and design goals, give an overview of EACAS, and describe EACAS in detail. Note that, since EACAS is primarily based on AKP-ABE proposed in Section IV, we denote the algorithms in AKP-ABE as `ABE` = {`ABE.Setup`, `ABE.Encrypt`, `ABE.KeyGen`, `ABE.Decrypt`}.

A. SYSTEM MODEL AND DESIGN GOALS

As shown in Figure. 1, the following three parties are included in our system.

- **Data owner (DO).** DO encrypts the data based on its attributes before uploading it to the cloud, and assigns access policies over the data attributes to the data users based on their system roles or credentials. DO is fully trusted in our system, and is in charge of the generation of keys.

- **Data user (DU).** DU is allowed to decrypt the ciphertext whose attributes satisfy his access policy. In addition, DU is able to define a search policy which is more restrictive than his access policy, and generate the corresponding trapdoor only based on his secret key. To obtain the ciphertext that satisfies the search policy, DU uploads the trapdoor to the cloud server to request the matching data. DU are not trusted, and they may collude to obtain data content outside the scope of their individual access privileges. They are also interested in the attribute information about the data.
- **Cloud server (CS).** CS is assumed with abundant storage and computing resources and is always online to render service. CS includes two parts: cloud storage server (CSS) and designated search server (DSS), where CSS helps DO store their data, and DSS performs data search on behalf of DU, and returns the matching data to DU. CS is semi-honest, which means it will follow the requests from DO and DU faithfully, but it is curious about the data information, including data content and attribute privacy. Note that, DSS is assigned with an extra private key to guarantee that others without the private key cannot deduce the attribute values in trapdoor through offline guessing attack.

The following goals should be fulfilled in EACAS.

- *Fine-grained access control.* The data stored at CSS is encrypted using its attributes, and can only be decrypted by DU whose access policy is satisfied by the ciphertext attributes. The access control should be embedded into the decryption process, but not performed by CS. Additionally, the expressive access policy with any threshold gate should be supported to guarantee the fine-graininess of access control.
- *Flexible and authorized search.* With the help of DSS, DU should be able to acquire the data ciphertext whose attributes satisfy the search policy. However, DU is only allowed to search the data within the scope of his access permission, which means he should only be able to generate trapdoor for a search policy which is more restrictive than his access policy. At the same time, the search policy should also be in an expressive form to support flexible search.
- *Attribute privacy preservation.* The generic attribute name is public in ciphertext and trapdoor, while the corresponding attribute value should be hidden to protect the data and user privacy. Any attacker cannot guess attribute values embedded in the ciphertext. In addition, the trapdoor should not leak attribute values in the search policy to any attackers without the private key for DSS.
- *Practical implementation.* The system operations should be completed with lower computation and storage overheads for practical applications.

B. OVERVIEW

AKP-ABE enables DO to define expressive access policy for DU and embed attribute values in the ciphertext implicitly, thus it can be applied to achieve fine-grained and privacy-preserving access control on the outsourced data. However, the hidden attribute information makes data search a challenge problem. To address this issue, we adopt the key delegation technique into AKP-ABE to enable DU to specify a search policy which is more restrictive than his access policy, and generate the corresponding trapdoor only based on his secret key. Figure. 1 gives an example of the access policy and search policy. Note that, the attribute values in the trapdoor are also concealed to protect the attribute information. In addition, a virtual attribute is introduced into both the ciphertext and trapdoor to prevent DSS accessing the data content. Concretely, the ciphertext is generated as two parts: (1) the original data encrypted under the original attribute set; (2) a trivial data "1" encrypted under the original attribute set added with the virtual attribute. While in the trapdoor, the virtual attribute is bound with the root node of the search tree through an AND gate, which makes it prerequisite for successful matching. When performing search on the ciphertext, DSS is able to retrieve the ciphertext whose attribute set satisfies the search policy by testing whether the trivial data "1" can be recovered, but cannot decrypt the ciphertext of the original data which is encrypted without the virtual attribute. As a result, EACAS is able to achieve fine-grained access control with authorized search on the data outsourced to cloud, simultaneously the data confidentiality and attribute privacy are protected effectively.

C. DETAILED EACAS

EACAS consists of six phases: system setup, key generation, data encryption, trapdoor generation, data search, and data decryption.

- **System Setup**

DO selects a security parameter ξ and calls the `Setup`(ξ) algorithm to generate PK and MSK. The `Setup` algorithm is the same with `ABE.Setup`, except that (1) a virtual attribute Att_v with the value v (different from the value of any real attribute) is included in the public key, and (2) an additional public and private key pair (pk_D, sk_D) for DSS is generated as $pk_D = g^\gamma$, and $sk_D = \gamma$, where γ is a random value in Z_p^*. Then, the system public key is published as

$$PK = \langle g, u, h, w, e(g, g)^\alpha, g_1, g_2, g_3, g_4, [Att_v : v], pk_D{}^1 \rangle.$$

The system master secret key is hold by DO as $MSK = \langle \alpha, \tau_1, \tau_2, \tau_3, \tau_4 \rangle$. Additionally, DO sends the private key $sk_D = \gamma$ to DSS.

- **Key Generation**

When DU joins the system, DO specifies an access policy $\mathcal{AP}$ according to his role, and distributes the secret key $SK = \langle \mathcal{AP}, \{D_x, D_{x,0}, D_{x,1}, D_{x,2}, D_{x,3}, D_{x,4}\}_{n_x \in atts(\mathcal{T})} \rangle$ generated

[1]For simplicity, the public key pk_D for DSS is included in the system public key PK.

by the KeyGen($PK, MSK, \mathcal{AP}$) algorithm to DU. Note that the KeyGen algorithm is the same with ABE.KeyGen. With the secret key SK, DU can decrypt the ciphertext whose attribute set satisfies $\mathcal{AP}$.

- **Data Encryption**

Before uploading the data M to CS, DO defines an attribute set S according to the data characteristics, and calls the Encrypt(PK, M, S) algorithm to produce the ciphertext CT. In the Encrypt algorithm, DO first picks two random values s and s', and computes $\widetilde{E} = M \cdot e(g,g)^{\alpha s}, \widetilde{E}' = e(g,g)^{\alpha s'}, E = g^s, E' = g^{s'}$. Then, for each attribute in S, it chooses random exponents $s_{x,1}, s_{x,2}, z_x$ from Z_p^*, and computes

$$E_{x,0} = w^{-s}(u^{s_x}h)^{z_x}, \quad E'_{x,0} = w^{-s'}(u^{s_x}h)^{z_x},$$
$$E_{x,1} = g_1^{z_x - s_{x,1}}, \quad E_{x,2} = g_2^{s_{x,1}},$$
$$E_{x,3} = g_3^{z_x - s_{x,2}}, E_{x,4} = g_4^{s_{x,2}}.$$

While for the virtual attribute Att_v, DO chooses a random value $r_v \in Z_p^*$, and computes $E_{v,0} = w^{-s'}(u^v h)^{r_v}, E_{v,1} = g^{r_v}$. Finally, the ciphertext to be uploaded to the cloud is produced as

$$CT = \langle N_S, \widetilde{E}, \widetilde{E}', E, E', E_{v,0}, E_{v,1}, \{E_{x,0}, E'_{x,0}, E_{x,1}, E_{x,2}, E_{x,3}, E_{x,4}\}_{x \in S}\rangle.$$

Remarks: The data encryption phase is based on the ABE.Encrypt algorithm, and the final ciphertext consists of two parts: CT_1 and CT_2, where $CT_1 = \langle N_S, \widetilde{E}, E, \{E_{x,0}, E_{x,1}, E_{x,2}, E_{x,3}, E_{x,4}\}_{x \in S}\rangle$ can be seen as the ciphertext generated from ABE.Encrypt(PK, M, S), which is the real data ciphertext, and $CT_2 = \langle N_S, \widetilde{E}', E', \{E'_{x,0}, E_{x,1}, E_{x,2}, E_{x,3}, E_{x,4}\}_{x \in S}, E_{v,0}, E_{v,1}\rangle$ can be seen as the ciphertext generated from ABE.Encrypt ($PK, 1, S \cup Att_v$), which is designed for data search. Note that there are two subtle points here. First, the ciphertext components $\{E_{x,1}, E_{x,2}, E_{x,3}, E_{x,4}\}_{x \in S}$ related to the attributes in S are shared between CT_1 and CT_2 to reduce computation and storage overheads. Second, since the virtual attribute is public, there is no need to hide its value, thus the corresponding ciphertext components $E_{v,0}$ and $E_{v,1}$ are computed as in the underlying KP-ABE [9], without using the linear splitting technique.

- **Trapdoor Generation**

DU first defines a search policy $\mathcal{SP}$ based on his access policy $\mathcal{AP}$, where $\mathcal{SP}$ has the similar expression form with $\mathcal{AP}$, and the search tree structure in $\mathcal{SP}$ should be more restrictive than the access tree structure in $\mathcal{AP}$, and the attribute value bound with the attribute name cannot be changed. Then, DU calls the *TrapGen*($PK, SK, \mathcal{SP}$) algorithm to generate the trapdoor TD associated with the search policy $\mathcal{SP}$ based on his secret key SK related with the access policy $\mathcal{AP}$. The key delegation technique is applied in the *TrapGen* algorithm, where a set of basic operations are executed step by step to convert the secret key SK for the access policy $\mathcal{AP}$ to the trapdoor for the search policy $\mathcal{SP}$. Specifically, following three steps are included in the *TrapGen* algorithm. The first step is to transfer the original secret key to a new trapdoor key by manipulating the existing gates, the second step is to prevent DSS decrypting the data ciphertext by adding an AND gate to the root node, and the last is to protect the attribute information in trapdoor against offline guessing attack from the attackers without the private key sk_D for DSS.

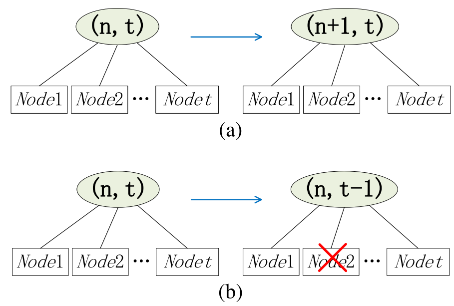


FIGURE 2. Operations of converting the access tree to a search tree. (a) Converting an (n, t)-gate to an (n + 1, t)-gate. (b) Converting an (n, t)-gate to an (n, t-1)-gate.

1) Manipulating an existing gate
This step can be executed multiple times on the gates of the access tree $\mathcal{T}$ in $\mathcal{AP}$ to convert it to a more restrictive search tree $\widetilde{\mathcal{T}}$, in which the two types of operations shown in Figure. 2 are included. Considering that the AND and OR gates can be seen as special cases of an (n, t)-gate, where n is the threshold and t is the number of children, we only give generic operations on the (n, t)-gate.
   a) Converting an (n, t)-gate to an $(n+1, t)$-gate
   Assume that node z is an (n, t)-gate associated with a polynomial p_z with the degree $(n-1)$. In order to change the threshold to $n+1$, DU defines a new polynomial $p'_z(X) = (X+1)p_z(X)$, such that the degree of p'_z is increased by 1 and $p'_z(0) = p_z(0)$. Then, for each child y of z, it computes $\delta_y = index(y) + 1$. Thus, the secret share for each leaf node x of the subtree $\mathcal{T}_y$ is changed to $p'_x(0) = \delta_y p_x(0)$, and the corresponding key components are computed as
   $$D'_x = (D_x)^{\delta_y}, \quad D'_{x,0} = (D_{x,0})^{\delta_y},$$
   $$D'_{x,1} = (D_{x,1})^{\delta_y}, \quad D'_{x,2} = (D_{x,2})^{\delta_y},$$
   $$D'_{x,3} = (D_{x,3})^{\delta_y}, \quad D'_{x,4} = (D_{x,4})^{\delta_y}.$$
   Note that, for those unaffected leaf nodes, δ_y can be seen as 1.
   b) Converting an (n, t)-gate to an $(n, t-1)$-gate
   This operation can be easily achieved by removing the key components associated with the leaves of the (n, t)-gate node from the original secret key.
2) Converting $\widetilde{\mathcal{T}}$ to $\{\widetilde{\mathcal{T}}$ AND $Att_v\}$
DU first adds a new node R representing a trivial (1,1)-gate above the root node r of $\widetilde{\mathcal{T}}$. Since the threshold

of R is 1, its corresponding polynomial is a constant, i.e., $p_R = p_r(0) = \alpha$, which has no effect on the secret shares related to the leaf nodes. Then, DU changes the (1,1)-gate into a (2,2) by adding the virtual attribute Att_v as a new child. It first picks a random value $k \in Z_p^*$, and changes the secret share related to the node r as $p_r'(0) = p_r(0)+k$, while the secret share for the node Att_v is $-k$, such that with the two secret shares associated with the node r and the node Att_v, the secret α related to node R can be recovered. Then, DU splits the secret k based on the search tree $\widetilde{\mathcal{T}}$, such that the secret share of k for the leaf node x is $q_x(0)$. Thus, the final secret share for the leaf node x in $\widetilde{\mathcal{T}}$ is changed to $\hat{p}_x(0) = p_x'(0) + q_x(0)$, and the corresponding key component is computed as

$$\hat{D}_x = D_x' g^{q_x(0)}.$$

While for the node Att_v, $p_{Att_v}(0) = -k$, and the related key components are computed as $D_v = g^{-k} w^{\lambda_v}, D_{v,0} = g^{\lambda_v}, D_{v,1} = (u^v h)^{-\lambda_v}$, where λ_v is a random number in Z_p^*. Note that, if the linear splitting technology is used to produce the ciphertext components for the virtual attribute in **Data Encryption** phase, DU cannot generate the corresponding key components without the master secret key here.

3) Binding with pk_D
DU selects a random value $\omega \in Z_p^*$, and computes $W = g^{\omega}$. For every leaf node x in $\widetilde{\mathcal{T}}$, DU computes $\hat{D}_{x,0} = (pk_d)^{\omega} D_{x,0}'$.

Then, the trapdoor is composed as

$$TD = \langle \widetilde{\mathcal{T}}, W, T_v, T_{v,0}, T_{v,1}, \{T_x, T_{x,0}, T_{x,1}, T_{x,2}, T_{x,3}, T_{x,4}\}_{n_x \in atts(\widetilde{\mathcal{T}})} \rangle,$$

where $T_x = \hat{D}_x$, $T_{x,0} = \hat{D}_{x,0}$, $T_{x,1} = D_{x,1}'$, $T_{x,2} = D_{x,2}'$, $T_{x,3} = D_{x,3}'$, $T_{x,4} = D_{x,4}'$, $T_v = D_v$, $T_{v,0} = D_{v,0}$ and $T_{v,1} = D_{v,1}$. To protect the attribute privacy in the trapdoor, only the search tree $\widetilde{\mathcal{T}}$ in the search policy $\mathcal{SP}$ is included in TD, while the attribute values corresponding to the leaf nodes are embedded into the trapdoor key components implicitly. Finally, DU sends the trapdoor TD to DSS.

- **Data Search**

To search the data ciphertext on behalf of DU, DSS calls the `Search`(PK, TD, CT, sk_D) algorithm for each ciphertext CT stored in CSS, and returns the reassembled ciphertext CT' to DU if CT satisfies the search policy associated with TD.

Specifically, in the `Search` algorithm, DSS first tests whether the attribute name set N_S in CT satisfies the search tree $\widetilde{\mathcal{T}}$ in TD. If not, it returns $\perp$. Otherwise, DSS computes $\widetilde{\mathcal{N}}$ from the search tree $\widetilde{\mathcal{T}}$, where $\widetilde{\mathcal{N}}$ means a set of minimum subsets of the attribute names in $atts(\widetilde{\mathcal{T}})$ that satisfy $\widetilde{\mathcal{T}}$.

Then, DSS tests whether there exists an attribute name set $\widetilde{N}_I \in \widetilde{\mathcal{N}}$ can pass the following test. During the test, it first computes $W' = (W)^{sk_D}$. Then, for each attribute name $n_x \in \widetilde{N}_I$, where it requires that $\widetilde{N}_I \subset N_S$, it computes

$$\begin{aligned} Q_x &= e(E_{x,1}, T_{x,1})e(E_{x,2}, T_{x,2})e(E_{x,3}, T_{x,3})e(E_{x,4}, T_{x,4}) \\ &= e(u^{t_x} h, g)^{-\delta_y z_x(\tau_1\tau_2\ t_{x,1}+\tau_3\tau_4\ t_{x,2})}, \\ P_x' &= e(E', T_x)e(E_{x,0}', T_{x,0}/W') \cdot Q_x \\ &= e(g, g)^{(\delta_y p_x(0)+q_x(0))s'} e(u, g)^{\delta_y z_x(s_x-t_x)(\tau_1\tau_2\ t_{x,1}+\tau_3\tau_4\ t_{x,2})} \\ &= e(g, g)^{\hat{p}_x(0)s'} e(u, g)^{\delta_y z_x(s_x-t_x)(\tau_1\tau_2\ t_{x,1}+\tau_3\tau_4\ t_{x,2})}. \end{aligned}$$

If the attribute values in the attribute set and trapdoor are matching for the same attribute name n_x (e.g. $s_x = t_x$), then $P_x' = e(g, g)^{\hat{p}_x(0)s'}$. Furthermore, if the attribute values are matching for all the attribute names in $\widetilde{N}_I$, $P_r' = e(g, g)^{p_r'(0)s'} = e(g, g)^{(\alpha+k)s'}$ can be calculated through combining $\{P_x'\}_{n_x \in \widetilde{N}_I}$ recursively with the Lagrange polynomial interpolation technique. In addition, for the virtual attribute, DSS computes $P_v = e(E', D_v)e(E_{v,0}, T_{v,0})e(E_{v,1}, T_{v,1}) = e(g, g)^{-ks'}$. Thus, $\widetilde{E}' = P_r' P_v$, which means that the ciphertext CT matches with the trapdoor. Finally, if no such $\widetilde{N}_I \in \widetilde{\mathcal{N}}$ exists, the `Search` algorithm outputs $\perp$.

For the ciphertext CT that matches with the trapdoor, DSS composes $CT' = \langle \widetilde{N}_I, \widetilde{E}, E, \{E_{x,0}, Q_x\}_{n_x \in \widetilde{N}_I} \rangle$, and returns it to DU.

- **Data Decryption**

After receiving the ciphertext from DSS, DU calls the `Decrypt`(CT', SK) algorithm to recover the data content. In the *Decrypt* algorithm, for each attribute name $n_x \in \widetilde{N}_I$, it computes

$$P_x = e(E, D_x)e(E_{x,0}, D_{x,0}) \cdot (Q_x)^{1/\delta_y} = e(g, g)^{p_x(0)s}$$

As in the `ABE.Decrypt` algorithm, the term $e(g, g)^{\alpha s}$ can be recovered, and M can be calculated through $\widetilde{E}/e(g, g)^{\alpha s}$.

Remarks: In this phase, we take advantage of the pairing result Q_x for each attribute computed in the phase of **Data Search**, such that a lot of computation cost is saved. In addition, since the leaf nodes in $\widetilde{N}_I$ satisfy the search tree $\widetilde{\mathcal{T}}$, they also satisfy the access tree $\mathcal{T}$, thus $e(g, g)^{\alpha s}$ can be recovered successfully. Note that, δ_y is the exponent used in the first step of the **Trapdoor Generation** phase.

VI. DISCUSSIONS

A. PROPERTY DISCUSSION

- *Fine-grained access control.* In EACAS, the data is encrypted under an attribute set, and DU is assigned with an access policy on those data attributes. DU can only decrypt the ciphertext whose attributes satisfy his access policy. Additionally, the expression of access policy supports any threshold gate, thus fine-graininess of the access control can be guaranteed.
- *Flexible and authorized search.* As shown in the phase of **Data Encryption**, in addition to the data ciphertext CT_1, the ciphertext CT_2 generated by encrypting the trivial data "1" under the attribute set $S \cup Att_v$ is also contained in the final ciphertext stored in CSS. The trapdoor generated by DU can be seen as the decryption key related to the tree structure $\widetilde{\mathcal{T}}$ AND Att_v. If the

data "1" can be recovered with the decryption key in the trapdoor, which means the attribute set $S \cup Att_v$ satisfies the policy of $\widetilde{\mathcal{T}}$ AND Att_v, then the attribute set S of the corresponding data ciphertext satisfies the policy associated with $\widetilde{\mathcal{T}}$. Thus, the ciphertext search is achieved through testing whether the trivial data "1" can be recovered. Additionally, with the key delegation technology, DU is only able to generate trapdoor for a search policy which is more restrictive than his access policy, thus only authorized search is allowed.

B. SECURITY DISCUSSION

- *Data confidentiality.* The user secret key SK and the data ciphertext CT_1 in EACAS have the same structures with those in AKP-ABE, such that DU cannot decrypt any ciphertext out of the scope of his access privilege, even by colluding. In addition, with the trapdoor submitted by DU, DSS may be able to recover the trivial data "1" from CT_2, but it cannot decrypt the data ciphertext CT_1 to get the data M. Because the trapdoor given to DSS can be seen as the decryption key bound with the additional virtual attribute through an AND gate, thus only the ciphertext possessing the virtual attribute can be decrypted successfully.
- *Attribute privacy.* We protect the attribute privacy from two levels. On the first level, for both the ciphertext and trapdoor, the attribute values are embedded into the corresponding components implicitly, and only the generic attribute name is public. On the second level, the linear splitting technique introduced in AKP-ABE can protect attribute privacy against the offline guessing attack on the ciphertext. While for the issue of offline guessing attack on trapdoor, we embed the public key pk_D of the designated search server (DSS) into the trapdoor TD, such that anyone without the private key sk_D for DSS cannot eliminate the random term $g^{\omega\gamma}$ in $T_{x,0}$, thus no attribute values can be derived from the trapdoor.

VII. PERFORMANCE EVALUATION

In this section, we evaluate the performance of EACAS on its storage and computation overheads, and conduct extensive experiments to show its practicality.

A. NUMERICAL ANALYSIS

Both AKP-ABE and EKS proposed in [17] are designed based on the large universe KP-ABE [9], and they only support either fine-grained access control or flexible keyword search, as shown in Table 1. ABE-EAKS in [13] realizes the same properties with EACAS, but it triggers large overheads.

Table 2 and 3 demonstrate the numerical analysis results of storage overhead and computation cost, respectively.[2] The storage and computation overheads in AKP-ABE is larger than that in the underlying KP-ABE [9]. This is due to the application of the linear splitting technique to protect attribute privacy. The same situation also occurs in the EKS [17], where the `TrapGen` and `Search` algorithms have the similar design as the `KeyGen` and `Decrypt` in AKP-ABE. Compared with both AKP-ABE and EKS, EACAS only has a small additional storage and computation overheads brought by the virtual attribute, but simultaneously achieves fine-grained access control and expressive ciphertext search. The cost of EACAS is far less than the cumulative cost of the double encryption in [13]. Additionally, the `Decrypt` algorithm in EACAS only requires a small number of paring and exponentiation operations in Table 3, since most of the pairing results calculated during the searching process are reused.

TABLE 1. Comparisons of functionalities.

Schemes	Fine-grained Access Control	Flexible Keyword Search	Attribute or Keyword Privacy	Underlying Scheme
KP-ABE [9]	✓	×	×	⊥
EKS [17]	×	✓	✓	KP-ABE [9]
AKP-ABE	✓	×	✓	KP-ABE [9]
ABE-EAKS [13]	✓	✓	✓	Anonymous ABE and EKS
EACAS	✓	✓	✓	AKP-ABE

Notions in Table 2 and 3 are clarified as follows.

- E, P: The operations of exponentiation and pairing, respectively.
- $|Z_p|, |\mathbb{G}|, |\mathbb{G}_T|$: The bit-length of the element in Z_p, $\mathbb{G}$ and $\mathbb{G}_T$, respectively.
- l_C, l_A, l_T: The number of attributes in the ciphertext, access policy and trapdoor, respectively.
- k: The number of attributes affected during the `TrapGen` algorithm in EACAS.
- l_I: The number of attributes for the final successful decryption.
- $\chi_{A,1}$: The number of elements in $\mathcal{N} = \{I_1, \ldots, I_{\chi_{A,1}}\}$, where $\mathcal{N}$ means the set of minimum subsets satisfying the access tree.
- $\chi_{A,2}$: The total number of attributes in all the subsets of $\mathcal{N}$, i.e., $|I_1| + \cdots + |I_{\chi_{A,1}}|$.
- $\chi_{T,1}$: The number of elements in $\widetilde{\mathcal{N}} = \{\hat{I}_1, \ldots, \hat{I}_{\chi_{T,1}}\}$, where $\widetilde{\mathcal{N}}$ means the set of minimum subsets satisfying the search tree.
- $\chi_{T,2}$: The total number of attributes in all the subsets of $\widetilde{\mathcal{N}}$, i.e., $|\hat{I}_1| + \cdots + |\hat{I}_{\chi_{T,1}}|$.
- $\perp$: It is not applicable for this scheme.

B. SIMULATION RESULTS

We implement EACAS with python 3.6 on a notebook equipped with Core i7 2.80GHz CPU and 16GB RAM. The operation system is Ubuntu 18.04. The Charm framework (v0.5) is used to perform the cryptographic operations with the supporting of the PBC library (v0.5.14) and the OpenSSL library (v1.0.2). Since Charm only supports operations in asymmetric groups, i.e., $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, we transform our constructions into asymmetric groups by choosing another element g' in $\mathbb{G}_2$ and computing $g_1, g_2, g_3, g_4, D_{x,0}$ with g'. We conduct the experiments over MNT224,

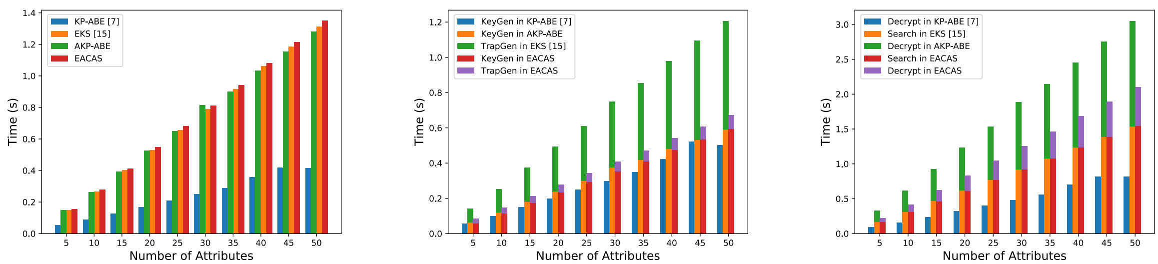
[2] Since the instantiation of the ABE-EAKS is not given in [13], we just take the cumulative analysis results of AKP-ABE and EKS as a contrast.

TABLE 2. Comparisons of storage overhead.

Schemes	PK	MSK	SK	TD	CT
KP-ABE [9]	$4\lvert\mathbb{G}\rvert + \lvert\mathbb{G}_T\rvert$	$\lvert Z_p\rvert$	$3l_A\lvert\mathbb{G}\rvert$	$\perp$	$(2l_C+1)\lvert\mathbb{G}\rvert + \lvert\mathbb{G}_T\rvert$
EKS [17]	$9\lvert\mathbb{G}\rvert + \lvert\mathbb{G}_T\rvert$	$5\lvert Z_p\rvert$	$\perp$	$(6l_T+2)\lvert\mathbb{G}\rvert$	$(5l_C+1)\lvert\mathbb{G}\rvert + \lvert\mathbb{G}_T\rvert$
AKP-ABE	$8\lvert\mathbb{G}\rvert + \lvert\mathbb{G}_T\rvert$	$5\lvert Z_p\rvert$	$6l_A\lvert\mathbb{G}\rvert$	$\perp$	$(5l_C+1)\lvert\mathbb{G}\rvert + \lvert\mathbb{G}_T\rvert$
EACAS	$9\lvert\mathbb{G}\rvert + \lvert\mathbb{G}_T\rvert$	$5\lvert Z_p\rvert$	$6l_A\lvert\mathbb{G}\rvert$	$(6l_T+4)\lvert\mathbb{G}\rvert$	$(6l_C+4)\lvert\mathbb{G}\rvert + 2\lvert\mathbb{G}_T\rvert$

TABLE 3. Comparisons of computation cost.

Schemes	Encrypt	KeyGen	TrapGen	Search	Decrypt
KP-ABE [9]	$(3l_C+3)E$	$5l_AE$	$\perp$	$\perp$	$3l_IP + l_IE$
EKS [17]	$(6l_C+3)E$	$\perp$	$(8l_T+3)E$	$(\chi_{T,2}+2)E + (6\chi_{T,2}+2)P$	$\perp$
AKP-ABE	$(6l_C+3)E$	$8l_AE$	$\perp$	$\perp$	$\leq 6\chi_{A,2}P + \chi_{A,2}E$
EACAS	$(6l_C+6)E$	$8l_AE$	$(6k+2l_T+1)E$	$(\chi_{T,2}+1)E + (6\chi_{T,2}+3)P$	$2l_IP + 2l_IE$



(a) (b) (c)

FIGURE 3. **Comparisons of the running time for the `Encrypt`, `KeyGen`, `TrapGen`, `Search` and `Decrypt` algorithms. (a) Encrypt. (b) KeyGen and TrapGen. (c) Search and Decrypt.**

TABLE 4. Algorithm execution time in EACAS.

Algorithms	Encrypt	KeyGen	TrapGen	Search	Decrypt
Average Time (*ms*)	142	78	31	150	38

a 224-bit asymmetric elliptic curve providing 112-bit security level. The experiments are run 20 times and the average value of running time is shown.

We collect the running time of the `Encrypt`, `KeyGen`, `TrapGen`, `Search` and `Decrypt` algorithms with a simple access policy in AND gates, and the number of attributes are from 5 to 50. To enable a visible comparison, we illustrate the results of `KeyGen` and `TrapGen`, `Search` and `Decrypt` in Figure. 3(b) and Figure. 3(c), respectively. The running time of the algorithms increases linearly with the number of attributes. Figure. 3(a) demonstrates that the encryption time in EACAS is almost the same as that in AKP-ABE and EKS. Figure. 3(b) shows that the total execution time of `KeyGen` and `TrapGen` algorithms in EACAS is less than the cumulative time of `TrapGen` in EKS and `KeyGen` in AKP-ABE. Figure. 3(c) exhibits that the total running time of `Search` and `Decrypt` algorithms in EACAS is also less than the cumulative time of `Search` in EKS and `Decrypt` in AKP-ABE, because more bilinear pairing should be computed in the combination of AKP-ABE and EKS.

Table 4 shows the algorithm execution time in EACAS under the specified polices shown in Figure. 1. All the operations can be completed less than 150*ms*, especially the `TrapGen` and `Decrypt` operations can be performed by DU within 31*ms* and 38*ms*. The `Search` algorithm is executed by DSS, which is powerful on computation, so that the time cost can be much lower than 150*ms*.

VIII. CONCLUSION

In this paper, we have proposed an efficient attribute-based access control with authorized search scheme (EACAS), which can meet the requirements for data sharing in cloud storage and protect the data confidentiality and attribute privacy effectively. In EACAS, the data users are able to specify the search policy based on their access policies granted by the data owner, and generate the corresponding trapdoor without the help of the data owner. Meanwhile, the cloud server is allowed to search the ciphertext on behalf of data users without knowing the attribute information and underlying plaintext. We have discussed the property, security and performance of EACAS, and implemented it to demonstrate that EACAS is efficient and effective for practical applications. In the future work, we will introduce anonymous KP-ABE with flexible data sharing and efficient data storage for e-health cloud.

APPENDIX

SECURITY PROOF OF THEOREM 1

Theorem 1: If the q-2 DBDH assumption [9] and the D-Linear assumption [19] hold, AKP-ABE with partially hid-

den attributes is selectively secure under chosen plaintext attack.

Proof: The proof utilizes a series of security games to argue that no adversary has non-negligible advantage to win the original game denoted by *Game*. Since the attribute name set N_S included in the ciphertext is the same for the two different challenge attribute sets, we remove it from the ciphertext for simplicity, and we define the challenge ciphertext for the adversary in *Game* as $CT = \langle \widetilde{E}^*, E^*, \{E^*_{\tau,0}, E^*_{\tau,1}, E^*_{\tau,2}, E^*_{\tau,3}, E^*_{\tau,4}\}_{\tau\in[1,m]}\rangle$, where m represents the number of attributes in S.

We first modify the original game *Game* into $Game_0$ by replacing the component $\widetilde{E}^*$ with a random element $Z \in \mathbb{G}_T$. Then, from $Game_0$ to $Game_m$, we change $E^*_{\tau,1}$ to X_τ one by one for $\tau \in [1, m]$, and from $Game_m$ to $Game_{2m}$ change $E^*_{\tau,3}$ to Y_τ one by one for $\tau \in [1, m]$, where $\{X_\tau\}_{\tau\in[1,m]}$, $\{Y_\tau\}_{\tau\in[1,m]}$ are random elements in $\mathbb{G}$.

- *Game*:
  $$CT = \langle \widetilde{E}^*, E^*, \{E^*_{\tau,0}, E^*_{\tau,1}, E^*_{\tau,2}, E^*_{\tau,3}, E^*_{\tau,4}\}_{\tau\in[1,m]}\rangle.$$
- $Game_0$:
  $$CT_0 = \langle Z, E^*, \{E^*_{\tau,0}, E^*_{\tau,1}, E^*_{\tau,2}, E^*_{\tau,3}, E^*_{\tau,4}\}_{\tau\in[1,m]}\rangle.$$
- $Game_1$:
  $$CT_1 = \langle Z, E^*, (E^*_{1,0}, X_1, E^*_{1,2}, E^*_{1,3}, E^*_{1,4}), \{E^*_{\tau,0}, E^*_{\tau,1}, E^*_{\tau,2}, E^*_{\tau,3}, E^*_{\tau,4}\}_{\tau\in[2,m]}\rangle.$$
- ……
- $Game_m$:
  $$CT_m = \langle Z, E^*, \{E^*_{\tau,0}, X_\tau, E^*_{\tau,2}, E^*_{\tau,3}, E^*_{\tau,4}\}_{\tau\in[1,m]}\rangle.$$
- $Game_{m+1}$:
  $$CT_{m+1} = \langle Z, E^*, (E^*_{1,0}, X_1, E^*_{1,2}, Y_1, E^*_{1,4}), \{E^*_{\tau,0}, X_\tau, E^*_{\tau,2}, E^*_{\tau,3}, E^*_{\tau,4}\}_{\tau\in[2,m]}\rangle.$$
- ……
- $Game_{2m}$:
  $$CT_{2m} = \langle Z, E^*, \{E^*_{\tau,0}, X_\tau, E^*_{\tau,2}, Y_\tau, E^*_{\tau,4}\}_{\tau\in[1,m]}\rangle.$$

Note that in the last game $Game_{2m}$, the component $\widetilde{E}^*$ bound with the message M is replaced with the random element Z, thus the adversary has a negligible advantage to distinguish between M_0 and M_1 from $Game_{2m}$. In addition, the components $E^*_{\tau,1}$ and $E^*_{\tau,3}$ bound with each attribute have also be replaced with random values, which prevent the adversary deducing any valuable information about z_τ, thus the adversary has a negligible advantage to distinguish between S_0 and S_1 according to the component $E^*_{\tau,0}$. Based on the above analysis, the adversary has a negligible advantage to win the game $Game_{2m}$. Furthermore, if the advantage for an adversary to distinguish the sequence of the games is negligible, then the advantage for it to win the original game *Game* is negligible.

Lemma 1: If the q-2 DBDH assumption holds, all PPT adversaries have a negligible advantage to distinguish between *Game* and $Game_0$.

Lemma 2: If the D-Linear assumption holds, all PPT adversaries have a negligible advantage to distinguish between $Game_j$ and $Game_{j+1}$ for $j \in [0, 2m - 1]$.

According to the lemmas above, it can be concluded that no PPT adversary can distinguish the sequence of the games with non-negligible advantage, thus no PPT adversary can win the original game *Game* with non-negligible advantage. Specifically, **Lemma** 1 guarantees the data confidentiality, and **Lemma** 2 guarantees the privacy of attribute values. Please refer to [17] for more details of the proof of **Lemma** 1 and **Lemma** 2.

REFERENCES

[1] *Cloud Storage*. Accessed: Jan. 1, 2019. [Online]. Available: https://en.wikipedia.org/wiki/Cloud-storage
[2] *Cloud Storage Service*. Accessed: Feb. 1, 2019. [Online]. Available: https://searchstorage.techtarget.com/definition/cloud-storage-service
[3] *The Best Cloud Storage Services*. Accessed: Feb. 1, 2019. [Online]. Available: https://www.digitaltrends.com/computing/best-cloud-storage-services-compared/
[4] G. Wang, R. Lu, and Y. L. Guan, "Enabling efficient and privacy-preserving health query over outsourced cloud," *IEEE Access*, vol. 6, pp. 70831–70842, 2018.
[5] J. Ni, X. Lin, and X. S. Shen, "Efficient and secure service-oriented authentication supporting network slicing for 5G-enabled IoT," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 3, pp. 644–657, Mar. 2018.
[6] W. Guo, J. Shao, R. Lu, Y. Liu, and A. A. Ghorbani, "A privacy-preserving online medical prediagnosis scheme for cloud environment," *IEEE Access*, vol. 6, pp. 48946–48957, 2018.
[7] C. Huang, R. Lu, H. Zhu, J. Shao, and X. Lin, "FSSR: Fine-grained EHRs sharing via similarity-based recommendation in cloud-assisted ehealthcare system," in *Proc. 11th ACM Asia Conf. Comput. Commun. Secur.*, 2016, pp. 95–106.
[8] D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Side channels in cloud services: Deduplication in cloud storage," *IEEE Security Privacy*, vol. 8, no. 6, pp. 40–47, Nov./Dec. 2010.
[9] Y. Rouselakis and B. Waters, "Practical constructions and new proof methods for large universe attribute-based encryption," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Berlin, Germany, 2013, pp. 463–474.
[10] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 89–98.
[11] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. IEEE INFOCOM*, San Diego, CA, USA, Mar. 2010, pp. 1–9.
[12] J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han, "Flexible and fine-grained attribute-based data storage in cloud computing," *IEEE Trans. Serv. Comput.*, vol. 10, no. 5, pp. 785–796, Sep./Oct. 2017.
[13] H. Cui, R. H. Deng, J. K. Liu, and Y. Li, "Attribute-based encryption with expressive and authorized keyword search," in *Proc. Conf. Inf. Secur. Privacy*. Auckland, New Zealand: Springer, 2017, pp. 106–126.
[14] Y. Chen, J. Zhang, D. Lin, and Z. Zhang, "Generic constructions of integrated PKE and PEKS," *Des., Codes Cryptogr.*, vol. 78, no. 2, pp. 493–526, 2016.
[15] J. Shi, J. Lai, Y. Li, R. H. Deng, and J. Weng, "Authorized keyword search on encrypted data," in *Proc. Eur. Symp. Res. Comput. Secur.* Wroclaw, Poland: Springer, 2014, pp. 419–435.
[16] P. Jiang, Y. Mu, F. Guo, and Q. Wen, "Public key encryption with authorized keyword search," in *Proc. Conf. Inf. Secur. Privacy*. Melbourne, VIC, Australia: Springer, 2016, pp. 170–186.
[17] H. Cui, Z. Wan, R. H. Deng, G. Wang, and Y. Li, "Efficient and expressive keyword search over encrypted data in cloud," *IEEE Trans. Depend. Sec. Comput.*, vol. 15, no. 3, pp. 409–422, May/Jun. 2016.
[18] X. Boyen and B. Waters, "Anonymous hierarchical identity-based encryption (without random oracles)," in *Proc. CRYPTO*. San Diego, CA, USA: Springer, 2006, pp. 290–307.
[19] T. Nishide, K. Yoneyama, and K. Ohta, "Attribute-based encryption with partially hidden encryptor-specified access structures," in *Proc. ACNS*. New York, NY, USA: Springer, 2008, pp. 111–129.
[20] J. A. Akinyele *et al.*, "Charm: A framework for rapidly prototyping cryptosystems," *J. Cryptograph. Eng.*, vol. 3, no. 2, pp. 111–128, 2013.

[21] F. Guo, Y. Mu, W. Susilo, D. S. Wong, and V. Varadharajan, "CP-ABE with constant-size keys for lightweight devices," *IEEE Trans. Inf. Forensics Security*, vol. 9, no. 5, pp. 763–771, May 2014.

[22] L.-Y. Yeh, P.-Y. Chiang, Y.-L. Tsai, and J.-L. Huang, "Cloud-based fine-grained health information access control framework for lightweightiot devices with dynamic auditing andattribute revocation," *IEEE Trans. Cloud Comput.*, vol. 6, no. 2, pp. 532–544, Apr./Jun. 2018.

[23] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proc. 5th ACM Symp. Inf., Comput. Commun. Secur.* Beijing, China: ACM, 2010, pp. 261–270.

[24] Z. Yan, X. Li, M. Wang, and A. V. Vasilakos, "Flexible data access control based on trust and reputation in cloud computing," *IEEE Trans. Cloud Comput.*, vol. 5, no. 3, pp. 485–498, Jul./Sep. 2017.

[25] K. Yang, K. Zhang, X. Jia, M. A. Hasan, and X. Shen, "Privacy-preserving attribute-keyword based data publish-subscribe service on cloud platforms," *Inf. Sci.*, vol. 387, pp. 116–131, May 2017.

[26] J. Hao, C. Huang, J. Ni, H. Rong, M. Xian, and X. Shen, "Fine-grained data access control with attribute-hiding policy for cloud-based IoT," *Comput. Netw.*, vol. 153, pp. 1–10, Apr. 2019.

[27] J. Lai, R. H. Deng, and Y. Li, "Fully secure cipertext-policy hiding CP-ABE," in *Proc. ISPEC*. Guangzhou, China: Springer, 2011, pp. 24–39.

[28] J. Li, K. Ren, B. Zhu, and Z. Wan, "Privacy-aware attribute-based encryption with user accountability," in *Proc. ISC*. Pisa, Italy: Springer, 2009, pp. 347–362.

[29] J. Lai, R. H. Deng, and Y. Li, "Expressive CP-ABE with partially hidden access structures," in *Proc. 7th ACM Symp. Inf., Comput. Commun. Secur.*, Seoul, South Korea, 2012, pp. 18–19.

[30] H. Cui, R. H. Deng, G. Wu, and J. Lai, "An efficient and expressive ciphertext-policy attribute-based encryption scheme with partially hidden access structures," in *Proc. PROVSEC*. Nanjing, China: Springer, 2016, pp. 19–38.

[31] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. EUROCRYPT*. Interlaken, Switzerland: Springer, 2004, pp. 506–522.

[32] Y. Yu, J. Ni, H. Yang, Y. Mu, and W. Susilo, "Efficient public key encryption with revocable keyword search," *Secur. Commun. Netw.*, vol. 7, no. 2, pp. 466–472, 2014.

[33] D. J. Park, K. Kim, and P. J. Lee, "Public key encryption with conjunctive field keyword search," in *Proc. WISA*. Jeju Island, South Korea: Springer, 2004, pp. 73–86.

[34] D. Boneh and B. Waters, "Conjunctive, subset, and range queries on encrypted data," in *Proc. TCC*. Amsterdam, The Netherlands: Springer, 2007, pp. 535–554.

[35] J. Lai, X. Zhou, R. H. Deng, Y. Li, and K. Chen, "Expressive search on encrypted data," in *Proc. 8th ACM SIGSAC Symp. Inf., Comput. Commun. Secur.*, Hangzhou, China, 2013, pp. 243–252.

[36] Z. Lv, C. Hong, M. Zhang, and D. Feng, "Expressive and secure searchable encryption in the public key setting," in *Proc. ISC*. Hong Kong: Springer, 2014, pp. 364–376.

[37] 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 *Proc. SDM*. Seoul, South Korea: Springer, 2006, pp. 75–83.

[38] E. Shen, E. Shi, and B. Waters, "Predicate privacy in encryption systems," in *Proc. TCC*. San Francisco, CA, USA: Springer, 2009, pp. 457–473.

[39] H. S. Rhee, J. H. Park, W. Susilo, and D. H. Lee, "Trapdoor security in a searchable public-key encryption scheme with a designated tester," *J. Syst. Softw.*, vol. 83, no. 5, pp. 763–771, 2010.

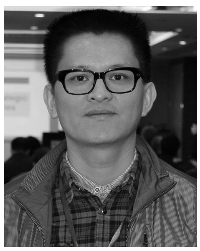
[40] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, "Protecting your right: Attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud," in *Proc. IEEE Conf. Comput. Commun.*, Toronto, ON, Canada, Apr./May 2014, pp. 226–234.

JIALU HAO received the B.E. degree from Shandong University, Jinan, Shandong, in 2013, and the M.S. degree from the National University of Defense Technology, Changsha, Hunan, in 2015, where he is currently pursuing the Ph.D. degree. In 2018, he joined the Department of Electrical and Computer Engineering, University of Waterloo, as a Visiting Student. His research interests include wireless network security, cloud security, and applied cryptography.

JIAN LIU received the B.S., M.S., and Ph.D. degrees from the National University of Defense Technology, in 2009, 2011, and 2016, respectively, where he is currently a Lecturer with the School of Electronic Science and Technology. His research interests include secure data outsourcing in cloud computing, public auditing, and access control mechanisms.

HUIMEI WANG received the B.S. degree from Southwest Jiaotong University, in 2004, and the M.S. and Ph.D. degrees from the National University of Defense Technology, in 2007 and 2012, respectively, where she is currently a Lecturer with the College of Electronic Science and Technology. Her research interests include evaluation techniques in network security, cloud computing, and distributed systems.

LINGSHUANG LIU received the B.E. and M.S. degrees from the School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu, China, in 2014 and 2017, respectively. She is currently an Assistant Engineer with the Science and Technology on Reactor System Design Technology Laboratory, Nuclear Power Institute of China, Chengdu. Her research interests include mobile security, smartphone authentication, and cloud storage security.



MING XIAN received the B.E., M.S., and Ph.D. degrees from the National University of Defense Technology, in 1991, 1995, and 1998, respectively, where he is currently a Professor with the College of Electronic Science and Technology. His research interests include cryptography and information security, cloud computing, and system modeling, and simulation and evaluation.

XUEMIN SHEN (M'97–SM'02–F'09) is currently a University Professor with the Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, ON, Canada. His research interests include resource management in interconnected wireless/wired networks, wireless network security, social networks, smart grid, and vehicular ad hoc and sensor networks. He is a registered Professional Engineer of Ontario, Canada, an Engineering Institute of Canada Fellow, a Canadian Academy of Engineering Fellow, a Royal Society of Canada Fellow, and a Distinguished Lecturer of the IEEE Vehicular Technology Society and Communications Society.

Dr. Shen received the Joseph LoCicero Award, in 2015, the Education Award, in 2017, from the IEEE Communications Society, and the James Evans Avant Garde Award, in 2018, from the IEEE Vehicular Technology Society. He is the Editor-in-Chief of the IEEE Internet of Things Journal and the Vice President on Publications of the IEEE Communications Society.