Meng *et al. J Wireless Com Network* (2021) 2021:20
https://doi.org/10.1186/s13638-020-01875-2

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Ciphertext-policy attribute-based encryption with hidden sensitive policy from keyword search techniques in smart city

Fei Meng[1,2], Leixiao Cheng[1] and Mingqiang Wang[1,2*]

*Correspondence: wangmingqiang@sdu.edu.cn
[1] School of Mathematics, Shandong University, South Shanda Road, No. 27, Jinan 250100, China
Full list of author information is available at the end of the article

Abstract

Countless data generated in Smart city may contain private and sensitive information and should be protected from unauthorized users. The data can be encrypted by Attribute-based encryption (CP-ABE), which allows encrypter to specify access policies in the ciphertext. But, traditional CP-ABE schemes are limited because of two shortages: the access policy is public i.e., privacy exposed; the decryption time is linear with the complexity of policy, i.e., huge computational overheads. In this work, we introduce a novel method to protect the privacy of CP-ABE scheme by keyword search (KS) techniques. In detail, we define a new security model called *chosen sensitive policy security*: two access policies embedded in the ciphertext, one is public and the other is sensitive and hidden. If user's attributes don't satisfy the public policy, he/she cannot get any information (attribute name and its values) of the hidden one. Previous CP-ABE schemes with hidden policy only work on the "AND-gate" access structure or their ciphertext size or decryption time maybe super-polynomial. Our scheme is more expressive and compact. Since, IoT devices spread all over the smart city, so the computational overhead of encryption and decryption can be shifted to third parties. Therefore, our scheme is more applicable to resource-constrained users. We prove our scheme to be selective secure under the decisional bilinear Diffie-Hellman (DBDH) assumption.

Keywords: Smart city, Attribute-based encryption, Hidden sensitive policy, Keyword search, Fog nodes

1 Introduction

Smart city is a new concept brought up with the technological revolution providing various digital services for citizens to make their life more convenient among all aspects of daily life including education, health care, traffic transport, job recruitment and so on. From the perspective of technological development, the construction of smart cities requires the realization of comprehensive perception, ubiquitous interconnection, pervasive computing and integrated applications through the Internet of Things, cloud computing and other new-generation information technology applications represented by mobile technology. From the perspective of social development, smart cities also require the application of tools and methods such as wikis, social

networks, Fab Lab, Living Lab and integrated integration methods to facilitate residents' lives and economic activities. To build a smart city, enormous data will be generated, processed and analysed.

Cloud server is an internet-based paradigm that provides massive data storage and processing services for innumerable enterprises and individuals. Fog nodes [1] are physical components (such as gateways, switches, routers, servers, etc.) or virtual components (such as ED switches, virtual machines, Cloudlet 9, etc.) that are tightly connected with the network and provide computing resources. As shown in Fig. 1, such devices can be found everywhere in the smart city providing more efficient and convenient services.

Generally speaking, data and services on the cloud are open and accessible to anyone, and data owner will lose any control on the data as soon as it uploaded to the cloud. In many distributed applications, it is necessary to enforce a specific access control policy on sensitive data, and only authorized users can access these data.

In this case, Sahai et al. [2] first introduced the concept of attribute-based encryption (ABE) achieving both scalable and fine-grained access control on ciphertext. ABE schemes are generally divided into two types: ciphertext-policy ABE (CP-ABE) [3] and key-policy ABE (KP-ABE) [4]. In CP-ABE, the access policy is embedded in the ciphertext and anyone can decrypt it as long as his/her attributes satisfy the policy.

One problem in traditional CP-ABE scheme [4, 5] is that the access policy is sent along with a ciphertext to inform end users which attributes satisfy the access policy, therefore the privacy in the policy could be exposed. However, this property is not suitable for many application scenarios, such as medical, industrial and financial fields. For instance, the access policy of an encrypted files of patient's medical record may reveal individual privacy; a company may hire eligible staff with specific qualification (i.e., attributes) which may expose the company's future development strategy.

If it is not known which attributes should be used for decryption, the decryption will be infeasible for authorized users. So, Nishide et al. [6] introduced the notion of partially hidden access policy in CP-ABE. In their scheme, the attribute is defined by two parts, attribute name and attribute values and only attribute value is concealed in their shceme. Although this method protects the privacy of the policy to some extent, it also has some drawbacks: in some cases, the attribute name still contains sensitive and valuable information, and it's still revealed in the access policy; If the end user has multiple

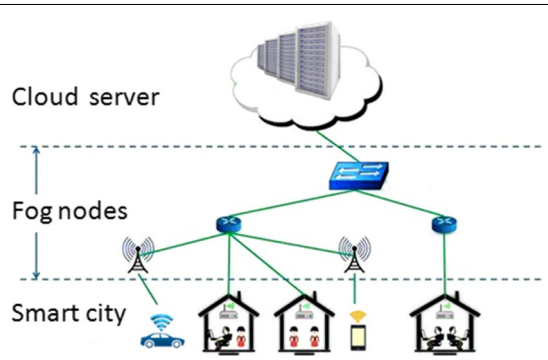

Fig. 1 Fog nodes in smart city

values for each attribute, the decryption time maybe super-polynomial, since he/she has to guess which attribute value is exactly embedded in the ciphertext. Inner-product predicate encryption (IPE) [7] is another method to protect policy privacy, but it could cause the ciphertext size to be super-polynomial.

Besides attribute values, sometimes, attribute names are also sensitive and should be kept secret. For example: a recruitment sharing in the cloud can only be accessed by specific applicants.The access policy may be defined as {[Gender: male or female] AND [Education: M.D. or PH.D.]} AND {[Probability and statistics: statistics or econometrics] AND [Computer science and technology: data mining or machine learning]}. In this case, the "Probability and statistics: statistics or econometrics" and "Computer science and technology: data mining or machine learning" in the access policy are obviously more sensitive, since it may reveal commercial confidentiality that the company is managing to achieve transformation with the help of IoT.

Another problem in the existing ABE schemes [3–5] is that the number of pairing and exponentiation operations for ciphertext decryption is linear with the complexity of access policy, which means the computation cost of end user is quite expensive. This property is not suitable for users on their resource-constrained mobile devices. To reduce the computational overhead of end user, some cryptographic operations with heavy computational load can be outsourced to third-party service [8, 9]. In smart cities, countless IoT devices connected to the Internet in every corner of the city can be utilized to provide much more convenient and faster computing resources for resource-limited end users.

1.1 Motivation

In the above-mentioned example, data owner could encrypt the data in a different way such that any one obtaining the ciphertext can only learn about the public access policy (i.e., [Gender: male or female] AND [Education: M.D. or PH.D.]), while the sensitive information in the secret policy (i.e., [Probability and statistics: statistics or econometrics] AND [Computer science and technology: data mining or machine learning]) including attribute name and its values should be fully hidden. Figures 2 and 3 graphically show this example.

There seems to be a simple solution to protect the privacy of sensitive policy. Specifically, one can use a classic CP-ABE to encrypt the sensitive access policy under the public access policy as first part of the ciphertext and use CP-ABE to encrypt the message under the secret-access-policy as the second part. However, in this method, the cloud can't check whether the end user has sufficient authorities to access the ciphertext, since the ciphertext can only be obtained by authorized end users. Another drawback of this method is that the decryption overhead of the second part ciphertext can't be outsourced to the cloud.

1.2 Contributions

Motivated by the above observation, in this paper, we propose a ciphertext-policy attribute-based encryption with hidden sensitive policy from keyword search techniques in smart city. In our scheme, data owner (employer) publishes an encrypted recruitment for the end user (potential employee) in the cloud. Only authorized end user can access

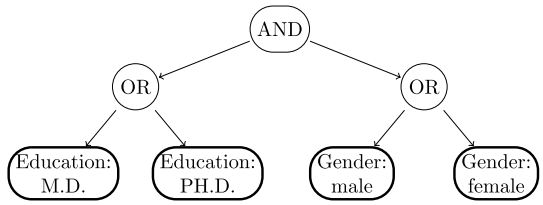

Fig. 2 Public access policy

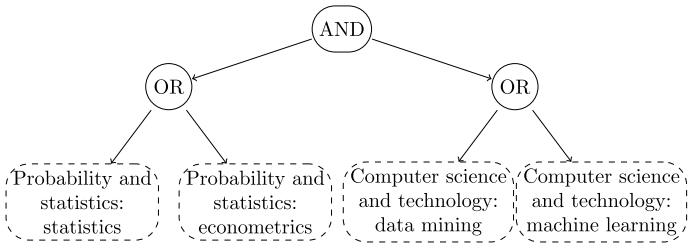

Fig. 3 Secret access policy. The information of attribute and name and its value in the dotted node is fully hidden

the ciphertext, and the privacy of the sensitive access policy is preserved from unauthorized end users. The contribution of our scheme is shown as follows.

- *Hidden sensitive policy* We embed two access policies in the encrypted data, one is public and the other is secret and hidden. When satisfying both policies, he/she can decrypt the ciphertext. For the privacy of sensitive policy, we propose a new security model i.e., *chosen sensitive policy attack* (CSPA): if user's attributes don't satisfy the public policy, he/she cannot learn anything of the secret one. Specifically, the end user generates two sets of randomized secret keys, where each component of the one set corresponds to a public attribute name (or value) and each component of the other set corresponds to an attribute index (user-generated hash value). The end user uploads the two sets to the cloud to check whether he/she has authority to decrypt the ciphertext with double policies. If user satisfies the public policy, the cloud can detect whether the attribute contained in each leaf node of secret policy belongs the user's attributes. This process is essentially a attribute-based keyword search (ABKS) mechanism. By keyword search (KS) methods, the cloud cannot learn about which attribute the leaf node in the access tree of secret policy stands for and unauthorized end users also can't obtain the ciphertext or learn about the secret policy. Therefore, our scheme protects the privacy in sensitive access policy of the ciphertext from unauthorized applicants.
- *Expressive and efficient* Our scheme is expressive and efficient. It supports any monotone access structure instead of restricted policy such as AND-gates on multi-values. The size of the ciphertext scales linearly with the complexity of the access policy. End user doesn't need to test several times, which could be super-polynomial in some previous schemes, before finding the attributes for successful decryption, even if he/she has multiple values for each attribute.

- *Applicable for resource-limited end user* With the help of thousands of fog nodes in the smart city, the computational overhead of data owner generating the sub-ciphertext of the public access policy can be outsourced, and most computational overhead of decryption is shifted from end user to the cloud, leaving a constant number of operations to decrypt the ciphertext. Therefore, it is more suitable for resource-constrained end users.

2 Discussion and result

2.1 Discussion

Sahai et al. [2] first introduced the concept of attribute-based encryption (ABE), which can be divided into two forms: ciphertext-policy ABE (CP-ABE) [4] and key-policy ABE (KP-ABE) [3] . Bethencourt et al. [4] proposed the first CP-ABE scheme, in which the access policy is very expressive and specified by the data owner. From then on, ABE schemes with various functionalities have been widely constructed, e.g., supporting regular languages [10, 11], with unbounded attribute size [10, 12, 13], with constant-size ciphertext [14], with multi-authority [15–17], and with adaptive security [18–20]. One drawback in traditional CP-ABE schemes [4, 5] is that the number of pairing and exponentiation operations for ciphertext decryption is linear with the complexity of access policy, which means the computation cost of end user is quite expensive. This defect in attribute encryption makes it unsuitable for users with resource-constrained devices. To reduce the computation cost of end user, Green et al. [9] provided a new methods for efficiently and securely outsourcing decryption of ABE ciphertexts. In their scheme, most of the heavy cryptographic operations of decryption algorithm are outsourced to the cloud, leaving only a small number of operations for the end user. Li et al. [21] also considered to outsource key-issuing and decryption simultaneously for ABE schemes by introducing two cloud service providers. In the wake of 5G and IoT techniques, fog computing [1] is considered to be a new data resource that can provide high-quality outsourcing services. In fog computing environment, Zuo et al. [22] proposed a practical CP-ABE scheme with outsourced decryption, while Zhang et al. [23] support outsourced encryption, outsourced decryption and attribute update.

However, the access policy must be revealed in most of these schemes, since end users need to know how they combine their secret key components for decryption. This may lead to privacy disclosure, so research on the anonymity of access policies is necessary. Nishide et al. [6] first introduced the concept of partially hidden access policy to achieve anonymity, in which the attribute is split into an attribute name and its values, and only the attribute values are hidden. Some other works [24–26] improved the efficiency and security of [6], but their policies are all restricted with AND-gates on multi-values as in [6]. Later, Lai et al. [27] proposed an expressive fully secure CP-ABE scheme with the LSSS-based partially hidden policy in composite order groups. Based on Lai's scheme, Cui et al. [28] proposed a more efficient one in prime order groups. However, looking for the correct attributes for successful decryption, both [27] and [28] need authorized users to test several times, which cloud be super-polynomial in special cases; for instance, user has many values for each attribute. All the above schemes focus on the partially hidden access policy, while the public attribute names may also lead to the leakage of sensitive information. Some other schemes based on the inner-product predicate encryption [18, 29] and hidden vector encryption

[30] are proposed to protect the policy privacy, but their efficiency is seriously restricted, which means that the size of ciphertext could be super-polynomial.

The keyword search techniques enables the cloud to search over encrypted data without revealing any sensitive information of the keyword. In 2000, Song et al. [31] initially introduced a searchable encryption (SE) technique. Boneh et al. [32] proposed the first public key encryption with keyword search. To achieve both fine-grained access control and keyword search simultaneously, attribute-based encryption with keyword search [33–36] was proposed. Among them, [35, 36] are constructed in fog computing environment.

2.2 Result

In our scheme, sensitive access policy of encrypted recruitment is fully hidden; expressive policy is supported; the ciphertext size is polynomial; the most computational overhead of decryption is outsourced to the cloud server, leaving a constant number of operations for the end user. We summarize the comparisons of various CP-ABE schemes with hidden policy in Table 1.

3 Method

3.1 Preliminaries

In this section, we introduce some background knowledge, which includes access structure, access tree, bilinear maps, Diffie–Hellman assumption and its variants.

3.1.1 Access structures

Definition 1 (*Access structure* [4]) Let $\{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,P_2,\ldots,P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, \ldots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1,P_2,\ldots,P_n\}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.

In this paper, attributes take the role of the parties, and we only focus on the monotone access structure $\mathbb{A}$, which consists of the authorized sets of attributes. Obviously, attributes can directly reflect a user's authority.

Definition 2 (*Access tree* [4]) Let $\mathcal{T}$ be a tree representing an access structure. Each non-leaf node of the tree represents a threshold gate, described by its children and a threshold value. If num_x is the number of children of a node x and k_x is its threshold value, then $0 \leq k_x \leq num_x$. When $k_x = 1$, the threshold gate is an OR gate and when $k_x = num_x$, it is an AND gate. Each leaf node x of the tree is described by an attribute and a threshold value $k_x = 1$.

We introduce a few functions defined in [4] as follows. $parent(x)$ denotes the parent of the node x in the tree. The access tree $\mathcal{T}$ also defines an ordering between the children of every node, that is, the children of a node are numbered from 1 to num. The function $index(x)$ returns such a number associated with the node x, where the index values are uniquely assigned to nodes in the access structure for a given key in an

Table 1 Comparison of CP-ABE schemes with hidden policy

Schemes	Access policy	Policy hidden	Ciphertext size	Decryption time	Decryption outsourced
Nishide et al. [6]	AND-gates on multi-values	Partially hidden	Linear	Deterministic and linear[a]	No
Li et al. [24]	AND-gates on multi-values	Partially hidden	Linear	Deterministic and linear	No
Lai et al. [25]	AND-gates on multi-values	Partially hidden	Linear	Deterministic and linear	No
Zhang et al. [26]	AND-gates on multi-values	Partially hidden	Linear	Deterministic and linear	No
Lai et al. [27]	LSSS	Partially hidden	Linear	Opportunistic and linear[c]	No
Cui et al. [28]	LSSS	Partially hidden	Linear	Opportunistic and linear	No
Lweko et al. [18]	Inner product predicates	Fully hidden	Super-polynomial	Opportunistic and linear	No
Michalevsky et al. [29]	Inner product predicates	Fully hidden	Super-polynomial	Opportunistic and linear	No
Khan et al. [30]	LSSS with hidden vectors	Fully hidden	Super-polynomial	Opportunistic and linear	No
Ours	Tree-based structure	Public policy is exposed and secret policy is fully hidden	Linear	Deterministic and constant[b]	Yes

[a] "Deterministic and linear": end user needs to test fixed the number of times, usually is one, to look for the correct attributes for successful decryption, and the decryption time scales linearly with the complexity of the access policy.

[b] "Deterministic and constant": the test time for the cloud is fixed and the decryption time is constant.

[c] "Opportunistic and linear": several tests may be required, which could be super-polynomial when user has many values for each attribute. The decryption time scales linearly with the complexity of the access policy

arbitrary manner. Each leaf node x corresponds to an attribute a_j, and this relationship should be revealed in the access tree $\mathcal{T}$. To protect the privacy of access policy, we defined the *access tree with hidden attributes* $\widehat{\mathcal{T}}$.

Definition 3 (*Access tree with hidden attributes* $\widehat{\mathcal{T}}$) $\widehat{\mathcal{T}}$ is an access tree with structure the same as normal access trees, except that it doesn't reveal any information about the correspondence between leaf nodes and attributes. It is very easy to check whether a combination of leaf nodes satisfies the access structure, but the information of attribute in these nodes is hidden. So, it is impossible to find out which attributes are embedded in the access tree just from the structure itself.

For better understanding, assuming that x_i, y_i are the indexes of leaf nodes and a_i is the corresponding attribute, the comparison of two kinds of access tree is shown in Figs. 4 and 5. Specifically, the attribute name and its value in the dotted node in Fig. 4 are exposed. However, it is on the contrary in Fig. 5 and only the index of the node is revealed in Fig. 5.

Definition 4 (*Satisfying an access tree* [4]) Let $\mathcal{T}$ be an access tree with root r. Denote by $\mathcal{T}_x$ the subtree of $\mathcal{T}$ rooted at the node x. Hence, $\mathcal{T}$ is the same as $\mathcal{T}_r$. If a set of attributes γ satisfies the access tree $\mathcal{T}_x$, we denote it as $\mathcal{T}_x(\gamma) = 1$. We compute $\mathcal{T}_x(\gamma)$

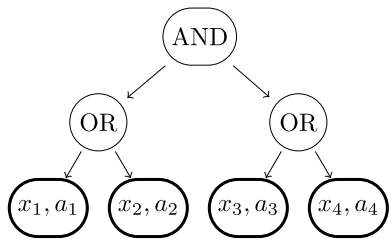

Fig. 4 Normal access tree

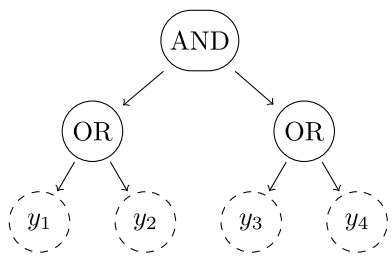

Fig. 5 Access tree with hidden attributes. The information of attribute name and its value in the dotted node is fully hidden, only the index of the node is revealed

recursively as follows. If x is a non-leaf node, evaluate $\mathcal{T}_{x'}(\gamma) = 1$ for all children x' of node x. $\mathcal{T}_x(\gamma)$ returns 1 if and only if at least k_x children return 1. If x is a leaf node, then $\mathcal{T}_x(\gamma)$ returns 1 if and only if $\text{att}(x) \in \gamma$.

3.1.2 Bilinear map and DBDH assumption

We briefly recall the definitions of the bilinear map and the decisional bilinear Diffie–Hellman (DBDH) assumption. Let $\mathbb{G}_0$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}_0$ and e be a efficient computable bilinear map, $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_T$. The bilinear map e has a few properties: (1) Bilinearity: for all $u, v \in \mathbb{G}_0$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$. (2) Non-degeneracy: $e(g, g) \neq 1$. We say that $\mathbb{G}_0$ is a bilinear group if the group operation in $\mathbb{G}_0$ and the bilinear map $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_T$ are both efficiently computable. Notice that the map e is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

Given the bilinear map parameter $(\mathbb{G}_0, \mathbb{G}_T, p, e, g)$ and three random elements $(x, y, z) \in \mathbb{Z}_p^3$, if there is no probabilistic polynomial time (PPT) adversary $\mathcal{B}$ can distinguish between the tuple $(g, g^x, g^y, g^z, e(g, g)^{xyz})$ and the tuple $(g, g^x, g^y, g^z, \vartheta)$, we say that the DBDH assumption holds, where ϑ is randomly selected from $\mathbb{G}_T$. More specifically, the advantage ϵ of $\mathcal{B}$ in solving the DBDH problem is defined as

$$\left|\Pr[\mathcal{A}(g, g^x, g^y, g^z, e(g, g)^{xyz}) = 1] - \Pr[\mathcal{A}(g, g^x, g^y, g^z, \vartheta) = 1]\right|. \tag{1}$$

Definition 5 (*DBDH*) We say that the DBDH assumption holds if no PPT algorithm has a non-negligible advantage ϵ in solving DBDH problem.

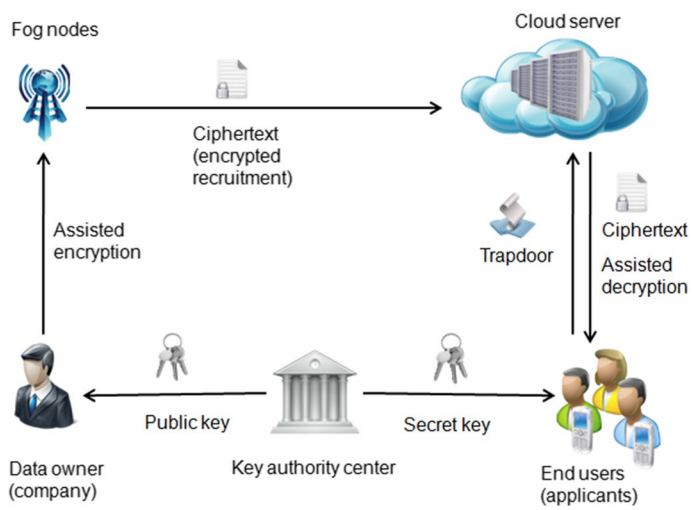

Fig. 6 System description of our scheme

3.2 System and security model

In this section, we introduce the system description, system model, threat model and security model of our scheme.

3.2.1 System description

As shown in Fig. 6, we consider a ciphertext retrieval scenario in fog computing environment. It consists of five parties: Key Authority Center (KAC), Data Owner (DO), Cloud Server (CS), End User (EU), and Fog Nodes (FN). The specific role of each party is given as follows:

- Key Authority Center (KAC): The KAC is a fully trusted third party which is in charge of generating public parameters and secret keys.
- Data Owner (DO): The DO defines the access structure to encrypt a ciphertext CT with the help of fog nodes.
- Cloud Server (CS): The CS has huge computing power and storage capacity; it can provide computing and storage services to both data owner and end user, especially to help end user partially decrypt the ciphertext.
- End User (EU): Resource-constrained user submits a trapdoor to the CS, which will help him/her to partially decrypt the ciphertext.
- Fog Nodes (FN): Some computational overheads can be outsourced from the DO to the fog nodes during the encryption process.

3.2.2 System model

Our scheme includes the following six algorithms:

- $Setup(1^{\lambda}, \mathcal{L}) \rightarrow (PK, MSK)$ Given security parameter λ and a set of all possible attributes $\mathcal{L}$, the KAC generates public key PK and master secret key MSK.

- $KeyGen(PK, MSK, S) \rightarrow SK$ On input the public key PK, the master secret key MSK and an attribute set S, the KAC generates a secret key SK for the EU.
- $Enc(PK, \mathcal{T}_1, \mathcal{T}_2, M) \rightarrow CT$ On input PK, two access policies $\mathcal{T}_1, \mathcal{T}_2$ and message M, with the help of fog nodes, the DO generates the ciphertext CT, in which $\mathcal{T}_1$ is public and $\mathcal{T}_2$ is fully hidden.
- $TrapSK \rightarrow Tr$ The EU generates the trapdoor Tr by his own secret key SK and submits Tr to the CS.
- $Tran(CT, Tr) \rightarrow \widetilde{CT}$ or $\perp$: This algorithm contains two steps:
  - At first, the CS interacts Tr and CT to verify whether the EU has authority to decrypt CT. If $S \nvDash \mathcal{T}_1 \bigvee S \nvDash \mathcal{T}_2$, it outputs $\perp$.
  - If $S \vDash \mathcal{T}_1 \bigwedge S \vDash \mathcal{T}_2$, the CS partially decrypts CT and returns the precomputed ciphertext $\widetilde{CT}$ to the EU.
- **Dec**$(\widetilde{CT}, SK) \rightarrow M$: On input $\widetilde{CT}, SK$, the EU decrypts $\widetilde{CT}$ and outputs M.

3.2.3 Threat model

In this paper, we assume that the KAC is a fully trusted third party, while the CS and FN are honest-but-curious entities, which exactly follow the protocol specifications but also are curious about the sensitive information of ciphertexts and trapdoors. Users are not allowed to collude with CS or FN. Nevertheless, malicious users may collude with each other to access some unauthorized ciphertexts.

3.2.4 Security model

Our scheme achieves chosen plaintext security by the following security game between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

- *Initialization* $\mathcal{A}$ chooses and submits two challenge access policies $\mathcal{T}_1^*$ and $\mathcal{T}_2^*$ to its challenger $\mathcal{C}$.
- *Setup* $\mathcal{C}$ runs **Setup** algorithm and returns the public key PK to $\mathcal{A}$.
- *Phase* 1 $\mathcal{A}$ adaptively submits any attribute set S to $\mathcal{C}$ with the restriction that $(S \nvDash \mathcal{T}_1^* \bigvee S \nvDash \mathcal{T}_2^*)$. In response, $\mathcal{C}$ runs **KeyGen** algorithm and answers $\mathcal{A}$ with the corresponding SK.
- *Challenge* $\mathcal{A}$ chooses two equal-length challenge messages (m_0, m_1) and submits them to $\mathcal{C}$. Then $\mathcal{C}$ picks a random bit $\vartheta \in \{0, 1\}$, runs **Enc** algorithm to encrypt m_ϑ with $\mathcal{T}_1^*$ and $\mathcal{T}_2^*$, and returns the challenge ciphertext CT^* to $\mathcal{A}$.
- *Phase* 2 This phase is the same as Phase 1.
- *Guess* $\mathcal{A}$ outputs a guess bit ϑ' of ϑ. We say that $\mathcal{A}$ wins the game if and only if $\vartheta' = \vartheta$.
  The advantage of $\mathcal{A}$ to win this security game is defined as $\mathrm{Adv}(\mathcal{A}) = \left|\Pr[\vartheta' = \vartheta] - \frac{1}{2}\right|$
  .

Definition 6 Our scheme achieves IND-CPA security if there exists no PPT adversary winning the above security game with a non-negligible advantage ϵ under the DBDH assumption.

In addition, we define a new security model *chosen sensitive policy attack* (CSPA) for our scheme by the following security game between $\mathcal{A}$ and $\mathcal{C}$.

- *Initialization* $\mathcal{A}$ chooses and submits a challenge access structure $\mathcal{T}_1^*$ to its challenger $\mathcal{C}$.
- *Setup* $\mathcal{C}$ runs **Setup** algorithm and gives PK to $\mathcal{A}$.
- *Phase* 1 $\mathcal{A}$ adaptively submits any attribute set S with the restriction that $S \nvDash \mathcal{T}_1^*$. In response, $\mathcal{C}$ runs **Trap** algorithm and responds $\mathcal{A}$ with the corresponding trapdoor Tr.
- *Challenge* $\mathcal{A}$ submits m^* and two challenge hidden policies $\mathcal{T}_2^{0*}$ and $\mathcal{T}_2^{1*}$ with the same structure. Then, $\mathcal{C}$ picks a random bit $\vartheta \in \{0,1\}$ and returns the challenge ciphertext CT^* encrypted with $\mathcal{T}_1^*$ and $\mathcal{T}_2^{\vartheta *}$.
- *Phase* 2 This phase is the same as Phase 1.
- *Guess* $\mathcal{A}$ outputs a guess bit ϑ' of ϑ. We say that $\mathcal{A}$ wins the game if and only if $\vartheta' = \vartheta$.
  The advantage of $\mathcal{A}$ to win this security game is defined as $\mathrm{Adv}(\mathcal{A}) = \left|\Pr[\vartheta' = \vartheta] - \frac{1}{2}\right|$.

Definition 7 Our scheme achieves IND-CSPA security if there exists no PPT adversary winning the above security game with a non-negligible advantage ϵ under the DBDH assumption.

3.3 Construction of our scheme

In this section, we present the concrete construction of CP-ABE scheme with hidden sensitive policy.

Without loss of generality, we suppose that there are n possible attributes in total and $\mathcal{L} = \{a_1, a_2, \ldots, a_n\}$ is the set of all possible attributes. Assume $\mathbb{G}_0, \mathbb{G}_T$ are multiplicative cyclic groups with prime order p and the generator of $\mathbb{G}_0$ is g. Let λ be the security parameter which determines the size of groups. Let $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_T$ be a bilinear map and $H : \{0,1\}^* \longrightarrow \mathbb{Z}_p$ be a hash function which maps any string to a random element of $\mathbb{Z}_p$. We also define the Lagrange coefficient $\Delta_{i,L}(x) = \prod_{j \in L, j \neq i} \frac{x-j}{i-j}$, where $i \in \mathbb{Z}_p$ and a set L of elements in $\mathbb{Z}_p$. The details of our scheme are as follows.

- **Setup**$(1^\lambda, \mathcal{L}) \to (PK, MSK)$: Given a security parameter λ and all possible attributes $\mathcal{L}$, the KAC chooses a bilinear group $\mathbb{G}_0$ with prime order p and generator g. Next, it picks $\alpha, \beta \in_R \mathbb{Z}_p^*$ and $h \in_R \mathbb{G}_0$. For each attribute $a_j \in \mathcal{L}$, it selects $v_j, v_j' \in_R \mathbb{Z}_p^*$. Finally, it generates the public key PK and master secret key MSK as

$$PK = \left\{ \begin{array}{c} \mathbb{G}_0, g, h, g^{\alpha}, e(g,g)^{\beta}, e(g,h)^{\beta}, \\ \left\{ g^{v_j}, h^{v_j}, g^{v'_j}, h^{v'_j}, a_j \mid \forall a_j \in \mathcal{L} \right\} \end{array} \right\}; \tag{2}$$

$$MSK = \left\{ \alpha, \beta, \left\{ v_j, v'_j, a_j \mid \forall a_j \in \mathcal{L} \right\} \right\}. \tag{3}$$

- **KeyGen**$(MSK, S) \rightarrow SK$: While receiving an attribute set S from the EU, the KAC selects $r, r' \in_R \mathbb{Z}_p^*$ and returns the secret key SK as

$$SK = \left\{ \begin{array}{c} g^{\beta+\alpha r}, h^{\beta+\alpha r}, g^{\alpha r} h^{r'}, h^{\alpha r + r'}, g^{r'}, \\ \{ g^{\frac{\alpha r}{v_j}}, h^{\frac{\alpha r}{v_j}}, g^{\frac{\alpha r}{v'_j}}, h^{\frac{\alpha r}{v'_j}}, a_j \mid \forall a_j \in S \} \end{array} \right\}. \tag{4}$$

- **Enc**$(PK, \mathcal{T}_1, \mathcal{T}_2, M) \rightarrow CT$: The DO chooses $ck \in_R \mathbb{Z}_p^*$ as a symmetric encryption key and encrypts message M with ck, $E_{ck}(M)$, by using symmetric encryption (AES). Then, it encrypts ck with the help of the FN as follows:

1 The DO sends $\mathcal{T}_1$ to the FN. The FN randomly choose a polynomial q_x for each node x of $\mathcal{T}_1$ from the root node R_1 in a top-down manner: for each node x of $\mathcal{T}_1$, the degree of q_x is $d_x = k_x - 1$, where k_x is the threshold value of x; beginning with root node R_1, the FN pick $s_1 \in_R \mathbb{Z}_p^*$, sets $q_{R_1}(0) = s_1$ and randomly chooses $d_{R_1} = k_{R_1} - 1$ other points of q_{R_1} to define the polynomial completely; for any other node x, they set $q_x(0) = q_{parent(x)}(index(x))$ and choose d_x other points to define q_x completely. The FN generate the CT'_1 as

$$CT'_1 = \left\{ \begin{array}{c} \mathcal{T}_1, g^{s_1}, h^{s_1}, \{ C_1^x = g^{v_j q_x(0)}, \\ C_2^x = h^{v_j q_x(0)}, x \mid \forall x \in \mathcal{X}_1 \} \end{array} \right\}, \tag{5}$$

where $\mathcal{X}_1$ is a set of attributes corresponding to all leaf nodes in $\mathcal{T}_1$ and each $x \in \mathcal{X}_1$ is corresponding to attribute a_j. Note that $\mathcal{T}_1$ is stored in CT'_1 in the form of plaintext, so the privacy of the access policy of $\mathcal{T}_1$ is exposed.

2 The DO picks $s_2 \in_R \mathbb{Z}_p^*$ and sends $g^{s_2}, h^{s_2}, e(g,h)^{\beta s_2}$ to FN to generate CT_1 as

$$CT_1 = \{ g^{s_2}, h^{s_2}, e(g,h)^{\beta s_2}, CT'_1 \}. \tag{6}$$

3 The DO picks $s_3, s_4 \in_R \mathbb{Z}_p^*$ and generates the ciphertext CT_2 corresponding to the $\mathcal{T}_2$ in the same way of CT_1. For each node y of $\mathcal{T}_2$, $q_y(0)$ and d_y are defined exactly the same with above $q_x(0)$ and d_x; for the root node R_2 of $\mathcal{T}_2$, $q_{R_2}(0) = s_3$. Then,

$$CT_2 = \{ g^{s_4}, h^{s_4}, CT'_2 \}, \tag{7}$$

and

$$CT'_2 = \left\{ \begin{array}{c} \widehat{\mathcal{T}}_2, g^{s_3}, h^{s_3}, \{ C_1^y = g^{v'_j (q_y(0) - s_2)}, \\ C_2^y = h^{v'_j (q_y(0) - s_2)}, C_3^y = g^{q_y(0)}, \\ C_4^y = h^{q_y(0)}, C_5^y = e(g,h)^{\beta q_y(0)}, y \mid \forall y \in \mathcal{X}_2 \} \end{array} \right\}, \tag{8}$$

where each leaf node y is corresponding to attribute a_j and $\mathcal{X}_2$ is a set of all leaf nodes in $\mathcal{T}_2$. Since $\widehat{\mathcal{T}_2}$ is defined the same as $\mathcal{T}_2$, except that the attribute name and its value in the $\widehat{\mathcal{T}_2}$ are fully hidden. Therefore, the privacy of $\mathcal{T}_2$ can be preserved.

4 The DO computes $C = ck \cdot e(g,g)^{\beta(s_2+s_4)}$ and sends $E_{ck}(M), C, CT_2$ to FN, which generate and upload to the final ciphertext CT to the CS, where

$$CT = \left\{\mathcal{T}_1, \widehat{\mathcal{T}_2}, E_{ck}(M), C, CT_1, CT_2\right\}. \tag{9}$$

- **Trap**$(SK) \rightarrow Tr$: The **Trap** algorithm proceeds as follows. The EU chooses $x', y', z', k \in_R \mathbb{Z}_p^*$ and computes the attribute index $H_j = H(k \parallel j)$ for each $a_j \in S$, and generates the following trapdoor Tr with its SK as

$$Tr = \left\{\begin{array}{c} T_0 = x' + y', T_1 = h^{(\alpha r + r')(x'+y')+z'}, T_2 = g^{r'(x'+y')+z'}, \\ T_3 = g^{(\beta+\alpha r)x'}, T_4 = h^{(\beta+\alpha r)y'}, T_5 = g^{\alpha r x'} h^{r'x'}, T_6 = g^{r'x'}, \\ \{T_{10}^j = g^{\frac{\alpha r x'}{v_j}}, T_{11}^j = h^{\frac{\alpha r y'}{v_j}}, a_j \mid \forall a_j \in S\}, \\ \{T_{12}^{H_j} = g^{\frac{\alpha r x'}{v_j'}}, T_{13}^{H_j} = h^{\frac{\alpha r y'}{v_j'}}, H_j \mid \forall a_j \in S\} \end{array}\right\}, \tag{10}$$

where the set $\{T_{12}^{H_j}, T_{13}^{H_j}, H_j \mid \forall a_j \in S\}$ is sorted by H_j. Due to the hash functions, the corresponding relationship between the tuple $\{T_{12}^{H_j}, T_{13}^{H_j}, H_j\}$ and attribute a_j is preserved. The EU keeps x', k secret and sends Tr to the CS.

- **Tran**$(CT, Tr) \rightarrow \widetilde{CT}$ or $\perp$: This algorithm conducts the following steps: access verification and ciphertext precomputation.

  – **Access verification** Because this process is a recursive procedure, we first define a recursive algorithm $F_x = F_x(C_1^x, C_2^x, T_{10}^j, T_{11}^j, x)$ intaking (C_1^x, C_2^x, x) in CT_1 and (T_{10}^j, T_{11}^j) in Tr, respectively.

  For each node of $\mathcal{T}_1$, the CS runs a recursive algorithm as follows:

  (1) If x is a leaf node of $\mathcal{T}_1$. Let a_j is the corresponding attributes of node x. If $a_j \in S$, the CS computes

$$\begin{aligned} F_x &= e(C_2^x, T_{10}^j) \cdot e(C_1^x, T_{11}^j) \\ &= e(h^{v_j q_x(0)}, g^{\frac{\alpha r x'}{v_j}}) \cdot e(g^{v_j q_x(0)}, h^{\frac{\alpha r y'}{v_j}}) \\ &= e(g,h)^{\alpha r q_x(0)(x'+y')}. \end{aligned} \tag{11}$$

  If $a_j \notin S$, set $F_x = null$.

  (2) If x is a non-leaf node, the recursive algorithm is defined as: for all child nodes z of x, where a_i is the corresponding attributes of node z, the CS computes $F_z = F_z(C_1^z, C_2^z, T_{10}^i, T_{11}^i, z)$ recursively. Let S_x be an arbitrary k_x-sized set of child nodes z satisfying $F_z \neq null$. If S_x does not exist, $F_x = null$. Otherwise, the CS calculates

$$\begin{aligned} F_x &= \prod_{z \in S_x} F_z^{\Delta_{i,S'_x}(0)} \\ &= \prod_{z \in S_x} (e(g,h)^{\alpha r(x'+y') q_{parent(z)}(index(z))})^{\Delta_{i,S'_x}(0)} \\ &= e(g,h)^{\alpha r q_x(0)(x'+y')}, \end{aligned} \tag{12}$$

where $i = index(z)$ and $S'_x = \{index(z) \mid \forall z \in S_x\}$.
By calling the above algorithm on the root node R_1 of $\mathcal{T}_1$, the CS gets $F_{R_1} = e(g,h)^{\alpha r s_1 (x'+y')}$. Then, the CS computes D as

$$\begin{aligned} D &= \frac{e(T_1, g^{s_1+s_2})}{F_{R_1} \cdot e(T_2, h^{s_1+s_2})} \\ &= \frac{e(h^{(\alpha r + r')(x'+y')+z'}, g^{s_1+s_2})}{e(g,h)^{\alpha r s_1 (x'+y')} \cdot e(g^{r'(x'+y')+z'}, h^{s_1+s_2})} \\ &= e(g,h)^{\alpha r s_2 (x'+y')}. \end{aligned} \tag{13}$$

Then, for each leaf node $y \in \widehat{\mathcal{T}_2}$, the CS defines

$$F_{y,H_j} = e(C_1^y, T_{13}^{H_j}) \cdot e(C_2^y, T_{12}^{H_j}), \tag{14}$$

and checks whether there exists an $H_j \in \{H_j\}_{a_j \in S}$ such that

$$C_5^{y^{T_0}} \cdot F_{y,H_j} \cdot D = e(T_3, C_4^y) \cdot e(T_4, C_3^y). \tag{15}$$

If $S \models \mathcal{T}_1$ and there exists an $H_j \in \{H_j\}_{a_j \in S}$ such that the a_j is the corresponding attribute of leaf node y, then

$$\begin{aligned} & C_5^{y^{T_0}} \cdot F_{y,j} \cdot D \\ &= e(g,h)^{\beta q_y(0)(x'+y')} \cdot e(g^{v'_j(q_y(0)-s_2)}, h^{\frac{\alpha r y'}{v'_j}}) \cdot \\ &\quad e(h^{v'_j(q_y(0)-s_2)}, g^{\frac{\alpha r x'}{v'_j}}) \cdot e(g,h)^{\alpha r s_2 (x'+y')} \\ &= e(g,h)^{(\beta+\alpha r) q_y(0)(x'+y')} \\ &= e(g^{(\beta+\alpha r)x'}, h^{q_y(0)}) \cdot e(h^{(\beta+\alpha r)y'}, g^{q_y(0)}) \\ &= e(T_3, C_4^y) \cdot e(T_4, C_3^y). \end{aligned} \tag{16}$$

By running the above functions recursively, the CS can find out whether this EU has the attribute corresponding to each leaf node of $\widehat{\mathcal{T}_2}$ and then check out whether it has the authority to access the ciphertext CT, e.t., $S \models \mathcal{T}_1$ and $S \models \widehat{\mathcal{T}_2}$. If CT is accessible, CS outputs the following Table 2. Otherwise, the algorithm outputs $\perp$. Assume that the EU has the attributes corresponding to t leaf nodes in $\widehat{\mathcal{T}_2}$.

– *Ciphertext precomputation* If CT is accessible, i.e., $S \models \mathcal{T}_1$ and $S \models \widehat{\mathcal{T}_2}$, the algorithm is similar to the recursive procedure defined in the above algorithm.
1. $S \models \mathcal{T}_1$. For each node of $\mathcal{T}_1$.
(1) If x is a leaf node of $\mathcal{T}_1$. If $a_j \in S$, the CS sets $G_x = G_x(C_1^x, T_{10}^j, x)$ and computes

Table 2 Correspondence between leaf node and attribute index

Leaf nodes of $\widehat{\mathcal{T}_2}$	Hash value of the index of the corresponding attribute
y_1	H_{j_1}
$\vdots$	$\vdots$
y_t	H_{j_t}

$$G_x = e(C_1^x, T_{10}^j) = e(g^{v_j q_x(0)}, g^{\frac{\alpha r x'}{v_j}}) = e(g,g)^{\alpha r q_x(0) x'}. \tag{17}$$

If $a_j \notin S$, set $G_x = null$.

(2) If x is a non-leaf node, for all child nodes z of x, the CS computes $G_z = G_x(C_1^z, T_{10}^j, z)$ recursively. Let S_x be an arbitrary k_x-sized set of child nodes z satisfying $G_z \neq null$. If S_x does not exist, $G_x = null$. Otherwise, the CS calculates

$$\begin{aligned} G_x &= \prod_{z \in S_x} G_z^{\Delta_{i,S_x'}(0)} \\ &= \prod_{z \in S_x} (e(g,g)^{\alpha r x' q_{parent(z)}(index(z))})^{\Delta_{i,S_x'}(0)} \\ &= e(g,g)^{\alpha r q_x(0) x'}, \end{aligned} \tag{18}$$

where $i = index(z)$ and $S_x' = \{index(z) \mid \forall z \in S_x\}$. Then CS gets $G_{R_1} = e(g,g)^{\alpha r s_1 x'}$ and computes

$$\begin{aligned} A' &= \frac{G_{R_1} \cdot e(T_6, h^{s_1+s_2})}{e(T_5, g^{s_1+s_2})} \\ &= \frac{e(g,g)^{\alpha r s_1 x'} \cdot e(g^{r'x'}, h^{s_1+s_2})}{e(g^{\alpha r x'} h^{r'x'}, g^{s_1+s_2})} \\ &= e(g,g)^{\alpha r s_2 x'}. \end{aligned} \tag{19}$$

$$A = \frac{e(T_3, g^{s_2})}{A'} = \frac{e(g^{(\beta+\alpha r)x'}, g^{s_2})}{e(g,g)^{\alpha r s_2 x'}} = e(g,g)^{\beta s_2 x'}. \tag{20}$$

2. $S \models \widehat{\mathcal{T}_2}$. For each y_i in Table 2, the CS sets $G_{y_i} = G_{y_i}(C_1^{y_i}, T_{10}^{H_{j_i}}, y_i, A)$ and computes

$$\begin{aligned} G_{y_i} &= e(g^{v'_{j_i}(q_{y_i}(0)-s_2)}, g^{\frac{\alpha r x'}{v'_{j_i}}}) \cdot A \\ &= e(g,g)^{\alpha r q_{y_i}(0) x'}. \end{aligned} \tag{21}$$

Since $S \models \widehat{\mathcal{T}_2}$, for the root node R_2 of $\widehat{\mathcal{T}_2}$, the CS can compute $G_{R_2} = e(g,g)^{\alpha r s_3 x'}$ in a recursive manner, and then computes

$$
\begin{aligned}
B &= \frac{e(T_3, g^{s_4}) \cdot e(T_5, g^{s_3+s_4})}{G_{R_2} \cdot e(T_6, h^{s_3+s_4})} \\
&= \frac{e(g^{(\beta+\alpha r)x'}, g^{s_4}) \cdot e(g^{(\alpha r x'} h^{r'x'}, g^{s_3+s_4})}{e(g,g)^{\alpha r s_3 x'} \cdot e(g^{r'x'}, h^{s_3+s_4})} \\
&= e(g,g)^{\beta s_4 x'}.
\end{aligned} \tag{22}
$$

Finally, the CS returns the precomputed ciphertext

$$\widetilde{CT} = \{E_{ck}(M), A, B, C = ck \cdot e(g,g)^{\beta(s_2+s_4)}\}$$

to the EU.

- **Dec**$((\widetilde{CT}, x') \rightarrow M$: The EU derives the symmetric secret key ck as

$$ck = \frac{C}{(AB)^{\frac{1}{x'}}} = \frac{ck \cdot e(g,g)^{\beta(s_2+s_4)}}{e(g,g)^{\frac{\beta(s_2+s_4)x'}{x'}}}, \tag{23}$$

and uses ck to decrypt $E_ck(M)$ by symmetric decryption.

Remark 1 Different from previous schemes, we let the attribute a_j (or leaf node x, attribute index H_j) appear in the component of secret key (or ciphertext, trapdoor) just to make it clearer which attribute (or leaf node, attribute index) the component of secret key (or ciphertext, trapdoor) corresponds to.

Remark 2 Once the EU has access to the ciphertext, the CS will find out that the leaf nodes in the access tree must correspond to some attributes in S. Therefore, we made a small modification to the **Trap** algorithm, replacing $\{T_{10}^j, T_{11}^j, a_j \mid \forall a_j \in S\}$ with $\{T_{10}^j, T_{11}^j, a_j \mid \forall a_j \in S_1\}$, where $S_1 \subseteq S$. Since the public access policy is revealed in the ciphertext, the end user can decide the attribute set S_1 by their own. In addition, the **Trap** algorithm can be run offline while the device is charging.

Remark 3 The computational overhead of the CS to run the **Access Verification** algorithm scales linearly with $\mid T_1 \mid + \mid S \mid \cdot \mid T_2 \mid$, supposed that $\mid S \mid$ is the number of attributes the EU owned and $\mid T_1 \mid, \mid T_2 \mid$ is the number of leaf nodes in the access tree $\mathcal{T}_1, \widehat{\mathcal{T}}_2$, respectively. Thus, it does not need a super-polynomial time to find out the correct attributes for successful decryption.

3.4 Analysis of our scheme

In this section, we provide a formal security analysis of our scheme.

3.4.1 Security analysis

Theorem 1 *Supposed that a PPT adversary $\mathcal{A}$ can break the IND-CPA security of our scheme with a non-negligible advantage $\epsilon > 0$, then there exists a PPT simulator $\mathcal{B}$ that can distinguish a DBDH tuple from a random tuple with an advantage $\frac{\epsilon}{2}$.*

Proof Given the bilinear map parameter $(\mathbb{G}_0, \mathbb{G}_T, p, e, g)$. The DBDH challenger $\mathcal{C}$ selects $a', b', c' \in \mathbb{Z}_p$, $\theta \in \{0,1\}$, $\mathcal{R} \in \mathbb{G}_T$ at random. Let $\mathcal{Z} = e(g,g)^{a'b'c'}$, if $\theta = 0$, $\mathcal{R}$ else. Next, $\mathcal{C}$ sends $\mathcal{B}$ the tuple $\langle g, g^{a'}, g^{b'}, g^{c'}, \mathcal{Z}\rangle$. Then, $\mathcal{B}$ plays the role of challenger in the following security game.

- *Initialization* $\mathcal{A}$ submits two challenge access policy $\mathcal{T}_1^*$ and $\mathcal{T}_2^*$ to $\mathcal{B}$.
- *Setup* $\mathcal{B}$ chooses $\beta', x \in \mathbb{Z}_p$ at random and sets $h = g^x$, $g^\alpha = g^{a'}$, $e(g,g)^\beta = e(g,g)^{\beta'+a'b'} = e(g,g)^{\beta'} e(g^{a'}, g^{b'})$, $e(g,h)^\beta = e(g,g)^{\beta x}$. For each attribute $a_j \in \mathcal{L}$, $\mathcal{B}$ picks $s_j, s_j' \in_R Z_p$. If $a_j \in \mathcal{T}_1^*$, set $g^{v_j} = g^{\frac{a'}{s_j}}$, otherwise $g^{v_j} = g^{s_j}$; if $a_j \in \mathcal{T}_2^*$, set $g^{v_j'} = g^{\frac{a'}{s_j'}}$, otherwise $g^{v_j'} = g^{s_j'}$. The $\mathcal{B}$ sets $h^{v_j} = g^{v_j x}, h^{v_j'} = g^{v_j' x}$ and sends PK to $\mathcal{A}$, where

$$PK = \left\{ \begin{array}{l} \mathbb{G}_0, g, h, g^\alpha, e(g,g)^\beta, e(g,h)^\beta, \\ \{g^{v_j}, h^{v_j}, g^{v_j'}, h^{v_j'}, a_j \mid \forall a_j \in \mathcal{L}\} \end{array} \right\}. \tag{24}$$

- *Phase* 1 $\mathcal{A}$ adaptively submits any attribute set $S \in \mathcal{L}$ to $\mathcal{B}$ with the restriction that $(S \nvDash \mathcal{T}_1^* \bigvee S \nvDash \mathcal{T}_2^*)$. In response, $\mathcal{B}$ picks $\hat{r}, \tilde{r} \in \mathbb{Z}_p$ at random, computes $g^r = \frac{g^{\hat{r}}}{g^{b'}} = g^{\hat{r}-b'}$, $g^{\beta+\alpha r} = g^{\beta'+a'b'+a'(\hat{r}-b')} = g^{\beta'+a'\hat{r}}$, $h^{\beta+\alpha r} = g^{(\beta'+a'\hat{r})x} = h^{\beta'+a'\hat{r}}$, $g^{\alpha\hat{r}} h^{\tilde{r}}$, $h^{\alpha\hat{r}} h^{\tilde{r}}, g^{\tilde{r}}$. For each $a_j \in S$, if $a_j \in \mathcal{T}_1^*$, $\mathcal{B}$ computes $g^{\frac{\alpha\hat{r}}{v_j}} = g^{s_j\hat{r}}$, $h^{\frac{\alpha\hat{r}}{v_j}} = h^{s_j\hat{r}}$, otherwise $g^{\frac{\alpha\hat{r}}{v_j}} = g^{\frac{a'\hat{r}}{s_j}}$, $h^{\frac{\alpha\hat{r}}{v_j}} = h^{\frac{a'\hat{r}}{s_j}}$; if $a_j \in \mathcal{T}_2^*$, $\mathcal{B}$ sets $g^{\frac{\alpha\hat{r}}{v_j'}} = g^{s_j'\hat{r}}$, $h^{\frac{\alpha\hat{r}}{v_j'}} = h^{s_j'\hat{r}}$, otherwise $g^{\frac{\alpha\hat{r}}{v_j'}} = g^{\frac{a'\hat{r}}{s_j'}}$, $h^{\frac{\alpha\hat{r}}{v_j'}} = h^{\frac{a'\hat{r}}{s_j'}}$.
  Afterward, $\mathcal{B}$ answers $\mathcal{A}$ with the corresponding secret key

$$SK = \left\{ \begin{array}{l} g^{\beta'+\alpha\hat{r}}, h^{\beta'+a'\hat{r}}, g^{\alpha\hat{r}} h^{\tilde{r}}, h^{\alpha\hat{r}} h^{\tilde{r}}, g^{\tilde{r}}, \\ \{g^{\frac{\alpha\hat{r}}{v_j}}, h^{\frac{\alpha\hat{r}}{v_j}}, g^{\frac{\alpha\hat{r}}{v_j'}}, h^{\frac{\alpha\hat{r}}{v_j'}}, a_j \mid \forall a_j \in S\} \end{array} \right\}. \tag{25}$$

- *Challenge* $\mathcal{A}$ submits two equal-length challenge messages $\{m_0, m_1\}$ to $\mathcal{B}$. Then, $\mathcal{B}$ chooses $s_1, s_2 \in_R \mathbb{Z}_p$ and generates

$$CT_1'^* = \left\{ \begin{array}{l} \mathcal{T}_1^*, g^{s_1}, h^{s_1}, \{C_{j,1} = g^{v_j q_{x^*}(0)}, \\ C_{j,2} = h^{v_j q_{x^*}(0)}, x^* \mid \forall x^* \in \mathcal{X}_1^*\} \end{array} \right\},$$

  where $\mathcal{X}_1^*$ is a set of attributes corresponding to all leaf nodes in $\mathcal{T}_1^*$ and each $x^* \in \mathcal{X}_1^*$ is corresponding to attribute a_j. The $\mathcal{B}$ sets

$$CT_1^* = \{g^{s_2}, h^{s_2}, e(g,h)^{\beta s_2}, CT_1'^*\}, \tag{26}$$

  and generates CT_2^* in a similar method. $\mathcal{B}$ picks $s_3 \in_R \mathbb{Z}_p$ and sets $g^{s_4} = \frac{g^{c'}}{g^{s_2}}$, $h^{s_4} = g^{s_4 x}$, $e(g,g)^{\beta(s_2+s_4)} = \mathcal{Z} \cdot e(g,g)^{\beta' c'}$. So,

$$CT_2^* = \{g^{s_4}, h^{s_4}, CT_2^{'*}\}, \tag{27}$$

where

$$CT_2^{'*} = \left\{ \begin{array}{c} \widehat{\mathcal{T}_2^*}, g^{s_3}, h^{s_3}, \{C_1^y = g^{v_j'(q_{y^*}(0)-s_2)}, \\ C_2^y = h^{v_j'(q_{y^*}(0)-s_2)}, C_3^y = g^{q_{y^*}(0)}, \\ C_4^y = h^{q_{y^*}(0)}, C_5^y = e(g,h)^{\beta q_{y^*}(0)}, y \mid \forall y \in \mathcal{X}_2\} \end{array} \right\}. \tag{28}$$

Finally, $\mathcal{B}$ randomly picks $\theta' \in \{0,1\}$, sets $C^* = m_{\theta'} \cdot \mathcal{Z} \cdot e(g,g)^{\beta' c'}$, and returns $\mathcal{A}$ the final challenge ciphertext $CT^* = \{\mathcal{T}_1^*, \widehat{\mathcal{T}_2^*}, C^*, CT_1^*, CT_2^*\}$.

- *Phase* 2 This phase is the same as Phase 1.
- *Guess* $\mathcal{A}$ outputs a guess bit θ'' of θ'. If $\theta'' = \theta'$, $\mathcal{B}$ guesses $\theta = 0$ which indicates that $\mathcal{Z} = e(g,g)^{a'b'c'}$ in the above game. Otherwise, $\mathcal{B}$ guesses $\theta = 1$, i.e., $\mathcal{Z} = \mathcal{R}$.

If $\mathcal{Z} = \mathcal{R}$, then CT^* is random from the view of $\mathcal{A}$. Hence, $\mathcal{B}'s$ probability to guess θ correctly is

$$\Pr\left[\mathcal{B}\left(g, g^{a'}, g^{b'}, g^{c'}, \mathcal{Z} = \mathcal{R}\right) = 1\right] = \frac{1}{2}. \tag{29}$$

Else $\mathcal{Z} = e(g,g)^{a'b'c'}$, then CT^* is available and $\mathcal{A}'s$ advantage of guessing θ' is ϵ. Therefore, $\mathcal{B}'s$ probability to guess θ correctly is

$$\Pr\left[\mathcal{B}\left(g, g^{a'}, g^{b'}, g^{c'}, \mathcal{Z} = e(g,g)^{a'b'c'}\right) = 0\right] = \frac{1}{2} + \epsilon. \tag{30}$$

In conclusion, $\mathcal{B}'s$ advantage to win the above security game is

$$\begin{aligned} \mathrm{Adv}(\mathcal{B}) = \frac{1}{2}\Big(&\Pr\left[\mathcal{B}\left(g, g^{a'}, g^{b'}, g^{c'}, \mathcal{Z} = e(g,g)^{a'b'c'}\right) = 0\right] \\ &+ \Pr\left[\mathcal{B}\left(g, g^{a'}, g^{b'}, g^{c'}, \mathcal{Z} = \mathcal{R}\right) = 1\right]\Big) - \frac{1}{2} = \frac{1}{2}\epsilon. \end{aligned} \tag{31}$$

□

Theorem 2 *Supposed that a PPT adversary $\mathcal{A}$ can break the IND-CSPA security of our scheme with a non-negligible advantage $\epsilon > 0$, then there exists a PPT simulator $\mathcal{B}$ that can distinguish a DBDH tuple from a random tuple with an advantage $\frac{\epsilon}{2}$.*

Proof The proof process of this theorem is similar to that of Theorem 1. The DBDH challenger $\mathcal{C}$ sends $\mathcal{B}$ the tuple $\langle g, g^{a'}, g^{b'}, g^{c'}, \mathcal{Z}\rangle$, in which $\mathcal{Z} = e(g,g)^{a'b'c'}$ or $\mathcal{R}$. $\mathcal{A}$ chooses a challenge access structure $\mathcal{T}^*$ initially. $\mathcal{B}$ returns public key in the same way as in Theorem 1. Then $\mathcal{A}$ adaptively submits any attribute set S with the restriction that $S \nvDash \mathcal{T}^*$. Since $\mathcal{B}$ can generate secret keys as in Theorem 1, it can naturally answer $\mathcal{A}$ with the corresponding trapdoor Tr. In the challenge phase, $\mathcal{A}$ submits m^* and two challenge policies $\mathcal{T}_0^*$ and $\mathcal{T}_1^*$ with the same structure, i.e., $\widehat{\mathcal{T}_0^*} = \widehat{\mathcal{T}_1^*}$. $\mathcal{B}$ randomly picks $\theta' \in \{0,1\}$, generates

CT_2^* with $\mathcal{T}_{\theta'}^*$ and returns the challenge ciphertext CT^*. If $\mathcal{A}$'s advantage of guessing θ' is ϵ, then $\mathcal{B}$'s advantage to distinguish a DBDH tuple from a random tuple is $\frac{\epsilon}{2}$. □

Remark 4 In Definition 6, it demonstrates the IND-CPA security of our scheme. Since the adversary needs to satisfies both $\mathcal{T}_1^*$ and $\mathcal{T}_2^*$ to decrypt the ciphertext, the restriction on the private key inquiry in Theorem 1 is $(S \nvDash \mathcal{T}_1^* \bigvee S \nvDash \mathcal{T}_2^*)$, otherwise the adversary can break our scheme trivially. While in Definition 7, it is required that the adversary cannot tell the challenge sensitive policy $\mathcal{T}_0^*$ apart from $\mathcal{T}_1^*$ if it doesn't satisfy the public access policy $\mathcal{T}^*$. So the restriction on the private key inquiry in Theorem 2 is $S \nvDash \mathcal{T}^*$, otherwise the adversary can break our scheme trivially.

3.4.2 Complexity analysis

In this section, we compare the computational overhead with some other related schemes with hidden policy from a technical point of view. Some schemes only support AND-gate policies, to represent an expressive policy f, we transmute it to a disjunctive normal form and $f = f_1 \bigvee \cdots \bigvee f_n$ and then represent each f_i by a conjunctive clause as $f_i = \text{att}_{i,1} \bigwedge \cdots \bigwedge \text{att}_{i,l}$. Without loss of generality, we suppose each attribute att_i has two values $\text{att}_{i,1}, \text{att}_{i,2}$ and consider such a simple access policy

$$f = (\text{att}_{1,1} \bigvee \text{att}_{1,2}) \bigwedge (\text{att}_{2,1} \bigvee \text{att}_{2,2}) \ldots \bigwedge (\text{att}_{n,1} \bigvee \text{att}_{n,2}),$$

which means that there are 2^n access strategies. Now, we discuss the computational overhead of each scheme in Table 3.

As shown in Table 3, schemes [6, 24, 25] restricted with AND-gates on multi-values give rise to an exponential size of ciphertext for supporting an expressive access policy. Since the attribute value is hidden in [28], for finding the correct attribute value for successful decryption, [28] needs end user to test super-polynomial times. In addition, all the above schemes only protect the privacy attribute value, but expose the information of attribute name. Some other schemes based on the inner-product predicate encryption [18, 29] that we haven't discussed in detail here, because transform an inner-product predicate to an expressive access policy will also cause an exponential size of ciphertext. In our scheme, the size of ciphertext, test time and decryption time are all polynomial. Partial overhead for generating the ciphertext is outsourced to fog nodes and most overhead during test and decryption is outsourced from end user to the cloud.

4 Conclusion

In this work, we provide a novel method to protect the privacy of sensitive policy in CP-ABE scheme. With the help of KS techniques,the access policy in our scheme is expressive and compact, and the computational overhead of encryption (half part) and decryption (most part) can be sifted to fog nodes and cloud server respectively. Hence,

Table 3 Computational overhead among various schemes

Schemes	Access policy	Hidden policy	Ciphertext size	Test time	Decryption time	Group order
Nishide et al. [6]	AND-gates on multi-values	Partially hidden	$2^n[\|\|G_T\|\| + (4n+1)\|\|G\|\|]$	User: super-polynomial	User: $(3n+1)e$	p
Li et al. [24]	AND-gates on multi-values	Partially hidden	$2^n[\|\|G_T\|\| + 8n\|\|G\|\|]$	User: super-polynomial	user: $4ne$	p
Lai et al. [25]	AND-gates on multi-values	Partially hidden	$2^n[\|\|G_T\|\| + (4n+2)\|\|G\|\|]$	user: super-polynomial	User: $(n+1)e$	pqr
Cui et al. [28]	LSSS	Partially hidden	$\|\|G_T\|\| + (6n+2)\|\|G\|\|$	User: super-polynomial	User: $(6n+1)e$	p
Ours	Tree-based structure	Secret policy is fully hidden	Fog: $(2+2n_1)\|\|G\|\|$ user: $(n_2+2)\|\|G_T\|\| + (4+4n_2)\|\|G\|\|$	Cloud: $2(n_1+n_2+nn_2+1)e$	Cloud: $(n_1+n_2+6)e$ User: no pairing	p

$\|\|G_T\|\|$: the size of group element of $\mathbb{G}_T$. $\|\|G\|\|$: the size of group element of $\mathbb{G}_0$

e, Bilinear pairing; *n*, number of possible attributes in the access policy; *m*, number of possible values of each attribute; n_1, number of attributes in the public access policy; n_2, number of attributes in the secret access policy. $n_1 + n_2 = n$

Table 4 List of abbreviations

Complete spellings	Abbreviations	Complete spellings	Abbreviations
Attribute-based encryption	ABE	Internet of things	IoT
ciphertext-policy attribute-based encryption	CP-ABE	Data owner	DO
Key-policy ABE	KP-ABE	Cloud server	CS
Decisional bilinear Diffie–Hellman	DBDH	End user	EU
Inner-product predicate encryption	IPE	Fog nodes	FN
Chosen sensitive policy attack	CSPA	Key authority center	KAC

our scheme is more friendly for resource-limited mobile devices. The list of abbreviations is shown in Table 4.

Acknowledgements
Not applicable.

Authors' contributions
Fei Meng, Leixiao Cheng and Mingqiang Wang are the main authors of the current paper. They contributed to the development of the ideas, design of the study, theory, result analysis, and paper writing. All authors read and approved the final manuscript.

Funding
The authors are supported by National Cryptography Development Fund (Grant No. MMJJ20180210) and National Natural Science Foundation of China (Grant Nos. 61832012 and 61672019).

Competing interests
The authors declare that they have no competing interests.

Author details
[1] School of Mathematics, Shandong University, South Shanda Road, No. 27, Jinan 250100, China. [2] Key Laboratory of Cryptologic Technology and Information Security, Ministry of Education, South Shanda Road, No. 27, Jinan 250100, China.

Received: 12 August 2020 Accepted: 2 December 2020
Published online: 03 February 2021

References

1. F. Bonomi, R.A. Milito, J. Zhu, S. Addepalli, Fog computing and its role in the internet of things, in *MCC@SIGCOMM 2012* (2012), pp. 13–16
2. A. Sahai, B. Waters, Fuzzy identity-based encryption, in *EUROCRYPT 2005* (2005), pp. 457–473
3. V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encrypted data. CCS 2006, 89–98 (2006)
4. J. Bethencourt, A. Sahai, B. Waters, Ciphertext-policy attribute-based encryption, in *S&P 2007* (2007), pp. 321–334
5. B. Waters, Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization, in *PKC 2011, Lecture Notes in Computer Science*, vol. 6571 (Springer, Taormina, 2011), pp. 53–70
6. T. Nishide, K. Yoneyama, K. Ohta, Attribute-based encryption with partially hidden encryptor-specified access structures, in *ACNS 2008. Lecture Notes in Computer Science*, vol. 5037, ed. by S.M. Bellovin, R. Gennaro, A.D. Keromytis, M. Yung (Springer, New York, 2008), pp. 111–129
7. J. Katz, A. Sahai, B. Waters, Predicate encryption supporting disjunctions, polynomial equations, and inner products, in *EUROCRYPT 2008. Lecture Notes in Computer Science*, vol. 4965, ed. by N.P. Smart (Springer, Istanbul, 2008), pp. 146–162
8. S. Yu, C. Wang, K. Ren, W. Lou, Achieving secure, scalable, and fine-grained data access control in cloud computing, in *INFOCOM 2010* (IEEE, San Diego, USA, 2010), pp. 534–542
9. M. Green, S. Hohenberger, B. Waters, Outsourcing the decryption of ABE ciphertexts, in *USENIX 2011* (2011)
10. N. Attrapadung, Dual system encryption via doubly selective security: framework, fully secure functional encryption for regular languages, and more, in *EUROCRYPT 2014. Lecture Notes in Computer Science*, vol. 8441, ed. by P.Q. Nguyen, E. Oswald (Springer, Copenhagen, 2014), pp. 557–577
11. B. Waters, Functional encryption for regular languages, in *CRYPTO 2012. Lecture Notes in Computer Science*, vol. 7417, ed. by R. Safavi-Naini, R. Canetti (Springer, Santa Barbara, 2012), pp. 218–235
12. A.B. Lewko, B. Waters, Unbounded HIBE and attribute-based encryption, in *EUROCRYPT 2011. Lecture Notes in Computer Science*, vol. 6632, ed. by K.G. Paterson (Springer, Tallinn, 2011), pp. 547–567
13. Y. Rouselakis, B. Waters, Practical constructions and new proof methods for large universe attribute-based encryption, in *CCS 2013*, ed. by A. Sadeghi, V.D. Gligor, M. Yung (ACM, Berlin, 2013), pp. 463–474

14. N. Attrapadung, B. Libert, E. de Panafieu, Expressive key-policy attribute-based encryption with constant-size ciphertexts, in *PKC 2011. Lecture Notes in Computer Science*, vol. 6571, ed. by D. Catalano, N. Fazio, R. Gennaro, A. Nicolosi (Springer, Taormina, 2011), pp. 90–108
15. M. Chase, Multi-authority attribute based encryption, in *TCC 2007. Lecture Notes in Computer Science*, vol. 4392, ed. by S.P. Vadhan (Springer, Amsterdam, 2007), pp. 515–534
16. M. Chase, S.S.M. Chow, Improving privacy and security in multi-authority attribute-based encryption, in *CCS 2009*, ed. by E. Al-Shaer, S. Jha, A.D. Keromytis (ACM, Chicago, 2009), pp. 121–130
17. A.B. Lewko, B. Waters, Decentralizing attribute-based encryption, in *EUROCRYPT 2011. Lecture Notes in Computer Science*, vol. 6632, ed. by K.G. Paterson (Springer, Tallinn, 2011), pp. 568–588
18. A.B. Lewko, T. Okamoto, A. Sahai, K. Takashima, B. Waters, Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption, in *EUROCRYPT 2010. Lecture Notes in Computer Science*, vol. 6110, ed. by H. Gilbert (Springer, Monaco, 2010), pp. 62–91
19. T. Okamoto, K. Takashima, Fully secure unbounded inner-product and attribute-based encryption, in *ASIACRYPT 2012. Lecture Notes in Computer Science*, vol. 7658, ed. by X. Wang, K. Sako (Springer, Beijing, 2012), pp. 349–366
20. A.B. Lewko, B. Waters, New proof methods for attribute-based encryption: achieving full security through selective techniques, in *CRYPTO 2012. Lecture Notes in Computer Science*, vol. 7417, ed. by R. Safavi-Naini, R. Canetti (Springer, Santa Barbara, 2012), pp. 180–198
21. J. Li, X. Chen, J. Li, C. Jia, J. Ma, W. Lou, Fine-grained access control system based on outsourced attribute-based encryption, in *ESORICS 2013. Lecture Notes in Computer Science*, vol. 8134, ed. by J. Crampton, S. Jajodia, K. Mayes (Springer, Egham, 2013), pp. 592–609
22. C. Zuo, J. Shao, G. Wei, M. Xie, M. Ji, CCA-secure ABE with outsourced decryption for fog computing. Future Gener. Comput. Syst. **78**, 730–738 (2018)
23. P. Zhang, Z. Chen, J.K. Liu, K. Liang, H. Liu, An efficient access control scheme with outsourcing capability and attribute update for fog computing. Future Gener. Comput. Syst. **78**, 753–762 (2018)
24. J. Li, K. Ren, B. Zhu, Z. Wan, Privacy-aware attribute-based encryption with user accountability, in *ISC 2009. Lecture Notes in Computer Science*, vol. 5735, ed. by P. Samarati, M. Yung, F. Martinelli, C.A. Ardagna (Springer, Pisa, 2009), pp. 347–362
25. J. Lai, R.H. Deng, Y. Li, Fully secure cipertext-policy hiding CP-ABE, in *ISPEC 2011. Lecture Notes in Computer Science*, vol. 6672, ed. by F. Bao, J. Weng (Springer, Guangzhou, 2011), pp. 24–39
26. Y. Zhang, X. Chen, J. Li, D.S. Wong, H. Li, Anonymous attribute-based encryption supporting efficient decryption test, in *ASIACCS 2013*, ed. by K. Chen, Q. Xie, W. Qiu, N. Li, W. Tzeng (ACM, Hangzhou, 2013), pp. 511–516
27. J. Lai, R.H. Deng, Y. Li, Expressive CP-ABE with partially hidden access structures, in *ASIACCS 2012*, ed. by H.Y. Youm, Y. Won (ACM, Seoul, 2012), pp. 18–19
28. H. Cui, R.H. Deng, G. Wu, J. Lai, An efficient and expressive ciphertext-policy attribute-based encryption scheme with partially hidden access structures, in *ProvSec 2016. Lecture Notes in Computer Science*, vol. 10005, ed. by L. Chen, J. Han (Springer, New York, 2016), pp. 19–38
29. Y. Michalevsky, M. Joye, Decentralized policy-hiding ABE with receiver privacy, in *ESORICS 2018. Lecture Notes in Computer Science*, vol. 11099, ed. by J. López, J. Zhou, M. Soriano (Springer, Barcelona, 2018), pp. 548–567
30. F. Khan, H. Li, L. Zhang, J. Shen, An expressive hidden access policy CP-ABE, in *DSC 2017* (IEEE Computer Society, Shenzhen, China, 2017), pp. 178–186
31. D.X. Song, D.A. Wagner, A. Perrig, Practical techniques for searches on encrypted data, in *2000 IEEE Symposium on Security and Privacy, Berkeley, California, USA, May 14–17 2000* (2000), pp. 44–55
32. D. Boneh, G.D. Crescenzo, R. Ostrovsky, G. Persiano, Public key encryption with keyword search, in *Proceedings of the Advances in Cryptology—EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2–6, 2004* (2004), pp. 506–522
33. Q. Zheng, S. Xu, G. Ateniese, VABKS: verifiable attribute-based keyword search over outsourced encrypted data, in *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27–May 2, 2014* (2014), pp. 522–530
34. W. Sun, S. Yu, W. Lou, Y.T. Hou, H. Li, Protecting your right: verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud. IEEE Trans. Parallel Distrib. Syst. **27**(4), 1187–1198 (2016)
35. Y. Miao, J. Ma, X. Liu, J. Weng, H. Li, H. Li, Lightweight fine-grained search over encrypted data in fog computing. IEEE Trans. Serv. Comput. **12**(5), 772–785 (2019)
36. F. Meng, M.W.L. Cheng, ABDKS: attribute-based encryption with dynamic keyword search in fog computing. Front. Comput. Sci. (2020). https://doi.org/10.1007/s11704-020-9472-7

Publisher's Note