He *et al. Hum. Cent. Comput. Inf. Sci.* (2020) 10:49
https://doi.org/10.1186/s13673-020-00255-5

Human-centric Computing and Information Sciences

RESEARCH Open Access

An efficient attribute-based hierarchical data access control scheme in cloud computing

Heng He[1,2*], Liang-han Zheng[1,2], Peng Li[1,2], Li Deng[1,2], Li Huang[1,2] and Xiang Chen[1,2]

*Correspondence:
heheng@wust.edu.cn
[1] School of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan, China
Full list of author information is available at the end of the article

Abstract

Security issues in cloud computing have become a hot topic in academia and industry, and CP-ABE is an effective solution for managing and protecting data. When data is shared in cloud computing, they usually have multiple access structures that have hierarchical relationships. However, existing CP-ABE algorithms do not consider such relationships and just require data owners to generate multiple ciphertexts to meet the hierarchical access requirement, which would incur substantial computation overheads. To achieve fine-grained access control of multiple hierarchical files effectively, first we propose an efficient hierarchical CP-ABE algorithm whose access structure is linear secret sharing scheme. Moreover, we construct an attribute-based hierarchical access control scheme, namely AHAC. In our scheme, when a data visitor's attributes match a part of the access control structure, he can decrypt the data that associate with this part. The experiments show that AHAC has good security and high performance. Furthermore, when the quantity of encrypted data files increases, the superiority of AHAC will be more significant.

Keywords: Cloud computing, Attribute-based encryption, Hierarchical access structure, Linear secret sharing

Introduction

The advent of the mobile Internet era has brought data sharing into people's daily life, and the relevant platforms are also widely used, like Facebook, Badoo and MySpace. In the meantime, cloud computing has become a promising technology used for massive data sharing [1]. Before sharing data, users usually choose to encrypt data to protect their security. One traditional method is to use symmetric encryption, the other is to use public key encryption [2–4]. Nevertheless, there are some problems with these methods. Some of them cannot achieve flexible access control [2], some schemes are poor in performance [3], and some have defects in security [4]. Therefore, attribute-based encryption (ABE) [5] was proposed to overcome these problems in unreliable storage environment. The access control strategy of ciphertext-policy ABE (CP-ABE) is encrypted into the ciphertext [6]. This feature makes it very suitable for data sharing. In CP-ABE, the time of plaintext encryption is only linearly proportional to the attribute number, so it's efficient. Shamir secret sharing scheme [6] is the foundation of traditional

CP-ABE algorithm. In 2011, a more efficient algorithm based on linear secret sharing scheme (LSSS) was proposed [7].

In actual application scenarios, shared data files usually have multiple access structures that have hierarchical relationships. This relationship is common in the health and military fields. Most CP-ABE do not consider this hierarchical relationship and just require data owners to generate multiple ciphertexts to encrypt these files, which would incur substantial computation overheads. Wang proposed a File Hierarchy ABE scheme (FH-CP-ABE) [8], which integrates multiple different access structures with hierarchical relationships into a single access structure. When a data visitor's attributes match the partial access structure, he can decrypt the data that associate with this part. Only when the entire access structure is satisfied, all the data can be decrypted. Since the data owner does not need to generate multiple access structures and ciphertexts, the efficiency is greatly improved. However, FH-CP-ABE uses a tree access structure, so its efficiency is still low.

In this article, our contributions are as follows:

(1) We design a hierarchical CP-ABE algorithm whose access structure is LSSS matrix. In the algorithm, multiple hierarchical access control structures of data files are integrated into a single LSSS matrix, so all the data are encrypted into an entire ciphertext.
(2) Based on the proposed CP-ABE algorithm, we construct an Attribute-based Hierarchical data Access Control scheme (AHAC) in the cloud computing. In AHAC, we achieve efficient and flexible access control. When a data visitor's attributes match a part of the access control structure, he can decrypt the data that associate with this part. Moreover, the scheme just requires one operation of encryption and decryption to complete the work that traditional schemes have to do multiple encryption and decryption.
(3) We conduct security analysis and performance evaluation for AHAC. Security analysis shows that AHAC has prominent security features. Performance evaluation demonstrates that the private key production time and storage cost of our scheme are only 25 percent of FH-CP-ABE, and the encryption and decryption time and ciphertext storage cost also have advantages.

The remaining parts of this paper are organized as follows. In "Related work" section, we introduce some related work in this field. In "Preliminaries" section, we introduce preliminaries which contain some notions and definitions. Then, the detailed construction of AHAC is presented in "AHAC: attribute-based hierarchy data access control scheme" section. In "Security analysis and performance evaluation" section, we provide security analysis and performance evaluation. Finally, the conclusions are given in "Conclusions" section.

Related work

The fuzzy identity-based encryption [5] put forward by Sahai and Waters in 2005 is the prototype of ABE. The basic ABE can only represent the "threshold" operation of attributes, and the threshold parameters are set by the authorized authority rather than by

the sender. Cheung realized the first CP-ABE scheme, which can just support "AND" gate access control strategy [9]. To implement a more flexible strategy, a new CP-ABE was designed by Bethencourt. His scheme applies the tree access control structure to realize the "AND", "OR", and "OF" strategy, and achieves fine-grained access control [6]. However, it cannot provide strong security. In 2008, Goyal and Jain put forward a CP-ABE that has selectively security in the decisional-Bilinear Diffie-Hellman (d-BDH) assumption [10]. Nevertheless, the time consumption by encryption and decryption, and the sizes of its private key and ciphertext grow up by $n^{3.42}$ (n represents the attribute number associated with the access control tree), which limits its practicability. In 2011, Lewko and Waters proposed a technology which can transform an access control tree to an LSSS representation. This technique makes it possible to replace tree structure with matrix structure. Thus, in the same year, Waters designed a CP-ABE scheme using matrix structure [7]. Its time consumption by encryption and decryption, and the sizes of private key and ciphertext increase linearly with its attribute number. Besides, the scheme has selectively security in the decisional q-parallel BDH Exponent (d-Parallel BDHE) assumption [7]. Some schemes [11–15] have applied CP-ABE to realize file access control in the cloud. There are also some schemes to improve the algorithm itself, such as [16, 17] fix the ciphertext size to improve performance, and [18, 19] improve security through authority control or accountability, and [20–22] support attribute revocation to improve practicability. Scheme [23] supports proxy computing to private servers, and [24] supports hidden access policy, and [25] proposes a lightweight and efficient CP-ABE. However, none of them consider the hierarchical access relationships of multiple shared files.

Researchers also proposed some hierarchical CP-ABE based on tree or LSSS matrix structure. The schemes proposed in [26–28] use multiple hierarchical authorized organizations to create secret keys cooperatively for users, and alleviate the burden of a single authority center. In [29–31], schemes without central authority were further proposed, which improved the system security. In [32], there is a hierarchical relationship between attributes, and attributes with high permission can replace the attributes with low permission when decrypting. In [8], FH-CP-ABE is proposed for cloud data access control, and an integrated tree access structure is used for encrypting all the data. However, its efficiency is still not high. It should be noticed that in our scheme, we focus on the issue of hierarchical access relationships of multiple shared files, which is the same as [8].

Preliminaries

First of all, we present the related preliminaries of AHAC, then we describe an example of using these techniques to implement hierarchical access control, and last we give the definition of d-Parallel BDHE.

Hierarchical sccess control

In the traditional CP-ABE scheme, users' attributes either satisfy the access control structure to obtain plaintext, or do not satisfy the access control structure to obtain plaintext. As shown in Fig. 1, only user 1 and 4 can recover the plaintext, because their attributes match the access control structure.

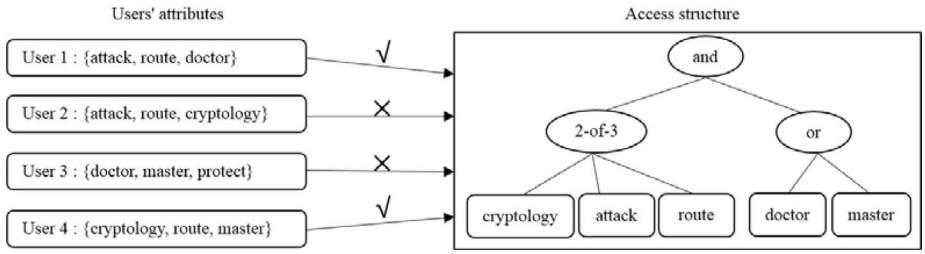


Fig. 1 An instance of the data access process in CP-ABE

In hierarchical access control, multiple different access structures with hierarchical relationships can be integrated into a single access structure. As shown in Fig. 2, T_1, T_2 represents the access structures of m_1, m_2 accordingly, and obviously they have hierarchical relationship, so they can be integrated into a single access structure T. As shown in Fig. 3 when a data visitor's attributes match the partial access structure, he can decrypt the data that associate with this part (User 2). Only when the entire access structure is satisfied, all the data can be decrypted (User 1). Since the data owner does not need to generate multiple access structures and ciphertexts, the efficiency is greatly improved.

Linear secret sharing scheme

Beimel first proposed the definition of LSSS in paper [33]: A secret sharing scheme Π over a collection of parties P is described linear on Z_p when:

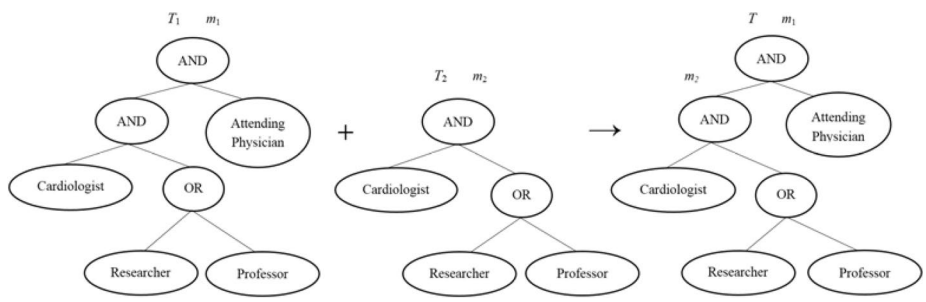


Fig. 2 An instance of the integrated access control structure

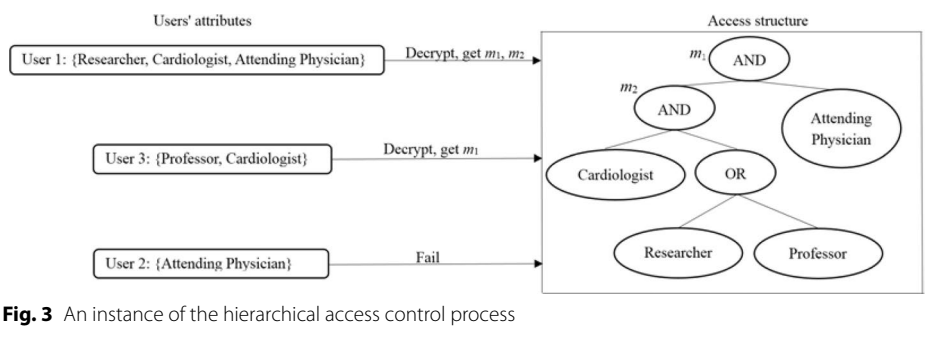


Fig. 3 An instance of the hierarchical access control process

(1) The shares of all the parties make up a vector on Z_p.
(2) Such a matrix M for Π is existed, which is used for producing shares. M has l rows and n columns. For $i = 1, 2, \ldots, l$, the ith row M_i of M is marked by a party $\rho(i)$ where function ρ satisfies: $\{1, 2, \ldots, l\} \rightarrow e$. Given a column vector $\vec{v} = (s, r_2, \ldots, r_n)$, in which $s \in Z_p$ is the shared secret and $r_2, \ldots r_d \in Z_p$ are randomly chosen, $M\vec{v}$ is the vector constructed by m shares of s decided by Π. The share $\lambda_i = (M\vec{v})_i$ is part of party $\rho(i)$.

It is shown in [33] that each LSSS has the linear reconstruction feature: Assume that there exists an LSSS Π corresponding to the access structure T, and $S \in T$ is an arbitrary authorized set, $I \subset \{1, \ldots, l\}$ is denoted as $I = \{i : \rho(i) \in S\}$. There are constants $\{\omega_i \in Z_p\}_{i \in I}$ that makes $\sum_{i \in I} \omega_i \lambda_i = s$, in which $\{\lambda_i\}$ are shares of arbitrary secret s decided by Π. In addition, $\{\omega_i\}$ will be found under polynomial time in the size of the share-generating matrix M.

There will exist a vector like that $\omega \cdot (1, 0, \ldots, 0) = -1$ and $\omega \cdot M_i = 0$ for all $i \in I$ for any unauthorized set of rows I.

It can be obtained by mathematical derivation for a randomly selected vector $\vec{v} = (s_1, \ldots, s_j, \ldots, s_n)$, where $s_j \in Z_p$ is the jth secret of the n secrets that need to be recovered, and it corresponds to a non-leaf node in the tree structure. When recovering a secret, if the set of attributes possessed can satisfy this non-leaf node, then $\{\omega_i \in Z_p\}_{i \in I}$ will be found under the polynomial time which satisfies $\sum_{i \in I} \omega_{i,j} M_i^T = \varepsilon_j$, where ε_j is a row vector whose length is n with the jth element is 1 and the remaining elements are 0. Then we can get $s_j = \sum_{i \in I} \omega_{i,j} \lambda_i$.

Marking method to construct LSSS matrix

Beimel proved that the access control strategy described by tree structure can be converted to matrix M in LSSS, but no specific conversion method is given in [33]. Until 2011, Lewko and Waters presented a construction method for an LSSS matrix in [34]: Given an access tree defined by a Boolean formula, it can be converted to an LSSS matrix by a marking method. And any one of the propositional paradigms can find its Boolean formula. The specific conversion method can be found in [33].

An example of hierarchical access control using LSSS matrix

There is a hierarchical access tree T which is shown in Fig. 2, and its Boolean formula is (A AND (B AND (C OR D))). We can use the above marking method to convert it to an LSSS matrix by Formula 1 as:

$$M = \begin{pmatrix} 1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & -1 \end{pmatrix} \tag{1}$$

Next, we give an example of how to use the LSSS matrix to achieve hierarchical access control.

When encrypting, we randomly select a vector $\vec{v} = (s_1, s_2, s_3) = (2, 5, 3)$, in which s_1, s_2, s_3 are secrets assigned to the non-leaf nodes in Fig. 2. Then we can calculate λ by Formula 2:

$$\lambda = M \cdot \vec{v} = \begin{pmatrix} 1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \\ 0 & 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} 2 \\ 5 \\ 3 \end{pmatrix} = \begin{pmatrix} 7 \\ -2 \\ -3 \\ -3 \end{pmatrix} \tag{2}$$

From "Linear secret sharing scheme", we know $s_j = \sum_{i \in I} \omega_{i,j} \lambda_i$, where $I = \{i : \rho(i) \in S\}$,$\rho(i)$ can convert the ith row into the attribute represented by this row, and S is the user's attribute set. Thus, we can get Formula 3:

$$s_j = \omega_j^T \lambda_A \quad \text{where } \lambda_A = \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_i \\ \vdots \\ \lambda_l \end{pmatrix}_{i \in I} \tag{3}$$

Obviously, we must get ω_j if we want to get s_j, then we make the following formula 4 derivation:

$$s_j = s_j^T = \lambda_A^T \omega_j = (M_A \cdot \vec{v}^T)^T \omega_j = \vec{v} \cdot (M_A^T \omega_j) \quad \text{where } M_A = \begin{pmatrix} M_1 \\ \vdots \\ M_i \\ \vdots \\ M_l \end{pmatrix}_{i \in I} \tag{4}$$

We make $M_A^T \omega_j = \varepsilon_j$, so $s_j = \vec{v} \cdot \varepsilon_j$. Then we can compute ε_j as a row vector whose length is n with the jth element is 1 and the remaining elements are 0.

When decrypting, if a decryptor only has the attributes B, C, i.e., it only satisfies the partial access structure, then he can get ω_2, ω_3 by Formula 5 and 6:

$$M_A^T \omega_3 = \begin{pmatrix} 0 & 0 \\ -1 & 0 \\ 1 & -1 \end{pmatrix} \cdot \omega_3 = \varepsilon_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \tag{5}$$

$$M_A^T \omega_2 = \begin{pmatrix} 0 & 0 \\ -1 & 0 \\ 1 & -1 \end{pmatrix} \cdot \omega_2 = \varepsilon_2 = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \tag{6}$$

Thus, $\omega_3 = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$, $\omega_2 = \begin{pmatrix} -1 \\ -1 \end{pmatrix}$. Finally, he can get s_3 and s_2 from Formulas 7 and 8:

$$s_3 = \omega_3^T \lambda_A = \begin{pmatrix} 0 & -1 \end{pmatrix} \cdot \begin{pmatrix} -2 \\ -3 \end{pmatrix} = 3 \tag{7}$$

$$s_2 = \omega_2^T \lambda_A = \begin{pmatrix} -1 & -1 \end{pmatrix} \cdot \begin{pmatrix} -2 \\ -3 \end{pmatrix} = 5 \tag{8}$$

Similarly, if the decryptor has the attributes A, B, and C, then he satisfies the entire access structure, and all the secrets s_1, s_2, s_3 can be computed by the above steps.

AHAC: attribute-based hierarchy data access control scheme

In the chapter, first we give the overview and the security assumptions of AHAC. After that, we design the core algorithm of AHAC, namely AHAC-CP-ABE. Finally, we present the system operations of AHAC detailedly.

Scheme overview

The system framework of AHAC is shown in Fig. 4. Firstly, central authority (CA) performs the system initialization operation and generates system attributes and relevant keys. Then, double encryption mechanism are used to promote the efficiency, that is, data owner chooses n symmetric keys $\{ck_1, \ldots, ck_n\}$ to encrypt the data files $\{f_1, \ldots, f_n\}$ respectively using a symmetric encryption algorithm (AES, DES, etc.), and encrypts $\{ck_1, \ldots, ck_n\}$ using AHAC-CP-ABE algorithm. The symmetric encryption algorithm with high efficiency is used to encrypt the files of large volume, and the CP-ABE algorithm is used to encrypt the symmetric key of small volume. Compared with the symmetric encryption algorithm, the performance of CP-ABE algorithm is relatively lower. However, the CP-ABE algorithm can bring the obvious advantage in key management, using which we can easily implement the access control of encrypted data. Thus, we utilize such double encryption method to achieve the secure, efficient and fine-grained data access control in the cloud.

The user then transfers the two ciphertexts to cloud server (CS) and CS stores them for sharing. When a data visitor wishes to obtain the data files, he should contact CA and CA distributes corresponding private keys to him according to his attributes. Then, this data visitor obtains the ciphertexts from CS. When his attributes match partial or entire access control structure, he can decrypt the symmetric keys that associated with this part. At last, the data visitor is able to get the corresponding files using the symmetric

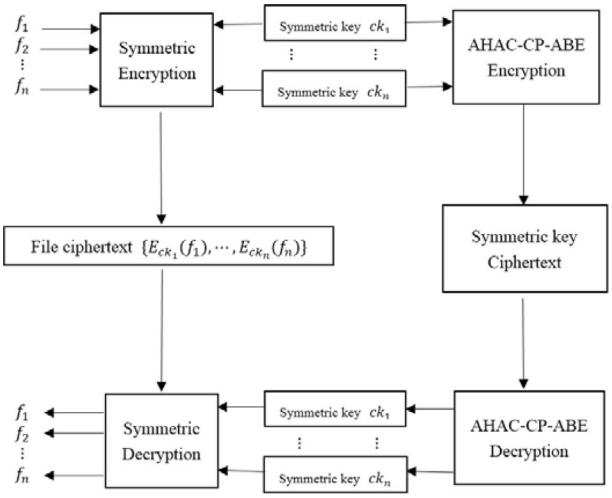


Fig. 4 The system framework of AHAC

keys. It is clear from our framework that only one encryption and decryption operation is needed to share multiple files securely, while traditional schemes have to do multiple encryption and decryption operations.

Security assumptions

In this section, we will present security assumptions for several entities in the system.

We consider that CS is honest but curious in AHAC like the related work [35] do, that is, CS will honestly perform the task of private key distribution yet it is also trying to gain the contents of the data files and symmetric keys stored in it. Besides, CS is online all the time to provide stable services.

CA is fully trusted and is online all the time. There is a security approach for CA to transfer private key to users. Users can get the services of the system at any time.

For any number of unauthorized users, they may launch collusion attacks and try to obtain the confidential data.

AHAC-CP-ABE

The AHAC-CP-ABE includes four functions: system initialization, private key production, encryption and decryption. These functions make the following cases: when a data visitor's attributes match a part of the access control structure, he can decrypt the data that associate with this part, and when the entire access structure is satisfied, all the data can be decrypted. Here are the details of the algorithm:

(1) System initialization

Function 1 takes an attribute set U of system and a parameter k specifying the system security as input, and produces a system master key MK and a corresponding public key PK.

Function 1. System initialization

INPUT: System attribute set U, security parameter k.
OUTPUT: System master key MK, system public key PK.
1. Select a bilinear group G_0 with prime order p, of which g is generator, and a bilinear map $e: G_0 \times G_0 \rightarrow G_1$.
2. Choose elements $h_1, \cdots, h_U \in G_0$ at random, which are related to each attribute in U.
3. Choose random elements $\alpha, \beta \in Z_p$.
4. Calculate $e(g,g)^{\alpha}, g^{\alpha}, g^{\beta}$.
5. Return $PK = (g, e(g,g)^{\alpha}, g^{\beta}, h_1, \cdots, h_U), MK = g^{\alpha}$.

(2) Private key production

As shown in Function 2, it inputs PK, MK, and the attribute set S of a user, and produces a user private key SK that is related to S.

Function 2. Private key production

INPUT: System public key *PK*, system master key *MK*, attribute set *S*.
OUTPUT: Private key *SK*.
1. Choose an element $t \in Z_p$ at random.
2. Calculate $K = g^{\alpha} g^{\beta t}, L = g^{t}, \forall x \in S : K_x = h_x^t$.
3. Return $SK = (K, L, \forall x \in S : K_x)$.

(3) Encryption

As shown in Function 3, the encryption function inputs a plaintext set $\{m_j, j \in (1, n)\}$, *PK*, and an LSSS matrix structure (M, ρ), and returns a ciphertext *CT*. For an LSSS matrix structure (M, ρ), the dimension of M is $l \times n$, M_i is the ith row of M, and $\rho(i)$ can convert M_i into the attribute represented by it.

Function 3. Encryption

INPUT: Plaintext set $\{m_j, j \in (1,n)\}$, system public key *PK*, LSSS matrix structure (M, ρ).
OUTPUT: Ciphertext *CT*.
1. Choose a random vector $\vec{v} = (s_1, s_2, \cdots, s_n) \in Z_p^n$.
2. Choose random elements $r_1, \cdots, r_l \in Z_p$.
3. Calculate $C_j^* = m_j e(g,g)^{\alpha s_j}, C_j^{'} = g^{s_j}, j \in (1,n)$.
4. Calculate $\lambda_i = M_i \vec{v}, i \in (1,l)$.
5. Calculate $C_i = g^{\beta \lambda_i} h_{\rho(i)}^{-r_i}, \ D_i = g^{r_i}, i \in (1,l)$.
6. Calculate $CT = ((M, \rho), C_j^*, C_j^{'}, C_i, D_i, j \in (1,n), i \in (1,l))$.

(4) Decryption

As shown in Function 4, *CT* and *SK* are inputs, and outputs is plaintext set m_j. M_A is a matrix composed of a set of row vectors in M that corresponds to the attribute set S associated with *SK*. ε_j is a row vector with length n, in which the jth element is 1 and the remaining elements are 0. $I = \{i : \rho(i) \in S\}$.

Function 4. Decryption

INPUT: Ciphertext CT, private key SK.
OUTPUT: Plaintext set $\{m_j, j \in (1,n)\}$.

1. For every $j \in (1,n)$:Calculate $\omega_{i,j}$ according to $M_A^T \omega_j = \varepsilon_j$, i.e., $\sum_{i \in I} \omega_{i,j} M_i^T = \varepsilon_j$.

If $\omega_{i,j}$ can be obtained successfully, calculate

$$\begin{aligned} F_j &= \frac{e(C_j^{'}, K)}{\prod_{i \in I,j} \left(e(C_i, L) e(D_i, K_{\rho(i)})\right)^{\omega_{i,j}}} \\ &= \frac{e(g^{sj}, g^{\alpha} g^{\beta t})}{\prod_{i \in I,j} \left(e(g^{\beta \lambda_i}, g^{t}) e(g^{\beta \lambda_i}, h_{\rho(i)}^{-r_i}) e(g^{r_i}, h_{\rho(i)}^{t})\right)^{\omega_{i,j}}} \\ &= \frac{e(g,g)^{\alpha s_j} e(g,g)^{\beta s_j t}}{\prod_{i \in I,j} e(g,g)^{t \beta \lambda_i \omega_{i,j}}} \\ &= e(g,g)^{\alpha s_j} \end{aligned}$$

Node that $\sum_{i \in I,j} \lambda_i \omega_{i,j} = s_j$. This formula has been explained in Section 3.2.

Calculate $m_j = C_j^* / F_j$.

If $\omega_{i,j}$ cannot be obtained, m_j cannot be decrypted, i.e., $m_j = NULL$.

2. Return $\{m_j, j \in (1,n)\}$.

The detailed operation process of AHAC

AHAC consists of six operations: System initialization, encryption of data files, encryption of symmetric keys, user authorization, decryption of symmetric keys and decryption of data files.

(1) System setup

CA designates an attribute set U and invokes Function 1 to produce a master key MK and a public key PK, and MK is safely stored in CA.

(2) Encryption of data files

Data owner (DO) chooses n symmetric keys $\{ck_1, \ldots, ck_n\}$ to encrypt his data files $\{f_1, \ldots, f_n\}$ by a symmetric encryption algorithm respectively. The data file ciphertext are denoted as: $EF = \{E_{ck_1}(f_1), \ldots, E_{ck_n}(f_n)\}$.

(3) Encryption of symmetric keys

DO defines access trees $\{T_1, \ldots, T_n\}$ for his data files $\{f_1, \ldots, f_n\}$ respectively and integrates them into a single access tree T. Then, he uses marking method to converted T to LSSS matrix structure (M, ρ). Next, he calls Function 3 to encrypt his symmetric keys $\{ck_1, \cdots, ck_n\}$ and generates a symmetric key ciphertext CT. Finally, he sends CT and EF to CS and CS stores them.

(4) User authorization

For any data visitor, CA specifies a set S of attributes and calls Function 2 to output the corresponding private key SK.

(5) Decryption of symmetric keys

When a user wants to obtain some files from CS, CS first checks whether his attributes match partial or entire access control structure of those data files. If not, CS refuses the user's request; otherwise, CS sends CT to the user. After obtaining CT, the user calls Function 4 to get the symmetric keys. When his attributes satisfy a part of the access tree, he can decrypt the symmetric keys that associated with this part, assuming $\{ck_1, \ldots, ck_n\}$. Only when his attributes match the entire access control structure, he can obtain all the symmetric keys.

(6) Decryption of data files

In the last step, the user downloads $\{E_{ck_1}(f_1), \ldots, E_{ck_n}(f_n)\}$ and uses $\{ck_1, \ldots, ck_n\}$ to decrypt the data files $\{f_1, \ldots, f_n\}$ by the symmetric decryption algorithm.

To further improve the efficiency, we make the following transformation:

$$\begin{aligned} ck_n^{'} &= ck_n, \\ ck_{n-1}^{'} &= ck_{n-1} \cup ck_n^{'}, \\ ck_1^{'} &= ck_1 \cup ck_2^{'} \end{aligned}$$

where $\{ck_1, \ldots, ck_n\}$ are n symmetric keys. After then, we call Function 3 to encrypt $\{ck_1^{'}, \ldots, ck_n^{'}\}$ and generates a symmetric key ciphertext CT. When decrypting, we call Function 4 to get the symmetric keys. In Function 4, once we successfully decrypt a $ck_j^{'}$, we can stop the decryption process immediately, since $ck_j^{'}$ contains all the contents of the rest symmetric keys.

Security analysis and performance evaluation

In this chapter, we give the analysis for the security and the evaluation results for the performance.

Security analysis

We give the security features of AHAC based on the security assumptions presented in chapter 4.2, containing data confidentiality, collusion defense and fine-grained access control.

(1) Data confidentiality

AHAC-CP-ABE algorithm is designed on top of Waters's algorithm [7]. The security of his scheme is based on d-Parallel BDHE assumption.

d-Parallel BDHE assumption: Select a bilinear group G of prime order p with generator g, and select $\beta, s, b_1, \ldots, b_q \in Z_p$ at random. Even if the adversary gets

$$\vec{y} = \left\{ \begin{array}{l} g, g^s, g^\beta, \ldots, g^{(\beta^q)}, \ , g^{(\beta^{q+2})}, \ldots g^{(\beta^{2q})} \\ \forall_{1 \le j \le q} \ g^{s \cdot b_j}, g^{\beta/b_j}, \ldots g^{(\beta^q/b_j)}, \ , g^{(\beta^{q+2}/b_j)}, \ldots g^{(\beta^{2q}/b_j)} \\ \forall_{1 \le j,k \le q, k \ne j} g^{\beta s b_k/b_j}, \ldots, g^{\beta^q s b_k/b_j} \end{array} \right\}$$

it's hard for him to get $e(g,g)^{\beta^{q+1}s} \in G_T$.

There exists a main difference between AHAC-CP-ABE algorithm and his algorithm. In AHAC-CP-ABE, we use all the elements in the secret vector $\vec{v}$ to allow multiple secrets to be carried in an access control policy, under which multiple plaintexts are encrypted. That is to say, AHAC-CP-ABE exploits all the elements in vector $\vec{v}$, using each of them to encrypt every plaintext respectively, as shown in Function 3, whereas in Waters's CP-ABE algorithm, just one element in the vector is used for encrypting a plaintext [7] and for multiple plaintexts, their algorithm needs to be executed multiple times. In [7], Waters's CP-ABE algorithm has the selectively security in d-Parallel BDHE assumption. Therefore, AHAC-CP-ABE has the same security under the same assumption.

In AHAC, data files are encrypted using symmetric encryption keys, and these keys are then encrypted using AHAC-CP-ABE. In this mechanism, just the ciphertexts of the files and the ciphertexts of the keys are given to cloud servers. Since the used symmetric encryption algorithm, such as AES, is secure, the security of this mechanism merely relies on the security of AHAC-CP-ABE. In the above paragraph, we have shown that AHAC-CP-ABE is secure under d-Parallel BDHE assumption. Thus, the AHAC is secure under the same model.

(2) Collusion defense

Any number of unauthorized users may launch collusion attacks, trying to access the confidential data files. In AHAC-CP-ABE, CA chooses an element t randomly for each user and uses t to generate a private key for each of them. When a user decrypts a ciphertext, he should compute $e(g,g)^{\alpha s_j}$ first, which requires the components of his private key contain the same t. That is to say, different data visitors can't integrate their private keys to strengthen their decryption power, since they have different values of t in private keys. Therefore, AHAC can resist collusion attacks effectively.

(3) Fine-grained access control

In AHAC, the LSSS matrix access structure is transformed from an access tree which supports "AND" "OR", and "OF" threshold operations, and it can represent any complex access control policy. Only data visitors who own the attributes matching the access control structure can obtain the plaintext successfully. Thus, AHAC realizes fine-grained access control.

Performance evaluation

We evaluate the performance of AHAC-CP-ABE from two aspects: its time costs, and the storage costs of ciphertext and private key. Both are compared with those of traditional CP-ABE [6], LSSS-based CP-ABE (hereinafter referred to as LS-CP-ABE) [7], and FH-CP-ABE [8].

We make the following access policy: assume that the plaintext $M = (m_1, m_2, \ldots, m_n)$, for the traditional CP-ABE and LS-CP-ABE, n policies are needed respectively for $m_1, m_2, \ldots, m_n$ as:

Policy(1): $\{(att_1, att_2, \ldots, att_i, j\ of\ i)\ AND\ att_{i+1}\ AND\ att_{i+2}\ AND \cdots AND\ att_{i+n-1}\}$

Policy(2): $\{(att_1, att_2, \ldots, att_i, j\ of\ i)\ AND\ att_{i+1}\ AND\ att_{i+2}\ AND \cdots AND\ att_{i+n-2}\}$

.....

Policy($n-1$): $\{(att_1, att_2, \ldots, att_i, j\ of\ i)\ AND\ att_{i+1}\}$

Policy(n): $\{att_1, att_2, \ldots, att_i, j\ of\ i\}$

FH-CP-ABE and AHAC-CP-ABE only need one access policy with n access structure level as:

Policy: $\{(att_1, att_2, \ldots, att_i, j\ of\ i)\ AND\ att_{i+1}\ AND\ att_{i+2}\ AND \ldots AND\ att_{i+n-1}\}$

In Table 1, we compare the performance of four CP-ABE algorithms by theoretical calculation. μ represents the global attribute set, $\omega \in \mu$ represents the attribute information contained in the user's private key, c represents the attribute contained in the access structure, n represents the access structure hierarchy, the power operation on the group G_0 is E_0, the power operation on the group G_T is E_T, and the multiplication calculation on the group is M. P represents the pairing operation in group G_0. The element size on group G_0 is represented as l_0, and the element size on group G_T is represented as l_T. Due to the trivial time consumption of hash operation, the time consumption of hash is ignored. As shown in Table 1, AHAC-CP-ABE has high performance in all aspects.

We conduct detailed experiments to simulate the complete access control process, in which all four algorithms are implemented based on JPBC [36]. In the experiments, a super singular elliptic curve $y^2 = x^3 + x$ is adopted of which the group order is 160 bits on a 512-bit finite field. The experiments are performed on a computer with Pentium G4560 3.50 Hz processor, and 8.00 GB RAM. We take the average of 10 experiments as results to make them more accurate.

Table 1 Compare of the performance of four algorithms

Scheme	CP-ABE [6]	FH-CP-ABE [8]	LS-CP-ABE [7]	AHAC-CP-ABE
Setup time	$2E_0 + E_T + P$	$2E_0 + E_T + P$	$2E_0 + E_T + P$	$2E_0 + E_T + P$
Private key generation time	$(2+2\omega)E_0 + (\omega+1)M$	$(2+2\omega)E_0 + (\omega+1)M$	$(2+\omega)E_0 + M$	$(2+\omega)E_0 + M$
Encryption time	$(2c+1)nE_0 + nE_T + nM$	$(2c+n)\cdot(E_0 + E_T + M)$	$(3c+1)nE_0 + nE_T + (c+1)nM$	$(3c+n)E_0 + nE_T + (n+c)M$
Decryption time	$cnE_T + nM + (2c+1)nP$	$cE_T + nM + (2c+1)P$	$cnE_T + nM + (2c+1)nP$	$cE_T + M + (2c+1)P$
Private key storage	$(2\omega+1)l_0$	$(2\omega+1)l_0$	$(2+\omega)l_0$	$(2+\omega)l_0$
Ciphertext storage	$(2c+1)nl_0 + nl_T$	$(2c+n)l_0 + (n+c)l_T$	$(2c+1)nl_0 + nl_T$	$(2c+n)l_0 + nl_T$

The private key generation time of four algorithms have been shown in Fig. 5. As the attribute number increases, the private key production time costs and the private key storage costs of AHAC-CP-ABE and LS-CP-ABE grow slower than those of the other two algorithms. This will significantly reduce the pressure of CA.

Figure 6 shows the encryption and decryption time costs with two fixed access structure levels as attributes increase. We can see that the time costs by encryption and decryption of AHAC-CP-ABE and FH-CP-ABE are always less than those of the other two algorithms.

Figure 7 shows the encryption and decryption time costs with different access structure level and fixed attribute number $N=30$ respectively. It's obvious that the encryption and decryption time costs of FH-CP-ABE and AHAC-CP-ABE are constants when the number of access structure levels increases, while in traditional CP-ABE and LS-CP-ABE there are rapid linear growth in the time costs.

From Figs. 5, 6 and 7, we can conclude that the time consumptions by encryption and decryption of AHAC-CP-ABE are still less than those of FH-CP-ABE. However, in the cloud environment with big data, the gap of them will be widened. Moreover, the private key production time consumption by private key production of AHAC-CP-ABE is much less than that of FH-CP-ABE.

Figure 8 shows the storage cost of private key. As the attribute number increases, the private key storage costs of AHAC-CP-ABE and LS-CP-ABE grow slower than those of the other two algorithms.

Figure 9a shows the storage cost of ciphertext with two fixed access structure levels as attributes increase. We can see that the ciphertext storage costs of FH-CP-ABE and AHAC-CP-ABE are very close, while the costs of traditional CP-ABE and LS-CP-ABE are about twice as those of them, since in this experiment, the access structure level is set to two. Figure 9b shows the storage cost of ciphertext with different access structure level and fixed attribute number $N=30$ respectively. We can see that the ciphertext storage costs of AHAC-CP-ABE and FH-CP-ABE increase slightly when

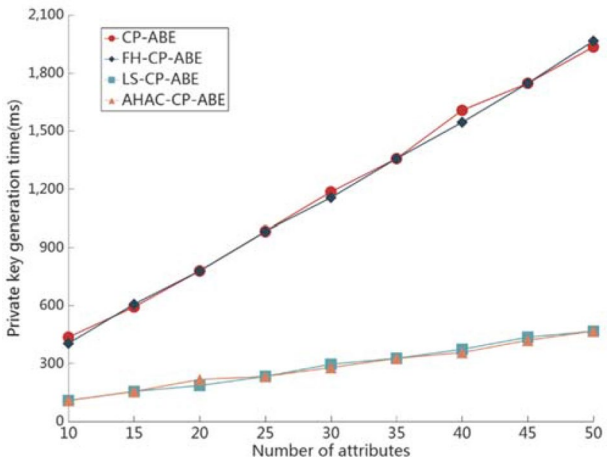
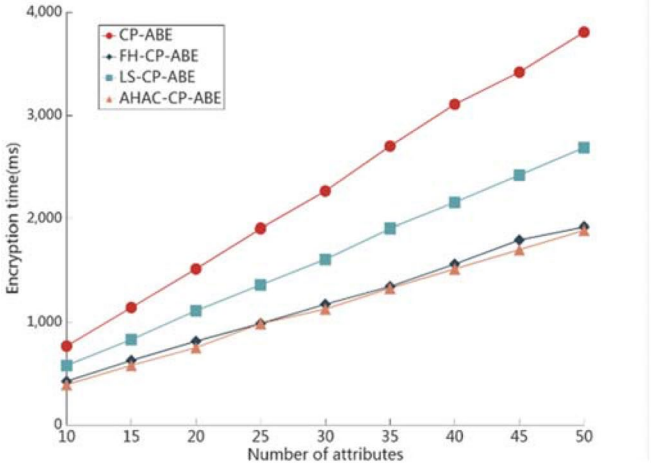
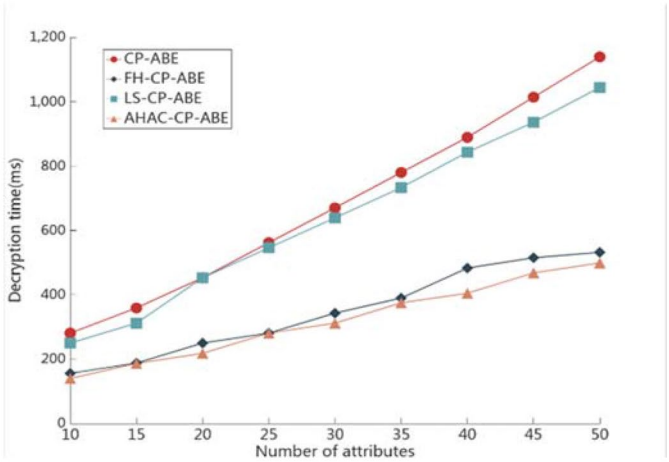


Fig. 5 Private key generation time



a Encryption time when the attribute number increases



b Decryption time when the attribute number increases

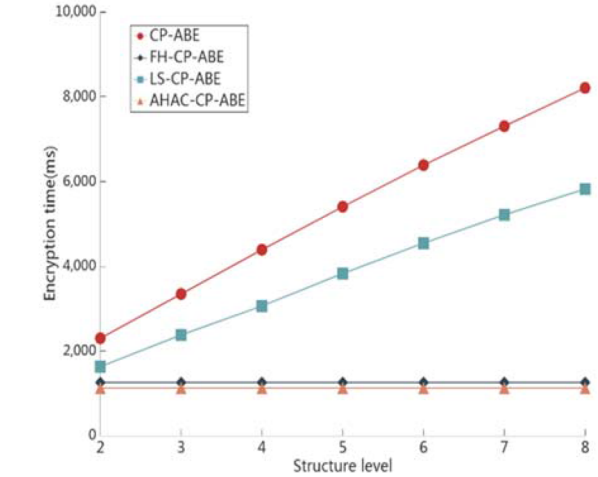
Fig. 6 Encryption and decryption time when the attribute number increases

the number of access structure level increases, and the ciphertext storage costs of traditional CP-ABE and LS-CP-ABE increase sharply.

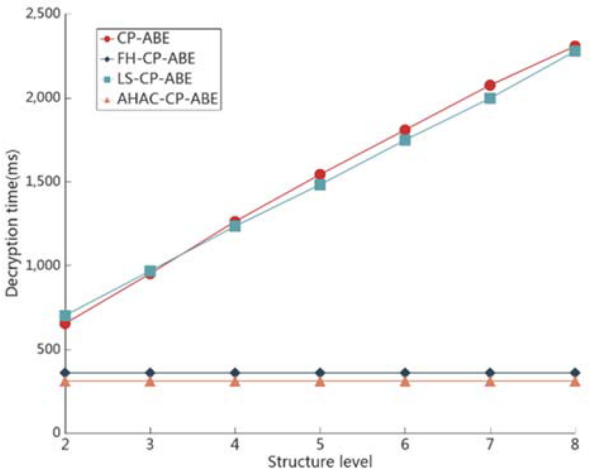
From Figs. 8 and 9, we can conclude that the ciphertext storage consumption of AHAC-CP-ABE is still less than that of FH-CP-ABE, and furthermore the private key storage consumption of AHAC-CP-ABE is obviously less than that of FH-CP-ABE.

Conclusions

Most of existing data access control schemes of CP-ABE do not consider the hierarchical access relationships of multiple shared data files, and just need data owners to generate multiple ciphertexts to meet the hierarchical access requirement, which would incur substantial computation overheads. To solve this problem, we first give an efficient hierarchical CP-ABE algorithm based on LSSS and furthermore, we



a Encryption time when the structure level increases



b Decryption time when the structure level increases

Fig. 7 Encryption and decryption time when the structure level increases

construct AHAC, which uses an integrated access structure that makes users be able to encrypt multiple data files with hierarchical access relationships at once. When a data visitor's attributes match a part of the access control structure, he can obtain the data that associate with this part by just one decryption. In addition, AHAC is secure, and has very low costs both in computation and storage aspects compared with related works.

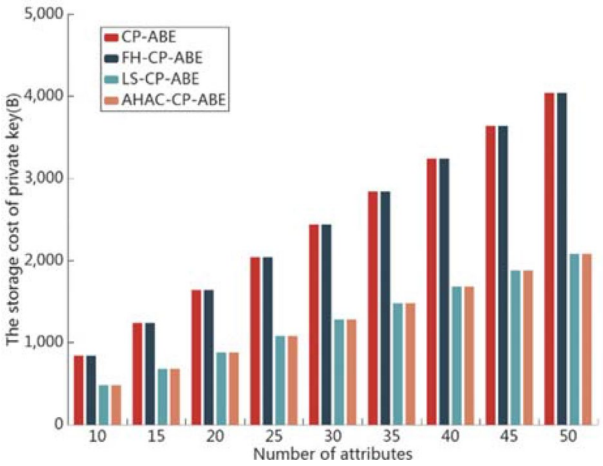


Fig. 8 The storage cost of private key

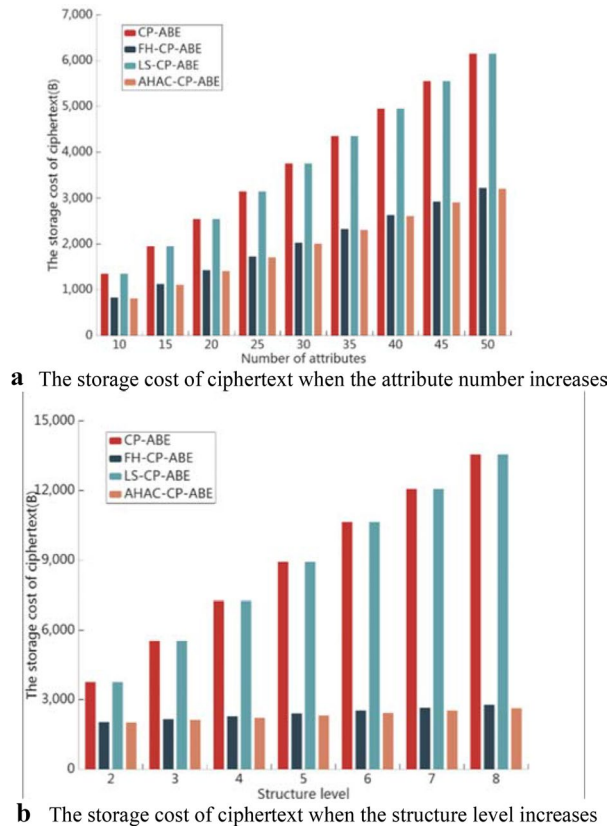


Fig. 9 The storage cost of ciphertext

In the future, we will work towards using blockchain technology to expand the single authority to multiple authorities, improve the security and stability of the authority, and support the accountability of authority.

Abbreviations
CP-ABE: Ciphertext-policy attribute-based encryption; AHAC: Attribute-based hierarchical data access control scheme; ABE: Attribute-based encryption; LSSS: Linear secret sharing scheme; PHR: Personal health record; FH-CP-ABE: File hierarchy attribute-based encryption scheme; d-BDH: Decisional-Bilinear Diffie-Hellman; d-Parallel BDHE: Decisional q-parallel Bilinear Diffie-Hellman Exponent; CA: Central Authority; CS: Cloud server; DO: Data owner.

Acknowledgements
Not applicable.

Author Contributions
Conceptualization HH and LZ; Implementation HH, LZ, and PL; Validation LD, LH, and XC; Writing and editing HH and LZ. All authors read and approved the final manuscript.

Funding
This work was supported by the National Natural Science Foundation of China under Grant Nos. 61602351, 61802286, 61502359, the Hubei Provincial Natural Science Foundation of China under Grant No. 2018CFB424.

Availability of data and materials
All data generated or analyzed during this study are included in this published article [and its supplementary information files]

Competing interests
The authors declare no competing interests.

Author details
[1] School of Computer Science and Technology, Wuhan University of Science and Technology, Wuhan, China. [2] Hubei Province Key Laboratory of Intelligent Information Processing and Real-Time Industrial System, Wuhan, China.

Received: 2 October 2019 Accepted: 20 November 2020
Published online: 05 December 2020

References

1. Rittinghouse JW, Ransome JF (2009) Cloud computing: implementation, management, and security. CRC press, Boca Raton
2. Kallahalla M, Riedel E, Swaminathan R, Wang Q, Fu K (2003) Scalable secure file sharing on untrusted storage. Paper presented at the 2nd USENIX Conference on File and Storage Technologies, San Francisco, CA, 31–31 March 2003
3. di Vimercati S D C, Foresti S, Jajodia S, Paraboschi S, Samarati P (2007) Over-encryption: management of access control evolution on outsourced data. Paper presented at the 33rd International Conference on Very Large Data Bases, Vienna, 23–27 September 2007
4. Ateniese G, Fu K, Green M, Hohenberger S (2006) Improved proxy re-encryption schemes with applications to secure distributed storage. ACM Trans Inf Syst Secur 9:1–30. https://doi.org/10.1145/1127345.1127346
5. Sahai A, Waters B (2005) Fuzzy identity-based encryption. Paper presented at the 24th annual international conference on Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, 22–26 May 2005
6. Bethencourt J, Sahai A, Waters B (2007) Ciphertext-policy attribute-based encryption. Paper presented at the 2007 IEEE Symposium on Security and Privacy, Washington, USA, 20–26 May 2007
7. Waters B (2011) Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. Paper presented at the 14th international conference on Practice and theory in public key cryptography conference on Public key cryptography, Taormina, Italy, 6–9 March 2011
8. Wang S, Zhou J et al (2016) An efficient file hierarchy attribute-based encryption scheme in cloud computing. IEEE Trans Inf Forensics Secur 11:1265–1277. https://doi.org/10.1109/TIFS.2016.2523941
9. Cheung L, Newport C (2007) Provably secure ciphertext policy ABE. Paper presented at the 14th ACM Conference on Computer and Communications Security, Alexandria, Virginia, 29 October–2 November 2007
10. Goyal V, Jain A, Pandey O, Sahai A (2008) Bounded ciphertext policy attribute based encryption. Paper presented at the 35th International Colloquium on Automata, Languages, and Programming, Reykjavik, Iceland, 7–11 July 2008
11. He H, Zhang J et al (2017) A fine-grained and lightweight data access control scheme for WSN-integrated cloud computing. Cluster Comput 20:1457–1472. https://doi.org/10.1007/s10586-017-0863-y
12. Li J, Zhang Y et al (2018) Secure attribute-based data sharing for resource-limited users in cloud computing. Comput Secur 72:1–12. https://doi.org/10.1016/j.cose.2017.08.007
13. Li J, Yao W et al (2017) Flexible and fine-grained attribute-based data storage in cloud computing. IEEE Trans Serv Comput 10:785–796. https://doi.org/10.1109/TSC.2016.2520932
14. Zhang Y, Zheng D et al (2018) Security and privacy in smart health: efficient policy-hiding attribute-based access control. IEEE Internet Things J 5:2130–2145. https://doi.org/10.1109/JIOT.2018.2825289
15. Kumar Premkamal Praveen, Kumar Pasupuleti Syam et al (2018) A new verifiable outsourced ciphertext-policy attribute based encryption for big data privacy and access control in cloud. J Ambient Intell Hum Comput 10:2693–2707. https://doi.org/10.1007/s12652-018-0967-0
16. Susilo W, Yang G, Guo F et al (2018) Constant-size ciphertexts in threshold attribute-based encryption without dummy attributes. Inf Sci 429:349–360. https://doi.org/10.1016/j.ins.2017.11.037

17. Wei T, Geng Y et al (2017) Attribute-based access control with constant-size ciphertext in cloud computing. IEEE Trans Cloud Comput 99:1–1. https://doi.org/10.1109/TCC.2015.2440247
18. Qiao H, Ren J et al (2018) Compulsory traceable ciphertext-policy attribute-based encryption against privilege abuse in fog computing. Future Gener Comput Syst 88:107–116. https://doi.org/10.1016/j.future.2018.05.032
19. Yu G, Ma X et al (2017) Accountable CP-ABE with public verifiability: how to effectively protect the outsourced data in cloud. Int J Found Comput Sci 28:705–723. https://doi.org/10.1142/S0129054117400147
20. Xue L, Yu Y et al (2018) Efficient attribute-based encryption with attribute revocation for assured data seletion. Inf Sci 479:640–650. https://doi.org/10.1016/j.ins.2018.02.015
21. Li J, Yao W et al (2017) User collusion avoidance CP-ABE with efficient attribute revocation for cloud storage. IEEE Syst J 99:1–11. https://doi.org/10.1109/JSYST.2017.2667679
22. Naruse T, Mohri M et al (2015) Provably secure attribute-based encryption with attribute revocation and grant function using proxy re-encryption and attribute key for updating. Hum Centric Comput Inf Sci 5:8. https://doi.org/10.1186/s13673-015-0027-0
23. Li R, Shen C, He H et al (2017) A lightweight secure data sharing scheme for mobile cloud computing. IEEE Trans Cloud Comput 99:1–1. https://doi.org/10.1109/TCC.2017.2649685
24. Khan F, Li H, Zhang L, et al (2017) An expressive hidden access policy CP-ABE. Paper presented at the 2017 IEEE Second International Conference on Data Science in Cyberspace, Shenzhen, China, 26–29 June 2017
25. He H, Li R, Dong X et al (2014) Secure, efficient and fine-grained data access control mechanism for P2P storage cloud. IEEE Trans Cloud Comput 2:471–484. https://doi.org/10.1109/tcc.2014.2378788
26. Chase M (2007) Multi-authority attribute based encryption. Paper presented at the 4th Theory of Cryptography Conference Amsterdam, The Netherlands, 21–24 February 2007
27. Bozovic V, Socek D, Steinwandt R et al (2012) Multi-authority attribute-based encryption with honest-but-curious central authority. Int J Comput Math 89:268–283. https://doi.org/10.1080/00207160.2011.555642
28. Wang Y, Li F, et al (2015) Achieving lightweight and secure access control in multi-authority cloud. Paper presented at the Trustcom 2015, Helsinki, Finland, 20–22 Aug 2015
29. Lin H, Cao Z, Liang X, Shao J (2008) Secure Threshold Multi Authority Attribute Based Encryption without a Central Authority. Paper presented at the 9th International Conference on Cryptology in India, Kharagpur, India, 14–17 December 2008
30. Chase M, Chow S S M (2009) Improving privacy and security in multi-authority attribute-based encryption. In Proceedings of the 16th ACM Conference on Computer and Communications Security, Chicago, IL, USA, 9–13 November 2009
31. Jung T, Li X, Wan Z, et al (2013) Privacy preserving cloud data access with multi-authorities. Paper presented at the INFOCOM 2013, Turin, Italy, 14–19 April 2013
32. Liu X, Ma J, Xiong J et al (2014) Ciphertext-policy hierarchical attribute-based encryption for fine-grained access control of encryption data. Int J Netw Secur 16:437–443. https://doi.org/10.6633/IJNS.201411.16(6).05
33. Beimel A (1996) Secure schemes for secret sharing and key distribution. Dissertation, Israel Institute of Technology
34. Lewko A, Waters B (2011) Decentralizing attribute-based encryption. Paper presented at the 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, 15–19 May 2011
35. Yu S, Wang C, Ren K, Lou W (2010) Achieving secure, scalable, and fine-grained data access control in cloud computing. In Proceedings of the 29th IEEE International Conference on Computer Communications, San Diego, California, USA, 14–19 March 2010
36. Caro AD, Iovino V (2011) jPBC: Java pairing based cryptography. In Proceedings of the 2011 IEEE Symposium on Computers and Communications, Kerkyra, Greece, 28 June–01 July 2011

Publisher's Note