Guan *et al. EURASIP Journal on Wireless Communications and Networking* (2016) 2016:208
DOI 10.1186/s13638-016-0705-2

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

An efficient traceable access control scheme with reliable key delegation in mobile cloud computing

Zhitao Guan*, Jing Li, Ying Zhang, Ruzhi Xu, Zhuxiao Wang and Tingting Yang

Abstract

With the increasing number of mobile applications and the popularity of cloud computing, the combination of these two techniques that named mobile cloud computing (MCC) attracts great attention in recent years. However, due to the risks associated with security and privacy, mobility security protection in MCC has become an important issue. In this paper, we propose an efficient traceable access control scheme with reliable key delegation named KD-TABE in MCC. Firstly, we present a traceable CP-ABE system and realize key delegation without loss of traceability. Secondly, a new type of re-encryption method is proposed, which is based on an intuitive method that supports any monotonic access tree instead of the re-encryption key. Lastly, to reduce trust on authority, we separate the authority into three parts, and each authority is responsible for generating different components of the key. The analysis shows that the proposed scheme can meet the security requirement of MCC. In addition, it cost less compared with the other popular models.

Keywords: Mobile cloud computing, Key delegation, Ciphertext delegation, Re-encryption

1 Introduction

With the increasing number of mobile applications and the popularity of cloud computing, the combination of these two techniques that named mobile cloud computing (MCC) attracts great attention listed by in recent years [1, 2]. MCC is a service that allows mobile users constrained with resources to adaptively adjust processing and storing capabilities by transparently partitioning and offloading the computationally intensive and storage demanding jobs on traditional cloud resources by providing ubiquitous wireless access [2]. In the former mobile computing paradigm, there are some problems such as resource scarcity, frequent disconnections, and mobility. With the support of MCC, the aforementioned problems can be addressed and the mobile users can achieve seamless access and handover for services, since mobile applications are executed on resource providers external to the mobile device.

However, the concerns with data privacy and security threats have become an obstacle to hinder MCC from being widely used [3]. According to the recent survey conducted by the International Data Corporation, most IT Executives and CEOs are not interested in adopting such services due to the risks associated with security and privacy [4, 5]. Therefore, it is necessary to eliminate the potential security threats in MCC.

Lots of researchers devote to security issues in MCC, such as secure MCC framework [4], access control [6–8], authentication [9], trust [10], and so on. The abovementioned works all focus on protecting MCC from external security threats. In [11], Liu et al. take insider threats into consideration and propose a solution named traceable ciphertext-policy attribute-based encryption (CP-ABE), which can trace the malicious users or traitors who intentionally leak the partial or modified decryption keys for profits. While, in traceable CP-ABE, key delegation is not supported, as they consider that key delegation will prevent most of the expressive ABE systems and their variants from being traceable [11]. Without key delegation, each new user has to apply to the authority for his own unique key. The overhead will be very heavy due to the large number of mobile users in MCC. In this paper, we propose an efficient traceable access control scheme with

* Correspondence: guanzhitao@126.com
School of Control and Computer Engineering, North China Electric Power University, Beijing, China

reliable key delegation named Key Delegation-Traceable Attributed Based Encryption (KD-TABE) to tackle with the aforementioned problems. And, the main contributions of this paper can be summarized as follows:

- We construct a traceable CP-ABE system with the access tree and realize the key delegation at servers without loss of traceability. We encrypt some components of the key to prevent the key from being maliciously delegated by malicious users or traitors.
- We propose a new type of re-encryption method, which is based on an intuitive method that supports any monotonic access tree, instead of the re-encryption key.
- To reduce trust on authority, we separate the authority into three parts, and each authority is responsible for generating different components of the key. One is trusted and responsible for user identity management; the other two are semi-trusted and responsible for generating temporary parameters.

The rest of this paper is organized as follows. Section 2 introduces the related work. Then, in Section 3, some preliminaries have been given. Our scheme is stated in Section 4. In Section 5, security analysis has been provided. In Section 6, we evaluate the performance of the proposed schema. Finally, the paper is concluded in Section 7.

2 Related work

Attribute-based encryption (ABE) is firstly proposed by Sahai and Brent in [12]. A user's identity is viewed as a set of descriptive attributes, the attributes are taken as public key, and the ciphertext will be decrypted as long as the number of user's attributes reaches a certain value which is set in the encryption process. Since then, ABE has become a research focus of the public key encryption field. Very soon afterwards, two ABE variants are proposed: key-policy attribute-based encryption (KP-ABE) and CP-ABE. In KP-ABE scheme [13], the data access policy (denoted as Au_KP) is specified by data users; the ciphertext is labeled by a set of attributes (denoted as A_o). The data user can decrypt the ciphertext only if A_o satisfies A_KP. While, in CP-ABE scheme [14], the data access policy (denoted as Ao_CP) is specified by data owners; the key is relevant to the attribute set A_u (A_u is holded by the data user). Only if A_u satisfies A_CP can the ciphertext be decrypted. Both KP-ABE and CP-ABE can achieve data confidentiality and fine-grained access control.

However, ABE allows users to share the same sets of attributes, and the decryption keys are generated with attributes sets without any identification information, which cause a problem: once the malicious key delegation happens, we cannot determine the owner's identification of the given key. To address this problem, there are two main ideas: one is to prevent the key from being cloned and misused, just like [15–19], the other one is to provide traceability, proposed and improved in [20–25]. In [20], the scheme proposes methods to trace the source of leaks and traitors in broadcast encryption; an index identifying a user is the foundation of realizing traceability. In [25], Ma et al. propose a new notion called multi-authority attribute-based traitor tracing. A pair of elements is introduced to describe a user, one represents its attribute set while the other one indicates its identity information. Based on the white box traceable CP-ABE [11], Liu et al. propose the black box traceable CP-ABE in 2015 [26]. In [11], the traceability is added to the CP-ABE scheme without weakening its security. Although the length of the ciphertext and decryption key is changed, the overhead is not increased significantly. In [27], the white-box traceable CP-ABE in large universe is realized and the storage for traitor tracing is constant. Katz et al. introduce the traceability into predicate encryption schemes in [28], which has the general applicability. Ning et al. firstly propose an accountable authority CP-ABE scheme that supports white box traceability in [29], which solved two types of key abuse problems simultaneously.

There are several studies on ciphertext delegation, and one of the approaches is to utilize proxy re-encryption. The proxy re-encryption(PRE) technique encrypts the ciphertexts with re-encryption keys and makes it possible for users to decrypt the re-encrypted ciphertexts with their own original decryption keys without changing. In [30], Luo et al. realize proxy re-encryption in CP-ABE scheme, with AND-gates that support multi-value attributes, negative attributes, and wildcards, the encryptor could choose any ciphertext to re-encrypt as they like. Lai et al. formalize a new cryptographic primitive called adaptable ciphertext-policy attribute-based encryption, which allows a semi-trusted proxy to modify a ciphertext under one access policy into another one of the same plaintext under any other access policies [31]. In [32, 33], Kaitai Liang et al. further optimize the system security and integrate the dual system encryption technology to realize the adaptively CCA secure in the standard model.

Table 1 shows the comparison between our work and other related works.

3 Preliminaries

3.1 Bilinear maps and complexity assumptions

Let G_0 and G_1 be two multiplicative cyclic groups of prime order p and g be the generator of $G_{0.}$

Table 1 Comparison with other related works

	Traceability	Supporting monotone access structures	Key delegation	Cipher delegation
Reference [11]	√	√	×	×
Reference [14]	×	√	√	×
Reference [26]	√	√	×	×
Reference [27]	√	√	×	×
Reference [28]	√	√	×	×
Reference [29]	√	√	×	×
Our work	√	√	√	√

The bilinear map e is, $e{:}G_0 \times G_0 \rightarrow G_1$, for all $a, b \in \mathbb{Z}_p$:

1. Bilinearity: $\forall u, v \in G_1, e(u^a, v^b) = e(u, v)^{ab}$
2. Non-degeneracy: $e(g, g) \neq 1$
3. Symmetric: $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$

Definition 1 *Discrete logarithm (DL) problem:*

Let G be a multiplicative cyclic group of prime order p and g be its generator, given $y \in_R G$ as input, try to get $x \in \mathbb{Z}_p$ that $y = g^x$.

The DL assumption holds in G if it is computationally infeasible to solve DL problem in G.

3.2 Access structure

Let $P = \{P_1, P_2, \ldots, P_n\}$ be a set of participants, let $U = 2^{\{P1, P2, \ldots, Pn\}}$ be the universal set. If $\exists AS \subseteq U \backslash \{\varnothing\}$, then AS can be viewed as an access structure.

If $A \in AS, \forall B \in U, A \subseteq B$, and $B \in AS$, then AS is monotonic.

If AS is an access structure, then the sets in it are called the authorized sets, and the sets not in it are called the unauthorized sets.

The access structure in our system is an access tree, which is the same as in [14]. The tree includes a root node, some interior nodes and some leaf nodes. The leaf nodes are associated with descriptive attributes while the interior nodes represent the logic operation, such as AND (n of n), OR (1 of n), n of m (m > n). A user can decrypt the ciphertext correctly only if the access tree is satisfied by his attributes set.

3.3 Notations

In Table 2, the notations used in this paper are listed. DO and DR are cloud users. DR1, DR2, and DR3 denote different data receivers. IA, RA, and AA are responsible for generating essential components of SK. KDS plays a role of generating new SK for new coming users based on the given SK if and only if new user's attribute set is the subset of the attribute set of the data owner that owns the given SK. CDS supports the user to re-encrypt his ciphertext with his own new access structure (AS) without the need to decrypt it.

Table 2 Notations

Acronym	Descriptions
PK	Public key
MK	Master key
SK	Secret key
CT	Ciphertext
M	Plaintext
AS	Access structure
DSK	Delegated secret key
DCT	Delegated ciphertext
DO	Data owner
DR	Data requester/receiver
IA	Identification authority
RA	Random number authority
AA	Attribute authority
KDS	Key delegation server
CDS	Ciphertext delegation server
CS	Cloud server
TDS	Trusted decryption server
SDS	Semi-trusted decryption server

4 Our system

We construct a new traceable CP-ABE system with access tree and focus on how to realize the key delegation and the ciphertext delegation based on our system. In order to achieve all this, our system is composed of the following parties: a Data Owner, some Data Receivers, and three authorities, the Key Delegation Servers, the Ciphertext Delegation Servers, the Cloud Server and two Decryption Servers.

DO sends his ciphertext to CS. We list three users in Fig. 1, which are viewed as the different participants in three functions. DR1 applies to authorities for decryption key, and the three authorities (IA, RA, AA) collaborate to generate a complete decryption key. Different authority outputs different key components. DR2 is a new user whose attributes set is a subset of DR1. KDS can be viewed as a substitute for AA, which is used for generating attribute-related components according to the DR1's key and finally outputs a DSK for DR2. There are some ciphertexts that can be decrypted by DR1, but DR1 wants some other users who belong to some attributes sets to decrypt them without changing the users' decryption keys. At this point, the CDS will handle the request and finally outputs a DCT for DR3 (DR3 denotes the eligible).

We use an access tree $\tilde{T}$ to express the access policy specified by data owners. We introduce a hash function $H{:}\{0, 1\}^* \rightarrow G_0$ and view it as a random oracle, which maps any attribute described as a binary string to a random group element.

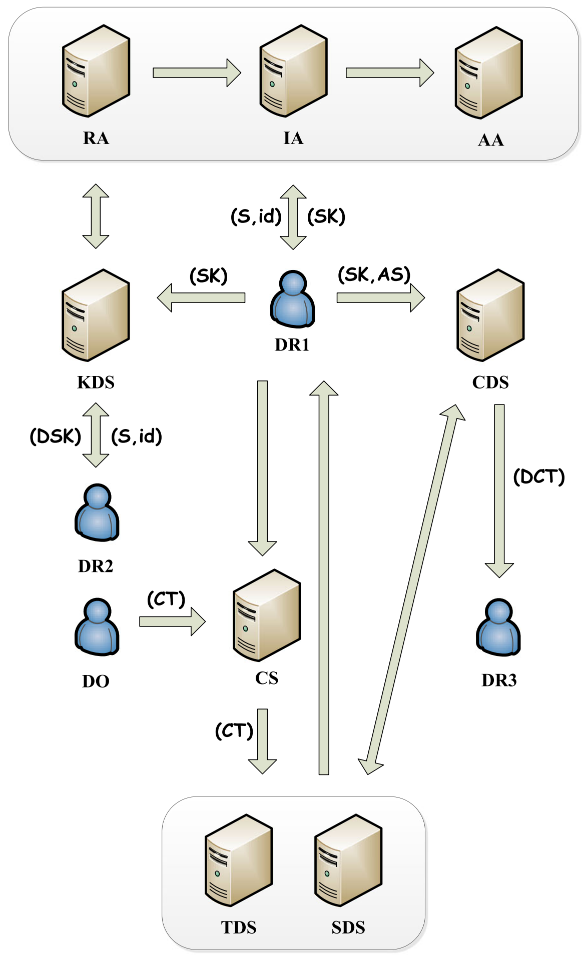

Fig. 1 System model

4.1 Setup

When the system starts up, the setup algorithm will choose a multiplicative cyclic group G_0 of prime order p with generator g and three random numbers $a, \alpha, \beta \in \mathbb{Z}_P$.

The public key is:

$$PK = \{G_0, g, h = g^\beta, e(g,g)^\alpha, k = g^\omega, l = k^a\}$$

The master key is:

$$MK = \{\beta, g^\alpha, a, \omega\}$$

This paper uses the Shamir' $(\bar{t}, \bar{n})$ threshold gates scheme to store tracing information that proposed in [27].

IA receives all MK components and keeps $f(x)^7$ and $\bar{t}-1$ points $\{(x_1 y_1), (x_2 y_2), \ldots, (x_{\bar{t}-1} y_{\bar{t}-1})\}$ as secret [27], while AA and RA get one of the MK component a, which is used to protect the parameters transmitted between them.

A symmetric encryption algorithm is introduced into the three authorities to encrypt the components of a secret key. The symmetric encryption keys are assigned to the three authorities, IA receives K_1 and K_2, RA receives K_3, and AA receives K_4. KDS receives K_4, TDS receives K_2, and SDS receives K_3 and K_4. CDS acquires the Hash function.

4.2 Encrypt (PK, M, T)

The encryption algorithm receives message M and access structure T (denoted by an access tree) from DO. First, the algorithm chooses a random number $s \in \mathbb{Z}_P$ for root node R polynomial, which means $q_R(0) = s$. Then, it chooses a polynomial q_x for each node x (leaf or none-leaf node) in top down manner, with the same method to construct the polynomials proposed in [25].

Let Y be the set of leaf nodes in T. The structure of the ciphertext is as follows:

$$CT = \begin{pmatrix} \tilde{T}, \tilde{C} = Me(g,g)^{\alpha s}, C = h^s, \\ \forall y \in Y : C_y = g^{q_y(0)}, C_y' = H(att(y))^{q_y(0)} \end{pmatrix}$$

4.3 Key generation (MK, PK, id, S)

As shown in Fig. 2, to reduce trust on authority, we split the authority into three: IA, RA, and AA. They generate different decryption key components respectively: encrypted random number r, encrypted id, and attribute-related components. Assume that collusion is prohibited, none of them will get complete key information.

When a user DR1 submits his id and attributes set to IA and AA, both of them run the algorithm to compute

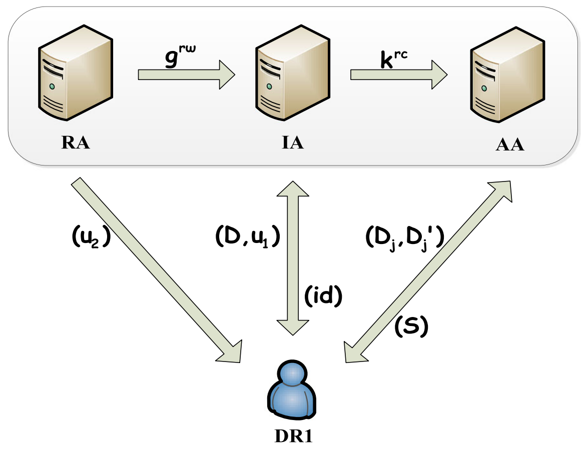

Fig. 2 Key generation

the id and attributes. RA chooses a random number $r \in \mathbb{Z}_P$ and transmits to DR1 after encrypting it:

$$u_2 = Enc_{K_3}(r)$$

Recording and encrypting the random number r aim to prevent the decryption key from being re-randomized at the user side. u2 will be decrypted when the decryption starts, and the random number r got from u_2 is used for decrypting. If the re-randomization occurs, the random number in keys will be changed, it will differ from r. Thus, decryption will be failed.

The parameter $g^{r\omega}$ will be received at IA, and IA computes as follows:

$$\begin{aligned} & x = Enc_{K_1}(id) \\ & y = f(x) \\ & c = x||y \\ & u_1 = Enc_{K_2}(c) \\ & D = g^{\frac{\alpha}{\beta}}(g^{r\omega})^{\frac{(a+c)}{\beta}}(g^{r\omega})^{\frac{1}{\omega}} = g^{\frac{\alpha+(a+c)\omega r}{\beta}} g^r \\ & (g^{r\omega})^c = k^{rc} \end{aligned}$$

u_1 and D will be sent to DR1, and k^{rc} will be sent to AA.

AA gets the parameters from IA, chooses random numbers $r_j \in \mathbb{Z}_P, \forall j \in S$, and computes as follows:

$$\begin{aligned} & D_j = k^{rc} H(j)^{r_j} \\ & D_j' = Enc_{K_4}(g^{r_j}) \end{aligned}$$

DR1 merges the key components from IA, RA, and AA:

$$SK = \begin{pmatrix} D = g^{\frac{\alpha+(a+c)\omega r}{\beta}} g^r, D_0 = u = u_1||u_2, \\ \forall j \in S, D_j = k^{rc} H(j)^{rj}, D_j' = Enc_{K_4}(g^{rj}) \end{pmatrix}$$

4.4 Key trace (SK)

The trace algorithm reference the method proposed in [27]. First, the algorithm decrypts u_1 to get (x,y) from D_0 in user's key, and then, it checks whether SK is issued by system.

If $(xy) \in \{(x_1 y_1), (x_2 y_2), \ldots, (x_{\bar{t}-1} y_{\bar{t}-1})\}$, the algorithm decrypts x to get id of the user. Otherwise, the algorithm computes the secret of INS $(\bar{t}, \bar{n})$ by interpolating with $\bar{t}$ points $\{(xy), (x_1 y_1), (x_2 y_2), \ldots, (x_{\bar{t}-1} y_{\bar{t}-1})\}$. If the recovered secret is equal to $f(0)$, the algorithm decrypts x to get id of the user. If not, SK is not issued by the system and cannot be traced.

IA stores the $f(x)$ when system sets up, and it holds the symmetric keys K_1 and K_2 that can decrypt x. So, IA runs the algorithm when a key needs to be traced.

4.5 Key delegation (SK, id')

As shown in Fig. 3, KDS runs this algorithm when a new user's attribute set is a subset of the given key's owner. Let us assume that DR1 is the given key's owner and DR2 is the new user whose set is a subset of DR1.

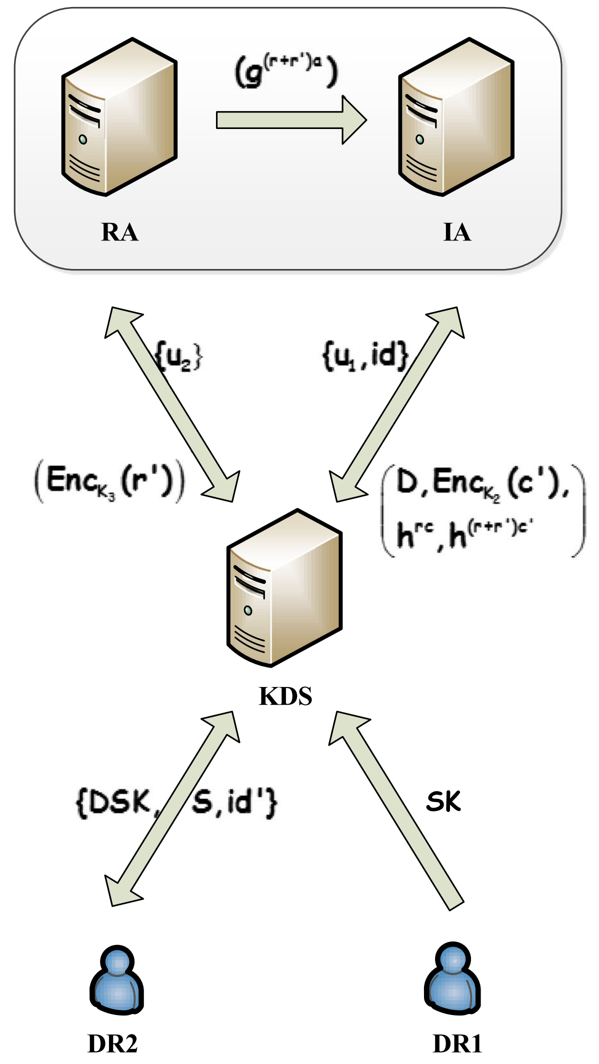

Fig. 3 Key delegation

IA and RA can decrypt u_1 and u_2, respectively; the former will compute new (x,y) according new user's id, and the latter will re-randomize the random number.

The parameter generation and transmission process of this section is similar to the key generation. RA gets the random number r from the given key SK and chooses a new random number $r' \in \mathbb{Z}_P$ for DR2, passing $g^{(r+r')\omega}$, $g^{r\omega}$ to IA.

$$u_2 = Enc_{K_3}(r + r')$$

IA gets the parameter from RA, gets c from the given key SK, and computes as follows: (id' denotes the DR2's id)

$$\begin{aligned}
&(g^{r\omega})^c = k^{rc}\\
&x = Enc_{K_1}(id')\\
&y = f(x)\\
&c' = x||y\\
&u_1 = Enc_{K_2}(c')\\
&\left(g^{(r+r')\omega}\right)^{c'} = k^{(r+r')c'}\\
&D = g^{\frac{\alpha}{\beta}}\left(g^{\frac{(r+r')(a+c)}{\beta}}\right)g^{r+r'} = g^{\frac{\alpha+(a+c)\omega(r+r')}{\beta}}g^{r+r'}
\end{aligned}$$

KDS receives D, u_2, u_1, k^{rc}, $^{k(r+r')c'}$ and chooses $r_j{}' \in \mathbb{Z}_P$, then computes as follows:

$$\begin{aligned}
&r_j r_j{}' = r_m\\
&D_m = k^{(r+r')c'}\left(\frac{D_j}{k^{rc}}\right)^{r_j{}'} = k^{(r+r')c'}H(m)^{r_m}\\
&Dec_{K_4}(D_m{}') = g^{r_j r_j{}'} = g^{r_m}
\end{aligned}$$

We assume that KDS does not keep any information about the Hash function, so Dk cannot be calculated like this:

$$\begin{aligned}
D_m &= k^{(r+r')c'}\left(\frac{D_j}{k^{rc}}\right)H(m)^{r_j}H(m)^{r_j{}'}\\
&= h^{(r+r')c'}H(m)^{r_j+r_j{}'}
\end{aligned}$$

The new delegation key is:

$$SK' = \begin{pmatrix} D = g^{\frac{\alpha+(a+c)\omega(r+r')}{\beta}}g^{r+r'}, D_0 = u = u_1||u_2,\\ \forall m \in S, D_m = k^{(r+r')c'}H(m)^{r_m}, D_k{}' = Enc_{K_4}(g^{r_m}) \end{pmatrix}$$

4.6 Ciphertext delegation (CT, PK, AS, D_0, D_j, D_j')

CDS is a trusted server, it stores the Hash function $H:\{0,1\}^* \to G_0$ to generate new access tree according the given access structure. It receives some decryption key components and a new access structure AS from DR1, while D_0 includes the owner's identity information and random number, which can be used for decrypting.

We first decrypt the access tree embedded in the ciphertext and get two expressions. Next, the ciphertext will be re-encrypted with AS. Then, D_0, the expressions, and re-encrypted ciphertext make up the new delegated ciphertext.

When x is a leaf node, let $i = \text{att}(x)$. Function att(x) denotes the attribute associated with the leaf node x in the tree. The ciphertext is not required to be decrypted completely, the components D_0, D_j, D_j' are enough. Both of DecryptNodeL_A' DecryptNodeL_A$_0$ and DecryptNodeNL are run in SDS. It can decrypt u_2 to get r.

$$\text{If } i \in S \quad \begin{aligned}
&DecryptNodeL\_A'(CT,PK,SK,x)\\
&= \frac{e(D_i(l)^r, C_x)}{e(D_i', C_x')}\\
&= \frac{e\left(k^{rc}H(i)^{r_i}(g^{\omega a})^r, g^{q_x(0)}\right)}{e(g^{r_i}, H(i)^{r_i})}\\
&= \frac{e\left(k^{r(a+c)}, g^{q_x(0)}\right)e\left(H(i)^{r_i}, g^{q_x(0)}\right)}{e(g^{r_i}, H(i)^{r_i})}\\
&= \frac{e\left(k^{r(a+c)}, g^{q_x(0)}\right)e\left(H(i), g^{q_x(0)}\right)^{r_i q_x(0)}}{e\left(g, H(i)^{r_i q_x(0)}\right)}\\
&= e\left(g^{\omega r(a+c)}, g^{q_x(0)}\right)\\
&= e(g,g)^{\omega r(a+c)q_x(0)}\\
&DecryptNode(CT, D_0, D_i, D_i', R) = e(g,g)^{\omega r(a+c)s}
\end{aligned}$$

Otherwise,

$i \notin S$, $DecryptNodeL\_A'(CT, PK, SK, x) = \perp$ When x is an interior node, call the algorithm DecryptNodeNL (CT,D$_0$,D$_i$,D$_i$',x).

For all nodes z that are children of x, it calls DecryptNodeL_A'(CT,PK,D$_0$,D$_i$,D$_i$',z) and stores the output as Fz. Let S_x be a k_x (the threshold value of interior node) random set and let $F_z \neq \perp$. If no such set exists, the function outputs $\perp$.

Otherwise, compute as follows and return the result:

$$\begin{aligned}
F_x &= \prod_{z \in S_x} F_z{}^{\Delta_{i,S_x}{}'(0)}, \quad \text{where } \begin{cases} i = \text{index}(z) \\ S_x' = \{\text{index}(z) : z \in S_x\} \end{cases}\\
&= \prod_{z \in S_x}\left(e(g,g)^{r.q_z(0)\omega(c+a)}\right)^{\Delta_{i,S_x'}(0)}\\
&= \prod_{z \in S_x}\left(e(g,g)^{\omega(c+a)r.q_{\text{parent}(z)}(index(z))}\right)^{\Delta_{i,S_x'}(0)}\\
&= \prod_{z \in S_x}\left(e(g,g)^{\omega(c+a)r.q_x(i)}\right)^{\Delta_{i,S_x'}(0)}\\
&= e(g,g)^{\omega(c+a)r \cdot q_x(0)}
\end{aligned}$$

For root node R,

$$A' = e(g,g)^{\omega sr(c+a)}$$

Repeat the steps described above, however, DecryptNodeL_A' algorithm is changed to DecryptNodeL_A0:

$$\begin{aligned}
&DecryptNodeL\_A_0(CT, D_0, D_i, D_i', x)\\
&= \frac{e(D_i, C_x)}{e(Dec_{K_4}(D_i'), C_x')}\\
&= e(g,g)^{\omega q_x(0)rc}
\end{aligned}$$

For root node R,

$$A_0 = e(g,g)^{\omega src}$$

Then, $A_1 = \frac{A'}{A_0} = e(g,g)^{\omega sra}$

$$\hat{C} = A_0||A_1||D_0$$

A_0 and A_1 are the expressions mentioned above; they will become the components of the delegated ciphertext with D_0.

Next, CDS re-encrypts the other CT components as the Encrypt algorithm does. It constructs an access tree $\tilde{T}$ for AS specified by DR1 and chooses a new random number $s' \in \mathbb{Z}_p$:

$$\begin{aligned}
&\tilde{C} = (Me(g,g)^{\alpha s})e(g,g)^{\alpha s'}\\
&C = h^s h^{s'}\\
&\hat{C} = A_0||A_1||D_0,\\
&C_z = g^{q_z(0)}, z \in S_{AS}\\
&C_z{}' = H(att(z))^{q_z(0)}, z \in S_{AS}
\end{aligned}$$

We get the delegated ciphertext as follows:

$$CT' = \left(\tilde{T}, \tilde{C}, \hat{C}, C, C_z, C_z{}'\right)$$

4.7 Decrypt (PK, CT, SK)

In fact, decryption can be viewed as two parts: satisfying the access tree and decrypting the ciphertext.

We introduce two servers to carry on the process respectively. SDS possesses symmetric key K_4 and K_3 so that it can get random number and verify whether his set satisfies the access tree or not and run the algorithms: DecryptNodeL_A', DecryptNodeL_A0, DecryptNodeNL. TDS possesses symmetric key K_2 so that it can decrypt u_1.

The general decryption process is described as follows:

When x is a leaf node, let $i = \text{att}(x)$. Function att(x) denotes the attribute associated with the leaf node x in the tree. SDS decrypts u_2 to get random number r.

$$\begin{aligned}
&DecryptNodeL\_A'(CT,PK,SK,x)\\
&\qquad = \frac{e(D_i(l)^r, C_x)}{e(D_i{}', C_x{}')}\\
&\qquad = \frac{e\left(k^{rc}H(i)^{r_i}(g^{\omega a})^r, g^{q_x(0)}\right)}{e(g^{r_i}, H(i)^{r_i})}\\
\text{If } \ If \ \ i \in S \qquad &\qquad = \frac{e\left(k^{r(a+c)}, g^{q_x(0)}\right)e\left(H(i)^{r_i}, g^{q_x(0)}\right)}{e(g^{r_i}, H(i)^{r_i})}\\
&\qquad = \frac{e\left(k^{r(a+c)}, g^{q_x(0)}\right)e\left(H(i), g^{q_x(0)}\right)^{r_i q_x(0)}}{e\left(g, H(i)^{r_i q_x(0)}\right.}\\
&\qquad = e\left(g^{\omega r(a+c)}, g^{q_x(0)}\right)\\
&\qquad = e(g,g)^{\omega r(a+c)q_x(0)}\\
&DecryptNode(CT, D_0, D_i, D_i{}', R) = e(g,g)^{\omega r(a+c)s}
\end{aligned}$$

Otherwise, $i \notin S$ $DecryptNodeL\_A'(CT, PK, SK, x) = \perp$

When x is an interior node, call the algorithm DecryptNodeNL(CT,SK,x).

For all nodes z that are children of x, it calls DecryptNodeL_A'(CT,PK,SK,z) and stores the output as Fz. Let S_x be a k_x (the threshold value of interior node) random set and let $F_z \neq \perp$. If no such set exists, the function outputs $\perp$.

Otherwise, compute as follows and return the result:

$$\begin{aligned}
F_x &= \prod_{z \in S_x} F_z{}^{\Delta_{i,S_x{}'}(0)}, \quad \text{where } \begin{cases} i = \text{index}(z) \\ S_x' = \{\text{index}(z) : z \in S_x\} \end{cases}\\
&= \prod_{z \in S_x}\left(e(g,g)^{r.q_z(0)\omega(c+a)}\right)^{\Delta_{i,S_x'}(0)}\\
&= \prod_{z \in S_x}\left(e(g,g)^{\omega(c+a)r.q_{\text{parent}(z)}(index(z))}\right)^{\Delta_{i,S_x'}(0)}\\
&= \prod_{z \in S_x}\left(e(g,g)^{\omega(c+a)r.q_x(i)}\right)^{\Delta_{i,S_x'}(0)}\\
&= e(g,g)^{\omega(c+a)r\cdot q_x(0)}
\end{aligned}$$

For root node R,

$$A' = e(g,g)^{\omega sr(c+a)}$$

Then, TDS decrypts u_1 and computes as follows:

$$\begin{aligned}
r &= Dec(u_1)\\
B &= \frac{\tilde{C}}{e\left(C, \frac{D}{g^r}\right)} = \frac{Me(g,g)^{\alpha s}}{e\left(h^s, g^{\frac{\alpha+(a+c)\omega r}{\beta}}\right)}\\
&= \frac{Me(g,g)^{\alpha s}}{e\left(g^{\beta s}, g^{\frac{\alpha+(a+c)\omega r}{\beta}}\right)} = \frac{Me(g,g)^{\alpha s}}{e(g,g)^{(\alpha+\omega r(a+c))s}}
\end{aligned}$$

The user gets A' from SDS and gets B from TDS:

$$A' = e(g,g)^{\omega r(a+c)s}$$

When a user DR3 try to decrypt a delegated ciphertext from DR1, the structure of the ciphertext is as follows: r' and c' denote DR1's random number and identity information and r and c denote DR3's random number and identity information.

$$CT' = \left(\tilde{T}, \tilde{C}, \hat{C}, C, C_z, C_z{}'\right)$$

Compared with the original ciphertext, the massage M is encrypted with (s + s') instead of s, and the access structure is replaced by AS. DR3 decrypts the access tree and get an expression related to s', which can be used for decrypting correctly with C' and C0 in DCT.

We can get $\hat{C}$ from CT': $\hat{C} = A_0||A_1||D_0$,

SDS computes (we only consider when the user's set satisfies the access tree):

$$DecryptNodeL\_A_0(CT', SK', x) = \frac{e(D_i, C_x)}{e(Dec_{K4}(D_i'), C_x')} = e(g,g)^{\omega q_x'(0) r' c'}$$

$$DecryptNodeL\_A'(CT', SK', z) = \frac{e\left(D_i(h^a)^{r'}, C_z\right)}{e(Dec_{K_4}(D_i'), C_z')} = e(g,g)^{\omega q_z'(0) r'(c'+a)}$$

For root node *R*', the intermediate calculation process is omitted.

$$B_0 = e(g,g)^{\omega s' r' c'}$$
$$B' = e(g,g)^{\omega s' r'(a+c')}$$
$$\hat{B} = \frac{B'}{B_0} = e(g,g)^{\omega s' r' a}$$

All of them are transferred to SDS:

$$r' = Dec(u_2')$$
$$r = Dec(u_2)$$
$$B_1 = (A_0)^{\frac{r'}{r}} = (e(g,g)^{\omega r a s})^{\frac{r'}{r}} = e(g,g)^{\omega r' a s}$$
$$B_2 = \hat{B} = e(g,g)^{\omega r' a s'}$$
$$B_3 = (A_1)^{\frac{r'}{r}} = (e(g,g)^{\omega r c s})^{\frac{r'}{r}} = e(g,g)^{\omega r' c s}$$
$$B_4 = IR_1' = e(g,g)^{\omega r' c' s'}$$
$$A = \frac{\tilde{C} B_1 B_2}{e\left(C, \frac{D}{g^{r'}}\right)} = \frac{Me(g,g)^{\alpha(s+s')}\left(e(g,g)^{\omega r' a s}\right)\left(e(g,g)^{\omega r' a s'}\right)}{e\left(h^{(s+s')}, g^{\frac{\alpha+(a+c')\omega r'}{\beta}}\right)}$$
$$= \frac{Me(g,g)^{\alpha(s+s')} e(g,g)^{\omega r' a(s+s')}}{e\left(g^{\beta(s+s')}, g^{\frac{\alpha+(a+c')\omega r'}{\beta}}\right)} = \frac{Me(g,g)^{(\alpha+\omega r' a)(s+s')}}{e(g,g)^{(\alpha+\omega r'(a+c'))(s+s')}}$$

A and B_3 will be sent to TDS:

$$c = Dec(u_1) \qquad c' = Dec(u_1')$$
$$B_5 = \left(e(g,g)^{\omega r' c s}\right)^{\frac{c'}{c}} = e(g,g)^{\omega r' c' s}$$
$$M' = AB_5 = \frac{Me(g,g)^{(\alpha+\omega r' a)(s+s')} e(g,g)^{\omega r' c' s}}{e(g,g)^{\left(\alpha+\omega r'\left(a+c'\right)\right)\left(s+s'\right)}}$$

The user will get *M*' from TDS and B_4 from SDS; thus, the message can be calculated as follows:

$$M = M' B_4 = \frac{Me(g,g)^{(\alpha+\omega r' a)(s+s')} e(g,g)^{\omega r' c' s} e(g,g)^{\omega r' c' s'}}{e(g,g)^{\left(\alpha+\omega r'\left(a+c'\right)\right)\left(s+s'\right)}}$$
$$= \frac{Me(g,g)^{(\alpha+\omega r'(a+c'))(s+s')}}{e(g,g)^{(\alpha+\omega r'(a+c'))(s+s')}}$$

5 Security analysis

5.1 Traceability

In [22], the access structure is a share-generating matrix, while in our system, the access structure is an access tree.

Theorem 1 *The security of traceability in our system is no weaker than that of [22].*

Proof The decryption key in our system has partially similar structure with the key in [22]. $D = g^{\frac{\alpha+(a+c)\omega r}{\beta}} g^r$ includes master keys β, g^α, a, ω, a random number r, and a parameter c that denotes the user's identity information. $K = g^{\frac{\alpha}{(a+c)}} h^t R$ includes master keys α, a, public key h, random numbers t and R, and a parameter c that denotes the user's identity information. Compared with K in [22], we construct D with extra master keys β and ω, without any public keys. It is pretty difficult to get anyone of $g^{\frac{\alpha}{\beta}}, g^r, g^{\frac{(a+c)}{\beta}}$.

According to security proof in [22], the design of the decryption key is secure. Thus, $D = g^{\frac{\alpha+(a+c)\omega r}{\beta}} g^r$ in our system is secure.

Theorem 2 *The security of the traceable decryption key in our system is no weaker than that of [25].*

Proof In $D_j = k^{rc} H(j)^{r_j}$, we add master key ω and the parameter c to D_j in [25] $\left(D_j = g^r H(j)^{r_j}\right)$. Additionally, the other components are protected by symmetric encryption. Thus, the security in this component is no weaker than that of [25].

5.2 Key delegation

This part of function is realized by KDS, IA and RA, which means AA is replaced by KDS to some extent. *SK* = (*D*, D_0, D_k, D_k'), *D* and D_0 are generated by IA and RA, and D_k and D_k' will be calculated by KDS according to the given key's information.

Theorem 3 *It is computationally infeasible to attack the calculation of KDS.*

Proof KDS receives another two parameters k^{rc}, $k^{(r+r')c'}$ from IA, which can be used for calculating and re-randomizing attribute-related components of the key. The values all of those KDS receives and retrieves are $k^{rc}, k^{(r+r')c'}, k^{rc} H(j)^{r_j}, g^{r_j}, H(j)^{r_j}$

According to DL problem, it is computationally infeasible to retrieve r_j, rc and $(r + r')c'$, let alone r or c. Thus, a semi-trusted server can be designated as KDS, which is competent in this work.

5.3 Ciphertext delegation

It can be considered as two parts: decrypting the access tree and re-encrypting with another specified access tree. Decrypting the access tree in our system is analogous to the decrypt algorithm in [25]. And, re-encryption is analogous to the encrypt algorithm in [25].

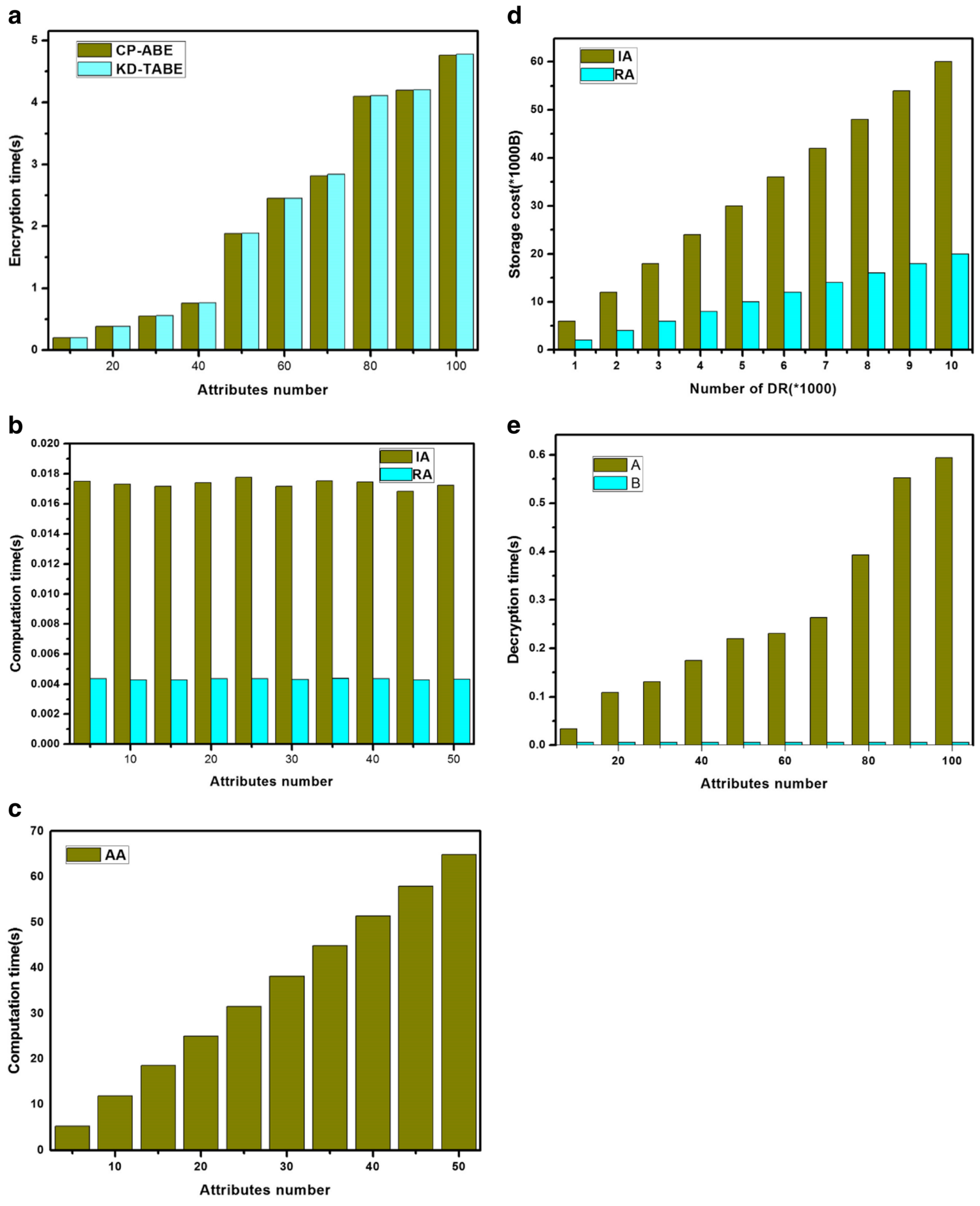

Fig. 4 a Encryption cost in CP-ABE and KD-TABE. **b** Computation cost of RA and IA is constant, which has no relationship with attributes number of DR's attributes set. **c** Computation cost of AA grows with the rise of attributes number in DR's attributes set. **d** Storage cost of RA and IA grows with the rising of DR. **e** Decryption cost of SDS (*A*) and TDS(*B*)

Theorem 4 *The security in ciphertext delegation is no weaker than that in [25].*

$A' = e(g,\ g)^{\omega sr(c+a)}$, $A_1 = e(g,\ g)^{\omega sra}$, $A_0 = e(g,\ g)^{\omega src}$ are intermediate results in our system, compared with $A = e(g, g)^{sr}$ in [14], the extra master key β, a is introduced to further protect the random number s.

$$\tilde{C}' = Me(g,g)^{\alpha s} e(g,g)^{\alpha s'} = Me(g,g)^{\alpha(s+s')}$$
$$C' = h^s h^{s'} = h^{(s+s')}$$

We re-encrypt with a new $s' \in \mathbb{Z}_p$ chosen by CDS, $\tilde{C}$, C have been proved to be secure in [25]. Thus, by that analogy, $\tilde{C}', C'$ in our system is secure.

6 Performance evaluation

6.1 Setup

The Setup procedure includes defining multiplicative cyclic group and generating PK and MK that will be used in encryption and key generation. There are three exponentiation operations and one pairing operation; three random numbers are chosen in Setup procedure. Time complexity of the procedure is O(1).

6.2 Encrypt

Plaintext and access tree are encrypted in this procedure, the computation cost is proportional to the number of attributes in the tree. If the universal attributes set in T is I ($|I|$ denotes the total number of attribute in set I), for each element in I, it needs two exponentiation operations, totally, the computation complexity is O ($|I|$). Compared with CP-ABE, we add an exponentiation operation in this algorithm, and the added operation has no obvious influence on computation cost as shown in Fig. 4a.

6.3 Key generation

This algorithm includes three parts: RA tackles the random number computation; IA computes the id information; AA generates the attribute-related key components. The time complexity of RA and IA is O(1) as shown in Fig. 4b. Computation cost of AA is proportional to the number of attributes is DR's set, when the attribute number is m, the time complexity is O(m) as shown in Fig. 4c.

We add the severe RA and IA to achieve the id embedding in SK, RA needs to store the computation result of u_2, IA stores results of u_1, D, k^{arc}, the total storage cost is proportional to the number of DR. If each result needs 2 bytes, the storage cost of RA and IA is shown in Fig. 4d.

6.4 Key delegation

For each DR, KDS computes the unique SK. The computation process and cost are similar to RA. The total cost is proportional to the number of DR.

6.5 Decrypt

This algorithm also includes three parts: SDS decrypts the access tree in CT, TDS decrypts the rest part, and DR retrieves the final message. Thus, computation cost of SDS is proportional to the attributes number in DR's set, and computation cost of TDS is constant as shown in Fig. 4e.

In brief, our system costs no more than CP-ABE by analysis but achieves more functions compared with existing works [22, 25].

7 Conclusions

Traceable CP-ABE is an important branch of CP-ABE, retaining the characteristics of CP-ABE. However, it cannot trace the owner with a decryption key. In contrast, key delegation is not supported in traceable CP-ABE. Without key delegation, the overhead will be very heavy due to the large number of new coming mobile users in MCC. Therefore, we re-constructed a new traceable CP-ABE system that supported key delegation and ciphertext delegation. We realized key delegation without loss of the traceability with the same computation overhead. To realize ciphertext delegation, we abandoned the re-encryption key and tried to decrypt the access tree first and re-encrypt the ciphertext with any monotonic access tree specified by the user next. In the future, we will study on making our system work under large universe and support more functions such as revocation.

Acknowledgements
This work was jointly supported by the National Natural Science Foundation of China (No. 61402171, No. 61305056, and No. 61300132), Beijing Higher Education Young Elite Teacher Project (No. YETP0702), and the Fundamental Research Funds for the Central Universities (Nos. 2015MS35 and 2016MS29).

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

Received: 1 February 2016 Accepted: 26 August 2016
Published online: 05 September 2016

References
1. N Fernando, SW Loke, W Rahayu, Mobile cloud computing: a survey[J]. Future Generation Comput. Syst **29**(1), 84–106 (2013)
2. AN Khan, MLM Kiah, SU Khan, SA Madani, Towards secure mobile cloud computing: a survey[J]. Future Generation Comput. Syst **29**(5), 1278–1299 (2013)
3. D Zissis, D Lekkas, Addressing cloud computing security issues[J]. Future Generation Comput. Syst **28**(3), 583–592 (2012)
4. S Subashini, V Kavitha, A survey on security issues in service delivery models of cloud computing[J]. J. Netw. Comput. Appl **34**(1), 1–11 (2011)
5. R Buyya, CS Yeo, S Venugopal, J Broberg, L Brandic, Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility[J]. Future Generation Comput. Syst **25**(6), 599–616 (2009)
6. Z Guan, T Yang, X Du, Achieving secure and efficient data access control for cloud-integrated body sensor networks[J]. Int. J. Distributed Sensor Netw **2015**, 142 (2015)
7. F Li, Y Rahulamathavan, M Conti, M Rajarajan, Robust access control framework for mobile cloud computing network[J]. Comput. Communicat **68**, 61–72 (2015)
8. YA Younis, K Kifayat, M Merabti, An access control model for cloud computing[J]. J. Inf. Secur. Appl **19**(1), 45–60 (2014)

9. Alizadeh M, Abolfazli S, Zamani M, et al. Authentication in mobile cloud computing: A survey[J]. J. Netw. Comput. Appl **29**(1), 84-106 (2016)
10. Lin H, Xu L, Mu Y, et al. A reliable recommendation and privacy-preserving based cross-layer reputation mechanism for mobile cloud computing[J]. Future. Generation. Comput. Syst **52**(C), 125-136 (2014)
11. Z Liu, Z Cao, D Wong, White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures[J]. Inf. Forensics Secur. IEEE Trans **8**(1), 76–88 (2013)
12. A Sahai, B Waters, *Fuzzy identity-based encryption[M]//Advances in Cryptology–EUROCRYPT 2005* (Springer, Berlin Heidelberg, 2005), pp. 457–473
13. Goyal V, Pandey O, Sahai A, Waters B. *Attribute-based encryption for fine-grained access control of encrypted data[C]// ACM Conference on Computer and Communications Security, CCS 2006*. (Alexandria, 2006), p. 89-98
14. Bethencourt J, Sahai A, Waters B. *Ciphertext-policy attribute-based encryption[C]//Security and Privacy, 2007*. SP'07. IEEE Symposium on. (Oakland, IEEE, 2007), p. 321-334
15. MJ Hinek, S Jiang, R Safavi-Naini, SF Shahandashti, Attribute-based encryption with key cloning protection[J]. IACR Cryptol. ePrint Arch **2008**, 478 (2008)
16. J Li, K Ren, K Kim, A2BE: accountable attribute-based encryption for abuse free access control[J]. IACR Cryptol. ePrint Arch **2009**, 118 (2009)
17. J Li, K Ren, B Zhu, Z Wan, *Privacy-aware attribute-based encryption with user accountability[M]//Information Security* (Springer, Berlin Heidelberg, 2009), pp. 347–362
18. MJ Hinek, S Jiang, R Safavi-Naini, SF Shahandashti, Attribute-based encryption without key cloning[J]. Int. J. Appl. Cryptography **2**(3), 250–270 (2012)
19. Chen C, Anada H, Kawamoto J, Sakurai K. *Hybrid encryption scheme using terminal fingerprint and its application to attribute-based encryption without key misuse[M]//Information and Communication Technology*. Springer International Publishing (Springer, Berlin Heidelberg, 2015), pp. 255-264.
20. B Chor, A Fiat, M Naor, *Tracing traitors[C]//Advances in cryptology—CRYPTO'94* (Springer, Berlin Heidelberg, 1994), pp. 257–270
21. S Mitsunari, R Sakai, M Kasahara, A new traitor tracing[J]. IEICE Trans. Fundamentals Electron. Commun. Comput. Sci **85**(2), 481–484 (2002)
22. R Safavi-Naini, Y Wang, *Sequential traitor tracing[C]//Advances in Cryptology—CRYPTO 2000* (Springer, Berlin Heidelberg, 2000), pp. 316–332
23. H Chabanne, DH Phan, D Pointcheval, *Public traceability in traitor tracing schemes[M]//Advances in Cryptology–EUROCRYPT 2005* (Springer, Berlin Heidelberg, 2005), pp. 542–558
24. D Boneh, A Sahai, B Waters, *Fully collusion resistant traitor tracing with short ciphertexts and private keys[M]//Advances in Cryptology-EUROCRYPT 2006* (Springer, Berlin Heidelberg, 2006), pp. 573–592
25. H Ma, G Zeng, Z Wang, J Xu, Fully secure multi-authority attribute-based traitor tracing[J]. J. Comput. Inf. Syst **9**(7), 2793–2800 (2013)
26. Z Liu, Z Cao, DS Wong, Traceable CP-ABE: how to trace decryption devices found in the wild[J]. Inf. Forensics Secur. IEEE Trans **10**(1), 55–68 (2015)
27. J Ning, X Dong, Z Cao, L Wei, X Lin, White-box traceable ciphertext-policy attribute-based encryption supporting flexible attributes[J]. Inf. Forensics Secur. IEEE Trans **10**(6), 1274–1288 (2015)
28. J Katz, D Schröder, *Tracing insider attacks in the context of predicate encryption schemes*, 2011
29. Ning, Jianting, et al. *Accountable authority ciphertext-policy attribute-based encryption with white-box traceability and public auditing in the cloud.* Computer Security – ESORICS 2015. Springer International Publishing (Springer, Berlin Heidelberg, 2015)
30. S Luo, J Hu, Z Chen, *Ciphertext policy attribute-based proxy re-encryption[M]// Information and Communications Security* (Springer, Berlin Heidelberg, 2010), pp. 401–415
31. Lai J, Deng R H, Yang Y, Weng J. *Adaptable ciphertext-policy attribute-based encryption[M]//Pairing-Based Cryptography–Pairing 2013*. Springer International Publishing (Springer, Berlin Heidelberg, 2013), pp. 199-214.
32. Liang K, Fang L, Susilo W, Wong DS. *A Ciphertext-Policy Attribute-Based Proxy Re-encryption with Chosen-Ciphertext Security[C]// International Conference on Intelligent NETWORKING and Collaborative Systems*. (Xi'an, IEEE, 2013), pp. 552-559
33. Liang K, Au M H, Susilo W, Wong D S, Yang G, Yu Y. *An adaptively CCA-secure ciphertext-policy attribute-based proxy re-encryption for cloud data sharing[M]//Information Security Practice and Experience*. Springer International Publishing (Springer, Berlin Heidelberg, 2014), pp. 448-461