# Accountable Authority Ciphertext-Policy Attribute-Based Encryption with White-Box Traceability and Public Auditing in the Cloud

Jianting Ning[1], Xiaolei Dong[2(✉)], Zhenfu Cao[2(✉)], and Lifei Wei[3]

[1] Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
jtning@sjtu.edu.cn

[2] Shanghai Key Lab for Trustworthy Computing, East China Normal University, Shanghai 200062, China
{dongxiaolei,zfcao}@sei.ecnu.edu.cn

[3] College of Information Technology, Shanghai Ocean University, Shanghai 201306, China
Lfwei@shou.edu.cn

**Abstract.** As a sophisticated mechanism for secure fine-grained access control, ciphertext-policy attribute-based encryption (CP-ABE) is a highly promising solution for commercial applications such as cloud computing. However, there still exists one major issue awaiting to be solved, that is, the prevention of key abuse. Most of the existing CP-ABE systems missed this critical functionality, hindering the wide utilization and commercial application of CP-ABE systems to date. In this paper, we address two practical problems about the key abuse of CP-ABE: (1) The key escrow problem of the semi-trusted authority; and, (2) The malicious key delegation problem of the users. For the semi-trusted authority, its misbehavior (i.e., illegal key (re-)distribution) should be caught and prosecuted. And for a user, his/her malicious behavior (i.e., illegal key sharing) need be traced. We affirmatively solve these two key abuse problems by proposing the first accountable authority CP-ABE with white-box traceability that supports policies expressed in any monotone access structures. Moreover, we provide an auditor to judge publicly whether a suspected user is guilty or is framed by the authority.

**Keywords:** Attribute-based encryption · Ciphertext-policy · Key abuse · White-box traceablity · Public auditing

## 1 Introduction

As a new commercial and exciting paradigm, cloud computing has attracted much attention from both industrial and academic world. Due to the advantage of cloud computing, plenty of enterprises and individuals can share and outsource their data to cloud servers instead of building and maintaining data centers of their own, and themselves or other authorized users can access the outsorced data anywhere

© Springer International Publishing Switzerland 2015
G. Pernul et al. (Eds.): ESORICS 2015, Part II, LNCS 9327, pp. 270–289, 2015.
DOI: 10.1007/978-3-319-24177-7_14

and anytime [1]. Despite lots of benefits provided by cloud computing, the concerns on data security are probably the main obstacles hindering the wide usage of cloud services. To address the data security concerns, encryption has been applied on the data of enterprises and individuals before outsourcing. Nevertheless, in some practical applications of cloud computing, data is often shared with some potential users without knowing who will receive it, thus a fine-grained access control over data is desired. Attribute-Based Encryption (ABE, [13]) is a promising approach to protect the confidentiality of sensitive data and express fine-grained access control for cloud computing. In a CP-ABE system, enterprises and individuals can specify access policies over attributes that the potential users possess. And the data customers whose attributes satisfy the specified access policy can decrypt successfully and get access to the outsourced data.

**A Motivating Story.** Consider a company employs a cloud storage system to outsource its data after encrypting the data under some access policies. Each employee is assigned with several attributes (such as "manager", "engineer", etc.). And those whose attributes satisfy the access policy over the outsourced data could decrypt the ciphertext and get access to the sensitive data stored in the cloud. As a versatile one-to-many encryption mechanism, CP-ABE system is quite suitable in this cloud storage scenario. If it happens to exist an employee from the company's competitor who is not authorized but could get access to the sensitive data stored in the cloud, as such, the company will suffer severe financial loss. Then, who leaks the decryption key to him? In addition, if an employee from the company named Bob is traced as the traitor (who leaks the decryption key) but claims to be innocent and framed by the system, then how to judge whether Bob is indeed innocent or not? Does Bob have an opportunity to argue for himself?

The problems, as described above, are the main obstacles when CP-ABE is implemented in cloud storage service. In a CP-ABE system, a user's decryption key is issued by a trusted authority according to the attributes the user possesses. The authority is able to generate and (re-)distribute decryption keys for any user without any risk of being caught and confronted in a court of law. Thus the security of a CP-ABE system relies heavily on trusting the authority. It is actually the key escrow problem in CP-ABE. One approach to reduce this trust is to employ multiple authorities [8,16,19]. However, this approach inevitably causes additional communication and infrastructure cost, and the problem of collusion among collaborating authorities remains. It is better to adopt the accountable authority approach to mitigate the key escrow problem in CP-ABE. The problem described above is the key abuse problem of authority. There exists another kind of key abuse problem: the key abuse problem of users. In a CP-ABE system, the decryption keys are defined over sets of attributes shared by multiple users. The misbehavior users may illegally share their decryption keys with others for profits without being detected. It is actually the malicious key delegation problem. It is necessary to trace the malicious users who leak their decryption keys illegally. Moreover, if a user is traced to be malicious (for leaking the decryption key) but claims to be innocent and framed by the system, it is necessary to

enable an auditor to judge whether the user is indeed innocent or is framed by the system.

### 1.1 Our Contribution

In this paper, we address the key abuse and the auditing problems of CP-ABE and affirmatively solve these by proposing an accountable authority CP-ABE system with white-box traceability and public auditing. To the best of our knowledge, this is the first CP-ABE scheme that supports the following properties: traceability of malicious users, accountable authority, almost no storage for tracing, public auditing and high expressiveness (i.e. supporting access policies expressed in any monotone access structures). Also, we prove that our new system is fully secure in the standard model.

We solve the obstacles of CP-ABE implementation in cloud storage scenario as follows:

1. Traceability of malicious users. Anyone who may leak their decryption keys to others for profits can be traced.
2. Accountable authority. The semi-trusted authority could be caught if it illegally generates and distributes legitimate keys to any unauthorized users.
3. Public auditing. We provide an auditor to judge whether a suspected user (for leaking his/her decryption key) is guilty or is framed by the authority. In addition, the auditability of our system is public, that is, anyone can run the `Audit` algorithm to make a judgement with no additional secret needed.
4. Almost no storage for tracing. We use a Paillier-style encryption as an extractable commitment in tracing the malicious users. And we do not need to maintain an identity table of users for tracing as used in [21]. As a result, we need almost no storage for tracing.

Table 1 gives the comparison between our work and some other related work.

**Table 1.** Comparison with other related work

| | [18] | [17] | [21] | [20] | [22] | Ours |
|---|---|---|---|---|---|---|
| Traceability of malicious users | × | × | × | √ | √ | √ |
| Accountable authority | √ | × | × | × | × | √ |
| Storage for tracing[a] | *none* | *none* | *linear* | *none* | *constant* | *none* |
| Supporting any monotone access structures | × | × | √ | √ | √ | √ |
| Public auditing | × | × | × | × | × | √ |
| Fully secure | × | × | √ | √ | × | √ |
| Standard model | × | √ | √ | √ | √ | √ |

[a] In [17,18,20] and this paper, the systems need almost no storage for tracing, for simplicity, we use *none* stands for almost no storage for tracing.

### 1.2 Our Technique

In this subsection, we briefly introduce the main idea we utilize to realize the properties of traceability of malicious users, accountable authority and public auditing before giving the full details in Sect. 4.

To trace malicious users who may leak their decryption keys to others for profits, we use a Paillier-style encryption as an extractable commitment to achieve white-box traceability. Specifically, we use a Paillier-style extractable commitment to make a commitment to a user's identity when the user queries for his decryption key. The commitment is further inserted into the user's decryption key as a necessary part for successful decryption. Due to the hiding and binding properties of the Paillier-style extractable commitment, the user does not know what is inserted into his decryption key and even cannot change the identity insert into his decryption key. When it comes to the `Trace` algorithm, the algorithm uses a trapdoor for the commitment to recover the identity of the user from his decryption key. Note that the decryption key needs to take a *key sanity check* algorithm to see whether it is *well-formed* or not prior to the tracing step. Take the advantage of the Paillier-style extractable commitment, we do not have to maintain the identity table as used in [21], as a result, we need almost no storage for tracing.

To achieve accountable authority, the main idea is to let the user's decryption key be jointly determined by both of the authority and the user himself, hence the authority does not have complete control over the decryption key. We let a user get his decryption key $sk$ corresponding to his attributes and identity from the authority using a secure key generation protocol. The protocol allows the user to obtain a decryption key $sk$ for his attributes and identity without letting the authority know which key he obtained. Now if the authority (re-)distribute a decryption key $\tilde{sk}$ (corresponding to a user's attributes and identity) for malicious usage, with all but negligible probability, it will be different from the key $sk$ which the user obtained. Hence the key pair $(sk, \tilde{sk})$ is a cryptographic proof of malicious behavior of the authority.

Furthermore, the difference between the user's decryption key $sk$ and the decryption key $\tilde{sk}$ (re-)distributed by the authority allows the auditor to judge publicly whether the malicious user is guilty or is framed by the system. And note that the auditor is assumed to be fair and credible.

### 1.3 Related Work

Attribute-Based Encryption, first introduced by Sahai and Waters [27], generalizes the notion of fuzzy Identity-Based Encryption (IBE) [6,28]. Goyal et al. [13] formalized two complementary forms of Attribute-Based Encryption (ABE): Key-Policy Attribute-Based Encryption (KP-ABE) and Ciphertext-Policy Attribute-Based Encryption (CP-ABE). In a CP-ABE system, every user's decryption key is associated with a set of attributes she/he possesses, and every ciphertext is associated with an access policy defined over attributes. KP-ABE is reversed in that every ciphertext is associated with a set of attributes and every

user's decryption key is associated with an access policy. ABE (especially CP-ABE) is envisioned as a highly promising public key primitive for implementing scalable and fine-grained access control over encrypted data, and has attracted much attention in the research community. A series of ABE (including CP-ABE and KP-ABE) systems have been proposed [4,11,14,15,20–22,24–26,29], aiming at better efficiency, expressiveness or security.

Li et al. first introduced the notion of accountable CP-ABE [18] to prevent illegal key sharing among colluding users. Then a user accountable multi-authority CP-ABE scheme was proposed in [17] which only supported AND gates with wildcard. White-box [21] and black-box [20] traceability CP-ABE systems which supported policies expressed in any monotone access structures were later proposed by Liu et al. Recently, Ning et al. [22] proposed a practical large universe CP-ABE system with white-box traceability. Deng et al. [9] provided a tracing mechanism of CP-ABE to find the leaked access credentials in cloud storage systems. Unfortunately, the above work either only support less expressive access policy, or do not consider the misbehavior of the authority, or do not address the auditing issue.

### 1.4 Organization

Section 2 introduces the background, including the notation, the access policy, the linear secret sharing scheme, the composite order bilinear groups, the assumptions and the zero-knowledge proof of knowledge of discrete log. Section 3 gives the formal definition of accountable authority CP-ABE with white-box traceability and public auditing (AAT-CP-ABE) and its security model. Section 4 presents the construction of our AAT-CP-ABE system as well as the security proof. Finally, Sect. 5 presents a brief conclusion and foresees our future work.

## 2 Background

### 2.1 Notation

We define $[l] = \{1, 2, ..., l\}$ for $l \in \mathbb{N}$. We denote by $s \xleftarrow{R} S$ the fact that $s$ is picked uniformly at random from the finite set $S$. By PPT we denote probabilistic polynomial-time. We denote $(v_1, v_2, ..., v_n)$ be a row vector and $(v_1, v_2, ..., v_n)^{\perp}$ be a column vector. By $v_i$ we denote the $i$-th element in a vector $\boldsymbol{v}$. And by $M\boldsymbol{v}$ we denote the product of matrix $M$ with vector $\boldsymbol{v}$. We denote $\mathbb{Z}_p^{l \times n}$ be the set of matrices of size $l \times n$ with elements in $\mathbb{Z}_N$. The set of column vectors of length $n$ (i.e. $\mathbb{Z}_N^{n \times 1}$) are the two special subsets and the set of row vectors of length $n$ (i.e. $\mathbb{Z}_N^{1 \times n}$).

### 2.2 Access Policy

**Definition 1.** *(Access Structure [2]): Let $S$ be the attribute universe. A collection (respectively, monotone collection) $\mathbb{A} \subseteq 2^S$ of non-empty sets of attributes is*

*an access structure (respectively, monotone access structure) on $S$. A collection $\mathbb{A} \subseteq 2^S$ is called monotone if $\forall B, C \in \mathbb{A} : if\ B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.*

For CP-ABE, if a user of the system possess an authorized set of attributes then he can decrypt the ciphertext. Otherwise, the set he possed is unauthorized and he can't get any information from ciphertext. In our construction, we restrict our attention to monotone access structure.

### 2.3 Linear Secret-Sharing Schemes

**Definition 2.** *(Linear Secret-Sharing Schemes (LSSS) [2,22]). Let $S$ denote the attribute universe and $p$ denote a prime. A secret-sharing scheme $\prod$ with domain of secrets $\mathbb{Z}_p$ realizing access structure on $S$ in called linear (over $\mathbb{Z}_p$) if*

1. *The shares of a secret $s \in \mathbb{Z}_p$ for each attribute form a vector over $\mathbb{Z}_p$.*
2. *For each access structure $\mathbb{A}$ on $S$, there exists a matrix $M$ with $l$ rows and $n$ columns called the share-generating matrix for $\prod$. For $i = 1, ..., l$, we define a function $\rho$ labels row $i$ of $M$ with attribute $\rho(i)$ from the attribute universe $S$. When we consider the column vector $\boldsymbol{v} = (s, r_2, ..., r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared and $r_2, ..., r_n \in \mathbb{Z}_p$ are randomly chosen. Then $M\boldsymbol{v} \in \mathbb{Z}_p^{l \times 1}$ is the vector of $l$ shares of the secret $s$ according to $\prod$. The share $(M\boldsymbol{v})_j$ "belongs" to attribute $\rho(j)$, where $j \in [l]$.*

As shown in [2], every linear secret-sharing scheme according to the above definition also enjoys the linear reconstruction property, defined as follows: we suppose that $\prod$ is an LSSS for the access structure $\mathbb{A}$, $S' \in \mathbb{A}$ is an authorized set and let $I \subset \{1, 2, ..., l\}$ be defined as $I = \{i \in [l] \wedge \rho(i) \in S'\}$. Then, there exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that for any valid shares $\{\lambda_i = (M\boldsymbol{v})_i\}_{i \in I}$ of a secret $s$ according to $\prod$, then $\sum_{i \in I} \omega_i \lambda_i = s$. Additionally, it is shown in [2] that these constants $\{\omega_i\}_{i \in I}$ can be found in time polynomial in the size of the share-generating matrix $M$. On the other hand, for any unauthorized set $S''$, no such constants $\{\omega_i\}$ exist.

Note that if we encode the access structure as a monotonic Boolean formula over attributes, there exists a generic algorithm by which we can generate the corresponding access policy in polynomial time [2].

In our construction, an LSSS matrix $(M, \rho)$ will be used to express an access policy associated to a ciphertext.

### 2.4 Composite Order Bilinear Groups

Composite order bilinear groups are widely used in IBE and ABE systems, which are first introduced in [7]. We let $\mathcal{G}$ denote a group generator, which takes a security parameter $\lambda$ as input and outputs a description of a bilinear group $G$. We define the output of $\mathcal{G}$ as $(p_1, p_2, p_3, G, G_T, e)$, where $p_1, p_2, p_3$ are distinct primes, $G$ and $G_T$ are cyclic groups of order $N = p_1 p_2 p_3$, and $e : G^2 \rightarrow G_T$ is a map such that:

1. Bilinearity: $\forall u, v \in G$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $\exists g \in G$ such that $e(g, g)$ has order $N$ in $G_T$.

We assume that group operations in $G$ and $G_T$ as well as the bilinear map $e$ are computable in polynomial time with respect to $\lambda$. We refer to $G$ as the *source group* and $G_T$ as the *target group*, and assume the group descriptions of $G$ and $G_T$ include a generator of each group. Let $G_{p_1}$, $G_{p_2}$, and $G_{p_3}$ be the subgroups of order $p_1$, $p_2$, and $p_3$ in $G$, respectively. Note that these subgroups are "orthogonal" to each other under the bilinear map $e$: for any $u_i \in G_{p_i}$ and $u_j \in G_{p_j}$ where $i \neq j$, $e(u_i, u_j) = 1$. Any element $E_N \in G$ can (uniquely) be expressed as $g_1^{r_1} g_2^{r_2} g_3^{r_3}$ for some values $r_1, r_2, r_3 \in \mathbb{Z}_N$, where $g_1, g_2, g_3$ are the generators of $G_{p_1}, G_{p_2}, G_{p_3}$ respectively. And we will refer to $g_1^{r_1}, g_2^{r_2}, g_3^{r_3}$ as the "$G_{p_1}$ part of $E_N$", "$G_{p_2}$ part of $E_N$" and "$G_{p_3}$ part of $E_N$", respectively. Assume $G_{p_1 p_2}$ be the subgroups of order $p_1 p_2$ in $G$. Similarly, any element $E_{p_1 p_2} \in G p_1 p_2$ can be expressed as the product of an element from $G_{p_1}$ and an element from $G_{p_2}$.

### 2.5 Complexity Assumptions

**Assumption 1.** *(Subgroup Decision Problem for 3 Primes): [14] Given a group generator $\mathcal{G}$, define the following distribution:*

$$\begin{gathered}\mathbb{G} = (N = p_1 p_2 p_3, G, G_T, e) \xleftarrow{R} \mathcal{G},\\ g \xleftarrow{R} G_{p_1}, X_3 \xleftarrow{R} G_{p_3},\\ D = (\mathbb{G}, g, X_3),\\ T_1 \xleftarrow{R} G_{p_1 p_2}, T_2 \xleftarrow{R} G_{p_1}.\end{gathered}$$

The advantage of an algorithm $\mathcal{A}$ in breaking this assumption is defined to be: $Adv1_{\mathcal{G},\mathcal{A}}(\lambda) = |Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]|$.

**Definition 3.** *We say that $\mathcal{G}$ satisfies Assumption 1 if $Adv1_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of $\lambda$ for any polynomial time algorithm $\mathcal{A}$.*

**Assumption 2.** *[14] Given a group generator $\mathcal{G}$, define the following distribution:*

$$\begin{gathered}\mathbb{G} = (N = p_1 p_2 p_3, G, G_T, e) \xleftarrow{R} \mathcal{G},\\ g, X_1 \xleftarrow{R} G_{p_1}, X_2, Y_2 \xleftarrow{R} G_{p_2}, X_3, Y_3 \xleftarrow{R} G_{p_3}\\ D = (\mathbb{G}, g, X_1 X_2, X_3, Y_2 Y_3),\\ T_1 \xleftarrow{R} G, T_2 \xleftarrow{R} G_{p_1 p_3}.\end{gathered}$$

The advantage of an algorithm $\mathcal{A}$ in breaking this assumption is defined to be: $Adv2_{\mathcal{G},\mathcal{A}}(\lambda) = |Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]|$.

**Definition 4.** *We say that $\mathcal{G}$ satisfies Assumption 2 if $Adv2_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of $\lambda$ for any polynomial time algorithm $\mathcal{A}$.*

**Assumption 3.** *[14] Given a group generator $\mathcal{G}$, define the following distribution:*

$$\mathbb{G} = (N = p_1p_2p_3, G, G_T, e) \xleftarrow{R} \mathcal{G}, \alpha, s \xleftarrow{R} \mathbb{Z}_N,$$
$$g \xleftarrow{R} G_{p_1}, X_2, Y_2, Z_2 \xleftarrow{R} G_{p_2}, X_3 \xleftarrow{R} G_{p_3}$$
$$D = (\mathbb{G}, g, g^{\alpha}X_2, X_3, g^{s}Y_2, Z_2),$$
$$T_1 = e(g,g)^{\alpha s}, T_2 \xleftarrow{R} G_T.$$

The advantage of an algorithm $\mathcal{A}$ in breaking this assumption is defined to be: $Adv3_{\mathcal{G},\mathcal{A}}(\lambda) = |Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]|$.

**Definition 5.** *We say that $\mathcal{G}$ satisfies Assumption 3 if $Adv3_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of $\lambda$ for any polynomial time algorithm $\mathcal{A}$.*

**Assumption 4.** *(l-SDH assumption [5, 10]): Let $\mathbb{G}$ be a bilinear group of prime order $p$ and $g$ be a generator of $\mathbb{G}$, the l-Strong Diffie-Hellman (l-SDH) problem in $\mathbb{G}$ is defined as follows: given a $(l+1)$-tuple $(g, g^x, g^{x^2}, ..., g^{x^l})$ as inputs, output a pair $(c, g^{1/(c+x)}) \in \mathbb{Z}_p \times \mathbb{G}$. An algorithm $\mathcal{A}$ has advantage $\epsilon$ in solving l-SDH in $\mathbb{G}$ if $Pr[\mathcal{A}(g, g^x, g^{x^2}, ..., g^{x^l}) = (c, g^{1/(c+x)})] \geq \epsilon$, where the probability is over the random choice of $x$ in $\mathbb{Z}_p^*$ and the random bits consumed by $\mathcal{A}$.*

**Definition 6.** *We say that the $(l, t, \epsilon)$-SDH assumption holds in $\mathbb{G}$ if no $t$-time algorithm has advantage at least in solving the l-SDH problem in $\mathbb{G}$.*

### 2.6 Zero-Knowledge Proof of Knowledge of Discrete Log

Informally, a zero-knowledge proof of knowledge (ZK-POK) of discrete log protocol enables a prover to prove that it possesses the discrete log $t$ of a given group element $T$ in question to a verifier.

A ZK-POK protocol has two distinct properties: the zero-knowledge property and the proof of knowledge property. The property of zero-knowledge implies that there exists a simulator $S$ which is able to simulate the view of a verifier in the protocol without being given the witness as input. The proof of knowledge property implies there exists a knowledge-extractor $Ext$ which interacts with the prover and extracts the witness using rewinding techniques [10]. We refer the reader to [3] for more details about ZK-POK.

## 3 Accountable Authority CP-ABE with White-Box Traceability and Public Auditing

### 3.1 Definition

An Accountable Authority CP-ABE with White-Box Traceability and Public Auditing (AAT-CP-ABE) is a CP-ABE system which could hold the misbehaved authority accountable, trace the malicious user by his/her decryption key and judge whether the suspected a user is indeed innocent or not. An AAT-CP-ABE system consists of seven algorithms as follows:

- $\texttt{Setup}(1^\lambda, \mathcal{U}) \rightarrow (pp, msk)$: The algorithm takes as input a security parameter $\lambda \in \mathbb{N}$ encoded in unary and the attribute universe description $\mathcal{U}$. It outputs the public parameters $pp$ and the master secret key $msk$.
- $\texttt{KeyGen}(pp, msk, id, S) \rightarrow sk_{id,S}$: This is an interactive protocol between the authority $AT$ and a user $U$. The public parameters $pp$ and a set of attributes $S$ for a user with identity $id$ are the common input to the $AT$ and $U$. The master secret key $msk$ is the private input to the $AT$. Additionally, the $AT$ and $U$ may use a sequence of random coin tosses as private input. At the end of the protocol, $U$ is issued a secret key $sk_{id,S}$ corresponding to $S$.
- $\texttt{Encrypt}(pp, m, \mathbb{A}) \rightarrow ct$: The encryption algorithm takes as input the public parameters $pp$, a plaintext message $m$, and an access structure $\mathbb{A}$ over the universe of attributes. It outputs the ciphertext $ct$[1].
- $\texttt{Decrypt}(pp, sk_{id,S}, ct) \rightarrow m$ or $\perp$: The decryption algorithm takes as input the public parameters $pp$, a secret key $sk_{id,S}$, and a ciphertext $ct$. If the set of attributes of the private key satisfies the access structure of the ciphertext, the algorithm outputs the plaintext $m$. Otherwise, it outputs $\perp$.
- $\texttt{KeySanityCheck}(pp, sk) \rightarrow 1$ or 0: The key sanity check algorithm takes as input the public parameters $pp$ and a secret key $sk$. If $sk$ passes the key sanity check, it outputs 1. Otherwise, it outputs 0. The key sanity check is a deterministic algorithm [10,12], which is used to guarantee the secret key to be well-formed in the decryption process.
- $\texttt{Trace}(pp, msk, sk) \rightarrow id$ or $\top$: The tracing algorithm takes as input the public parameters $pp$, the master secret key $msk$ and a secret key $sk$. The algorithm first checks whether $sk$ is *well-formed* or not so as to determine whether $sk$ needs to be traced. A secret key $sk$ is defined as *well-formed* which means that $\texttt{KeySanityCheck}(pp, sk) \rightarrow 1$. If $sk$ is well-formed, the system extracts the identity $id$ from $sk$. Then it outputs an identity $id$ with which the $sk$ associates. Otherwise, it outputs a special symbol $\top$ indicates that $sk$ does not need to be traced.
- $\texttt{Audit}(pp, sk_{id}, sk_{id}^*) \rightarrow$ *guilty* or *innocent*. This is an interactive protocol between a user $U$ and a public auditor $PA$. It judges whether a user is *guilty* or *innocent*.

### 3.2 Security

An AAT-CP-ABE system is deemed secure if the following three requirements are satisfied. First, it must satisfy the standard semantic security notion for CP-ABE system: ciphertext indistinguishability under chosen plaintext attacks (IND-CPA). Second, it is intractable for the authority to create a decryption key such that the `Trace` algorithm outputs a user and the `Audit` algorithm outputs the user is guilty. Finally, it is infeasible for a user to create a decryption key such that the `Audit` algorithm implicates the user is innocent. To define security for AAT-CP-ABE system satisfies the above three requirements, we define the following three games, respectively.

[1] We assume that $\mathbb{A}$ is implicitly in the ciphertext $ct$.

**The IND-CPA game**. The IND-CPA game for AAT-CP-ABE system is similar to that of the CP-ABE system [15], excepting every key query is companied with an explicit identity. The game proceeds as follows:

- **Setup:** The challenger runs the $\texttt{Setup}(1^\lambda, \mathcal{U})$ algorithm and sends the public parameters $pp$ to the attacker.
- **Query Phase 1:** In this phase the attacker can adaptively query the challenger for secret keys corresponding to sets of attributes $(id_1, S_1), (id_2, S_2), ..., (id_{Q_1}, S_{Q_1})$. For each $(id_i, S_i)$ the challenger calls $\texttt{KeyGen}(pp, msk, id, S_i) \rightarrow sk_{id,S_i}$ and sends $sk_{id,S_i}$ to the attacker.
- **Challenge:** The attacker declares two equal length messages $m_0$ and $m_1$ and an access structure $\mathbb{A}^*$. Note that this access structure cannot be satisfied by any of the queried attributes sets $(id_1, S_1), (id_2, S_2), ..., (id_{Q_1}, S_{Q_1})$. The challenge flips a random coin $\delta \in \{0, 1\}$ and calls $\texttt{Encrypt}(pp, m_\delta, \mathbb{A}^*) \rightarrow ct$. It sends $ct$ to the attacker.
- **Query Phase 2:** The attacker adaptively queries the challenger for the secret keys corresponding to sets of attributes $(id_{Q_1+1}, S_{Q_1+1}), ..., (id_Q, S_Q)$ with the added restriction that none of these satisfy $\mathbb{A}^*$. For each $(id_i, S_i)$ the challenger calls $\texttt{KeyGen}(pp, msk, id, S_i) \rightarrow sk_{id,S_i}$ and sends $sk_{id,S_i}$ to the attacker.
- **Guess:** The attacker outputs a guess $\delta' \in \{0, 1\}$ for $\delta$.

An attacker's advantage in this game is defined to be $Adv = |\Pr[\delta' = \delta] - 1/2|$.

**Definition 7.** *An AAT-CP-ABE system is fully secure if all probabilistic polynomial time (PPT) attackers have at most a negligible advantage in the above game.*

**The DishonestAuthority Game.** The DishonestAuthority game for the AAT-CP-ABE system is defined as follows. The intuition behind this game is that an adversarial authority attempts to create a decryption key which will frame a user. It is described by a game between a challenger and an attacker.

- **Setup:** The attacker (acting as a malicious authority) generates public parameters $pp$, and sends $pp$, a user's $(id, S)$ to the challenger. The challenger runs a sanity check on $pp$ and $(id, S)$ aborts if the check fails.
- **Key Generation:** The attacker and the challenger engage in the key generation protocol $\texttt{KeyGen}$ to generate a decryption key $sk^*_{id}$ corresponding to the user's $id$ and $S$. The challenger gets the decryption key $sk^*_{id}$ as input and runs a sanity check on it to ensure that it is well-formed. It aborts if the check fails.
- **Output:** The attacker outputs a decryption key $sk^*_{id}$ and succeeds if $\texttt{Trace}(pp, msk, sk^*_{id}) \rightarrow id$, and $\texttt{Audit}(pp, sk_{id}, sk^*_{id}) \rightarrow guilty$.

The attacker's advantage in this game is defined to be $Adv = |\Pr[\mathcal{A}\ succeeds]|$ where the probability is taken over the random coins of $\texttt{Trace}$, $\texttt{Audit}$, the attacker and the challenger.

**Definition 8.** *An AAT-CP-ABE system is DishonestAuthority secure if all PPT attackers have at most a negligible advantage in the above security game.*

**The DishonestUser Game.** The DishonestUser game for the AAT-CP-ABE system is defined as follows. The intuition behind this game is that a malicious user attempts to create new decryption key which will frame the authority. It is described by a game between a challenger and an attacker.

- **Setup:** The challenger runs the $\texttt{Setup}(1^\lambda, \mathcal{U})$ algorithm and sends the public parameters $pp$ to the attacker.
- **Key Query:** The attacker submits the sets of attributes $(id_1, S_1), ..., (id_q, S_q)$ to request the corresponding decryption keys. The challenger calls $\texttt{KeyGen}(pp, msk, id, S_i) \rightarrow sk_{id,S_i}$ and returns $sk_{id,S_i}$ to the attacker.
- **Key Forgery:** The attacker will output a decryption key $sk_*$. If $\{\texttt{Trace}(pp, msk, sk_*) \neq \top$ and $\texttt{Trace}(pp, msk, sk_*) \notin \{id_1, ..., id_q\}\}$ *or* $\{\texttt{Trace}(pp, msk, sk_*) = id$ and $\texttt{Audit}(pp, sk_{id}, sk^*_{id}) \rightarrow innocent\}$, the attacker wins the game.

An attacker's advantage in this game is defined to be $Adv = |\Pr[\mathcal{A}\ succeeds]|$ where the probability is taken over the random coins of `Trace`, `Audit`, the attacker and the challenger.

**Definition 9.** *An AAT-CP-ABE system is fully traceable if all PPT attackers have at most a negligible advantage in the above security game.*

**The Key Sanity Check Game**. According to [23], the Key Sanity Check game for the AAT-CP-ABE system is defined as follows. It is described by the following game between an attacker and a simulator. On input a security parameter $1^\lambda$ ($\lambda \in \mathbb{N}$), a simulator invokes an attacker $\mathcal{A}$ on $1^\lambda$. $\mathcal{A}$ returns the public parameters $pp$, a ciphertext $ct$ and two different secret keys $sk_{id,S}$ and $\tilde{sk}_{id,S}$ corresponding to the same set of attributes $S$ for a user with identity $id$. $\mathcal{A}$ wins the game if

(1) $\texttt{KeySanityCheck}(pp, sk_{id,S}) \rightarrow 1$.
(2) $\texttt{KeySanityCheck}(pp, \tilde{sk}_{id,S}) \rightarrow 1$.
(3) $\texttt{Decrypt}(pp, sk_{id,S}, ct) \neq \perp$.
(4) $\texttt{Decrypt}(pp, \tilde{sk}_{id,S}, ct) \neq \perp$.
(5) $\texttt{Decrypt}(pp, sk_{id,S}, ct) \neq \texttt{Decrypt}(pp, \tilde{sk}_{id,S}, ct)$.

$\mathcal{A}$'s advantage in the above game is defined as $\Pr[\mathcal{A}\ wins]$. And it is easy to see that the intuition of "Key Sanity Check" is captured combining the notion captured in the above game and the related algorithms (`KeySanityCheck` and `Decrypt`) defined in this section [23].

## 4 Our System

### 4.1 Construction

- $\texttt{Setup}(\lambda, \mathcal{U}) \rightarrow (pp, msk)$: The algorithm calls the group generator $\mathcal{G}$ with $\lambda$ as input and gets a bilinear group $G$ of order $N = p_1p_2p_3$ (3 distinct primes), $G_{p_i}$ the subgroup of order $p_i$ in $G$, and $g, g_3$ the generator of the subgroup

$G_{p_1}, G_{p_3}$ respectively. It then chooses exponents $\alpha, a, \kappa, \mu \in \mathbb{Z}_N$ and a group element $v \in G_{p_1}$ randomly. For each attribute $i \in \mathcal{U}$, the algorithm chooses a random value $u_i \in \mathbb{Z}_N$. Also, the algorithm chooses two random primes $p$ and $q$ for which it holds $p \neq q$, $|p| = |q|$ and $gcd(pq, (p-1)(q-1)) = 1$, and then let $n = pq$, $\pi = lcm(p-1, q-1)$, $Q = \pi^{-1}\ mod\ n$ and $g_1 = (1+n)$. The public parameters are set to $pp = (N, n, g_1, v, g, g^a, g^\kappa, g^\mu, e(g,g)^\alpha, \{\mathcal{U}_i = g^{u_i}\}_{i\in\mathcal{U}})$. And the master secret key $msk$ is set to $msk = (p, q, \alpha, g_3)$.

– $\texttt{KeyGen}(pp, msk, id, S) \rightarrow sk_{id,S}$: The authority $AT$ and a user $U$ (with the identity $id$[2]) interact in the key generation protocol as follows.

1. $U$ first chooses $t \in \mathbb{Z}_N$ randomly and computes $R_U = g^t$. Next, it sends $g^t$, the identity $id$ and a set of attributes $S$ to $AT$. Then, it runs an interactive ZK-POK of the discrete log of $R_U$ with respect to $g$ with $AT$.
2. $AT$ first checks whether the ZK-POK is valid or not. If the check fails, $AT$ aborts the interaction. Otherwise, it chooses a random $c \in \mathbb{Z}_N$, a random $r \in \mathbb{Z}_n^*$ and random elements $R, R_0, R_0', \{R_i\}_{i\in S} \in G_{p_3}$. Then, it computes the primary secret key $sk_{pri}$ as follows:

$$\langle S,\ \overline{K} = g^{\frac{\alpha}{a+\overline{T}}}(g^t)^{\frac{\kappa}{a+\overline{T}}} v^c R,\ \overline{T} = g_1^{id} r^n\ mod\ n^2,$$

$$\overline{L} = g^c R_0,\ \overline{L'} = g^{ac} R_0',\ \{\overline{K_i} = \mathcal{U}_i^{(a+\overline{T})c} R_i\}_{i\in S}\rangle.$$

   And it sends $(c, sk_{pri})$ to $U$.
3. $U$ checks whether the following equalities hold or not:
   (1) $e(\overline{L'}, g) = e(\overline{L}, g^a) = e(g^a, (g)^c)$.
   (2) $e(\overline{K}, g^a g^{\overline{T}}) = e(g,g)^\alpha e(\overline{L'}(\overline{L})^{\overline{T}}, v) e(R_U, g^\kappa)$.
   (3) $\exists x \in S\ s.t.\ e(\mathcal{U}_x, \overline{L'}(\overline{L})^{\overline{T}}) = e(\overline{K_x}, g)$.
   If no, $U$ aborts the interaction. Otherwise, $U$ computes $t_{id} = \frac{c}{t}$ and sets his decryption key $sk_{id,S}$ as follows:

$$\langle S,\ K = \overline{K}(g^\mu)^{t_{id}},\ T = \overline{T},\ L = \overline{L},\ L' = \overline{L'},\ R_U,\ t_{id},\ \{K_i = \overline{K_i}\}_{i\in S}\rangle.$$

– $\texttt{Encrypt}(pp, m, (A, \rho)) \rightarrow ct$: The algorithm takes the access structure encoded in an LSSS policy[3], the public parameters $pp$ and a plaintext message $m$. The algorithm chooses $\overrightarrow{y} = (s, y_2, ..., y_n)^\perp \in \mathbb{Z}_N^{n\times 1}$ randomly, where $s$ is the random secret to be shared among the shares according to Subsect. 2.3. Then it chooses $r_j \in \mathbb{Z}_N$ for each row $A_j$ of $A$ randomly. The ciphertext $ct$ is set as follows:

$$\langle C = m \cdot e(g,g)^{\alpha s}, C_0 = g^s, C_1 = (g^a)^s, C_2 = (g^\kappa)^s, C_3 = (g^\mu)^s,$$

$$\{C_{j,1} = v^{A_j \overrightarrow{y}} \mathcal{U}_{\rho(j)}^{-r_j}, C_{j,2} = g^{r_j}\}_{j\in[l]}, (A, \rho)\rangle.$$

[2] We assume that the identity $id$ is an element in $\mathbb{Z}_n$. One can extend the construction to arbitrary identities in $\{0,1\}^*$ easily by adopting a collision-resistant hash $H : \{0,1\}^* \rightarrow \mathbb{Z}_n$.

[3] where $A$ is an $l \times n$ matrix and $\rho$ is a map from each row $A_j$ of $A$ to an attribute $\rho(j)$.

– $\texttt{Decrypt}(pp, sk_{id,S}, ct) \rightarrow m$ or $\perp$: The algorithm first parses the $sk_{id,S}$ to $(S, K, T, L, L', R_U, t_{id}, \{K_i\}_{i\in S})$ and $ct$ to $(C, C_0, C_1, C_2, C_3, \{C_{j,1}, C_{j,2}\}_{j\in[l]}, (A, \rho))$. The algorithm will output $\perp$ if the attribute set $S$ cannot satisfy the access structure $(A, \rho)$ of $ct$. Otherwise, the algorithm first computes constants $\omega_j \in \mathbb{Z}_N$ such that $\sum_{\rho(j)\in S} \omega_j A_j = (1, 0, ..., 0)$. It then computes:

$$D = e((C_0)^T C_1, K)(e(C_2, R_u)e(C_3, (g^T g^a)^{t_{id}}))^{-1}$$

$$E = \Pi_{\rho(j)\in S}(e(C_{j,1}, (L)^T L')e(C_{j,2}, K_{\rho(j)}))^{\omega_j}$$

$$F = D/E = e(g, g)^{\alpha s}, m = C/F$$

– $\texttt{KeySanityCheck}(pp, sk) \rightarrow 1$ or 0: The algorithm takes as input the public parameters $pp$ and a secret key $sk$. The secret key $sk$ passes the key sanity check if
(1) $sk$ is in the form of $(S, K, T, L, L', R_U, t_{id}, \{K_i\}_{i\in S})$ and $T \in \mathbb{Z}^*_{n^2}$, $K, L, L', R_U, \{K_i\}_{i\in S} \in G$.
(2) $e(L', g) = e(L, g^a)$.
(3) $e(K, g^a g^T) = e(g, g)^\alpha e(L'(L)^T, v)e(R_U, g^\kappa)e((g^a g^T)^{t_{id}}, g^\mu)$.
(4) $\exists x \in S$ *s.t.* $e(\mathcal{U}_x, L'(L)^T) = e(K_x, g)$.
If $sk$ passes the key sanity check, the algorithm outputs 1. Otherwise, it outputs 0.

– $\texttt{Trace}(pp, msk, sk) \rightarrow id$ or $\intercal$: If $\texttt{KeySanityCheck}(pp, sk) \rightarrow 0$, the algorithm outputs $\intercal$. Otherwise, it is a well-formed decryption key[4], and the algorithm will extract the identity $id$ from $T = g_1^{id} r^n \ mod\ n^2$ in $sk$ as follows: note that $Q = \pi^{-1} \ mod\ n$ and observe that $T^{\pi Q} = g_1^{id\cdot\pi Q} \cdot r^{n\cdot\pi Q} = g_1^{id} = 1 + id \cdot n \ mod\ n^2$. Thus, it recovers $id = \frac{((T)^{\pi Q} \ mod\ n^2) - 1}{n} \ mod\ n$ and outputs the identity $id$.

– $\texttt{Audit}(pp, sk_{id}, sk^*_{id}) \rightarrow$ *guilty* or *innocent*: Suppose a user $U$ (with identity $id$ and decryption key $sk_{id}$) is identified as a malicious user by the system (through the traced key $sk^*_{id}$), but claims to be innocent and framed by the system. $U$ will interact with the public auditor $PA$ in the following protocol.
(1) $U$ sends its decryption key $sk_{id}$ to $PA$. If $\texttt{KeySanityCheck}(pp, sk) \rightarrow 0$, $PA$ aborts. Otherwise, go to (2).
(2) $PA$ tests whether the equality $t_{id} = t^*_{id}$ hold or not. If no, it outputs *innocent* indicates that $U$ is innocent and is framed by the system. Otherwise, it outputs *guilty* indicates that $U$ is malicious and $sk^*_{id}$ is leaked by $U$.

### 4.2 IND-CPA Security

Since our construction of accountable authority traceable CP-ABE system is based on the CP-ABE system in [14], for simplicity, we will reduce the IND-CPA security proof of our construction to that of the system in [14]. We denote by $\Sigma_{cpabe}$, $\Sigma_{aatcpabe}$ the CP-ABE system in [14] and our system respectively.

[4] i.e. the decryption privilege of the key is described by attribute set $S_\tau = \{x | x \in S \wedge e(K_x, g) = e(\mathcal{U}_x, L'(L)^T) \neq 1\}$.

The security model of $\Sigma_{cpabe}$ in [14] is almost the same with the IND-CPA security model of our system $\Sigma_{aatcpabe}$ in Subsection in 3.2, excepting every key query is companied with an identity and the decryption key is jointly determined by a user and the authority.

**Lemma 1.** *[14] If Assumptions 1,2,3 hold, then the CP-ABE system $\Sigma_{cpabe}$ in [14] is secure.*

*(2) IND-CPA Security of our AAT-CP-ABE system:*

**Lemma 2.** *[14] If the CP-ABE system $\Sigma_{cpabe}$ in [14] is secure, then our AAT-CP-ABE system $\Sigma_{aatcpabe}$ in is secure in the IND-CPA security game of Subsect. 3.2.*

Due to space, we refer the reader to Appendix A for the proof of this lemma.

**Theorem 1.** *If Assumptions 1,2,3 hold, then our AAT-CP-ABE system $\Sigma_{aatcpabe}$ is secure.*

**Proof.** It follows directly from Lemmas 1 and 2.

### 4.3 DishonestAuthority Security

**Theorem 2.** *If computing discrete log is hard in $G_{p_1}$, the advantage of an adversary in the DishonestAuthority game is negligible for our AAT-CP-ABE system.*

Due to space, we refer the reader to Appendix B for the proof of this theorem.

### 4.4 DishonestUser Security

In this subsection, we prove the DishonestUser secure of our AAT-CP-ABE system based on $q$-SDH assumption and Assumption 2. We adopt a similar method from [5] and [21].

**Theorem 3.** *If $q$-SDH assumption and Assumption 2 hold, then our AAT-CP-ABE system is DishonestUser secure provided that $q' < q$.*

Due to space, we refer the reader to Appendix C for the proof of this theorem.

### 4.5 Key Sanity Check Proof

In this subsection, we will give the key sanity check proof of our AAT-CP-ABE system. We use the proof method from [23].

**Theorem 4.** *The advantage of an attacker in the key sanity check game (in Subsect. 3.2) is negligible for our AAT-CP-ABE system.*

Due to space, we refer the reader to Appendix D for the proof of this theorem.

## 5 Conclusion and Future Work

In this work, we addressed two practical problems about the key abuse of CP-ABE in the cloud, and have presented an accountable authority CP-ABE system supporting white-box traceability and public auditing. Specifically, the proposed system could trace the malicious users for illegal key sharing. And for the semi-trusted authority, its illegal key (re-)distributing misbehavior could be caught and prosecuted. Furthermore, we have provided an auditor to judge whether a malicious user is innocent or framed by the authority. As far as we known, this is the first CP-ABE system that simultaneously supports white-box traceability, accountable authority and public auditing. We have also proved that the new system is fully secure in the standard model.

Note that there exists a stronger notion for traceability called black-box traceability. In black-box scenario, the malicious user could hide the decryption algorithm by tweaking it, as well as the decryption key. And in this case, the proposed system with white-box traceability in this paper will fail since both the decryption key and decryption algorithm are not well-formed. In our future work, we will focus on constructing an accountable authority CP-ABE system which is black-box traceability and public auditing.

**Acknowledgements.** We are grateful to the anonymous reviewers for their invaluable suggestions. This work is supported in part by the National Natural Science Foundation of China under Grant 61321064, Grant 61371083, Grant 61373154, Grant 61402282, and Grant 61411146001, in part by the Specialized Research Fund for the Doctoral Program of Higher Education of China through the Prioritized Development Projects under Grant 20130073130004, and in part by the Natural Science Foundation of Shanghai of Yang-Fan Plan under Grant 14YF1410400.

## A Proof of Lemma 2

*Proof.* Suppose there exists a PPT attacker $\mathcal{A}$ that has advantage $Adv_{\mathcal{A}}\Sigma_{aatcpabe}$ in breaking $\Sigma_{aatcpabe}$. We construct a PPT algorithm $\mathcal{B}$ that has advantage $Adv_{\mathcal{B}}\Sigma_{cpabe}$ in breaking the underlying CP-ABE system $\Sigma_{cpabe}$, which equals to $Adv_{\mathcal{A}}\Sigma_{aatcpabe}$.

- **Setup:** $\Sigma_{cpabe}$ gives $\mathcal{B}$ the public parameters $pp_{cpabe} = (N, g, g^{\beta}, e(g,g)^{\alpha}, \{\mathcal{U}_i = g^{u_i}\}_{i\in\mathcal{U}})$. $\mathcal{B}$ randomly chooses $a, \kappa \in \mathbb{Z}_N$, it also chooses two random primes $p$ and $q$ for which it holds $p \neq q$, $|p| = |q|$ and $gcd(pq, (p-1)(q-1)) = 1$, and then let $n = pq$, $\pi = lcm(p-1, q-1)$, $Q = \pi^{-1}\ mod\ n$ and $g_1 = (1+n)$. $\mathcal{B}$ gives $\mathcal{A}$ the public parameters $(N, n, g_1, v = g^{\beta}, g, g^a, g^{\kappa}, g^{\mu}, e(g,g)^{\alpha}, \{\mathcal{U}_i = g^{u_i}\}_{i\in\mathcal{U}})$.
- **Query Phase 1:** The attacker $\mathcal{A}$ will submit $(id, S)$ to $\mathcal{B}$ to query a decryption key, then $\mathcal{B}$ submits $S$ to $\Sigma_{cpabe}$ and gets the corresponding decryption key in the form of $\tilde{sk} = \langle \tilde{K} = g^{\alpha}g^{\beta\tilde{c}}R, \tilde{L} = g^{\tilde{c}}R_0, \{\tilde{K}_i = \mathcal{U}_i^{\tilde{c}}R_i\}_{i\in S}\rangle$. Note that in the proof of [14], the authority is free to choose a decryption key on its own and passes it on to the user. In our setting, however, the authority and the user

engage in a key generation protocol where the decryption key is jointly determined by both of them (via the choice of numbers $t$ and $c$). Hence the authority does not have complete control over the decryption key. The problem can be solved as follows. The authority generates a primary secret key $sk_{pri}$ on its own and then "forces" the output of a user during key generation. Recall that during the key generation protocol, a user first chooses a random $t \in \mathbb{Z}_N$ and sends $R_U = g^t$ to the authority. The user gives to the authority a zero-knowledge proof of knowledge of the discrete log of $R_U$. The proof of knowledge property of the proof system implies the existence of a *knowledge extractor* **Extr** (see Sect. 2.6). Using **Extr** on the user during the proof of knowledge protocol, the authority can extract the discrete log $t$ (by rewinding the user during protocol execution) with all but negligible probability. Thus, in the IND-CPA security game, $\mathcal{B}$ could extract the discrete log $t$ of $R_U$ (which was sent by the attacker $\mathcal{A}$). Then $\mathcal{B}$ chooses a random $r \in \mathbb{Z}_N^*$. It computes $T = \bar{T} = g_1^{id} r^n \ mod\ n^2$ and $1/(a+T)$ *modulo* $N$. Then $\mathcal{B}$ sets $c = \tilde{c}/(a+T)$, $t_{id} = c/t$ implicitly and randomly chooses $R_0' \in G_{p_3}$ by using $g_3$, then computes $\bar{K} = (\tilde{K})^{\frac{1}{a+T}}(g^t)^{\frac{\kappa}{a+T}} = (g^\alpha g^{\beta\tilde{c}} R)^{\frac{1}{a+T}} g^{\frac{\kappa t}{a+T}} = g^{\frac{\alpha}{a+T}} v^c g^{\frac{\kappa t}{a+T}} R^{\frac{1}{a+T}}, K = \bar{K}(g^\mu)^{t_{id}}$, $\bar{L} = (\tilde{L})^{\frac{1}{a+T}} = (g^{\tilde{c}} R_0)^{\frac{1}{a+T}} = g^c R_0^{\frac{1}{a+T}}, L = \bar{L}$, $\bar{L}' = (\tilde{L})^{\frac{a}{a+T}} = (g^{\tilde{c}} R_0)^{\frac{a}{a+T}} = g^{ac} R_0^{\frac{a}{a+T}} R_0', L' = \bar{L}'$, $\{\bar{K}_i = \tilde{K}_i = \mathcal{U}_i^{\tilde{c}} R_i = \mathcal{U}_i^{(a+T)c} R_i\}_{i\in S}, \{K_i = \tilde{K}_i\}_{i\in S}$. $\mathcal{B}$ gives $\mathcal{A}$ the decryption key $sk_{id,S} = \langle S, K, T, L, L', R_U, t_{id}, \{K_i\}_{i\in S}\rangle$.[5]

- **Challenge:** The attacker $\mathcal{A}$ submits to $\mathcal{B}$ two equal length messages $(m_0, m_1)$ and an LSSS matrix $(A^*, \rho)$. Then $\mathcal{B}$ submits $(m_0, m_1)$ and $(A^*, \rho)$ to $\Sigma_{cpabe}$, and obtains the challenge ciphertext as follows: $\tilde{ct} = \langle \tilde{C} = m_\delta \cdot e(g,g)^{\alpha s}, \tilde{C}_0 = g^s, \{\tilde{C_{j,1}} = g^{\beta A_j \overrightarrow{y}} \mathcal{U}_{\rho(j)}^{-r_j}, \tilde{C_{j,2}} = g^{r_j}\}_{j\in[l]}, (A^*, \rho)\rangle$. $\mathcal{B}$ sets $C = \tilde{C}, C_0 = \tilde{C}_0, C_1 = (\tilde{C}_0)^a = g^{as}, C_2 = (\tilde{C}_0)^\kappa = g^{\kappa s}, C_3 = (\tilde{C}_0)^\mu = g^{\mu s}, C_{j,1} = \tilde{C_{j,1}} = v^{A_j \overrightarrow{y}} \mathcal{U}_{\rho(j)}^{-r_j}$, $C_{j,2} = \tilde{C_{j,2}}$. Then, $\mathcal{B}$ gives the challenge ciphertext $ct = \langle C, C_0, C_1, C_2, C_3, \{C_{j,1}, C_{j,2}\}_{j\in[l]}, (A^*, \rho)\rangle$ to $\mathcal{A}$.
- **Query Phase 2:** This phase is the same with Phase 1.
- **Guess:** $\mathcal{A}$ outputs and gives his guess $\delta'$ to $\mathcal{B}$. Then $\mathcal{B}$ gives $\delta'$ to $\Sigma_{cpabe}$.

Since the distributions of the public parameters, decryption keys and challenge ciphertext in the above game are the same as that in the real system, we have $Adv_{\mathcal{B}}\Sigma_{cpabe} = Adv_{\mathcal{A}}\Sigma_{aatcpabe}$.

## B Proof of Theorem 2

*Proof.* Suppose there exists a PPT attacker $\mathcal{A}$ that has non-negligible advantage in winning the DishonestAuthority game for our AAT-CP-ABE system. We construct a PPT algorithm $\mathcal{B}$ that has non-negligible advantage in solving discrete log in $G_{p_1}$.

$\mathcal{B}$ proceeds as follows. $\mathcal{B}$ runs the algorithm $\mathcal{A}$ and gets the public parameters $pp = (N, n, g_1, v, g, g^a, g^\kappa, g^\mu, e(g,g)^\alpha, \{\mathcal{U}_i = g^{u_i}\}_{i\in\mathcal{U}})$ and a user's $(id, S)$ from

[5] Note that $R_0'$ makes the $G_{p_3}$ part of $L'$ uncorrelated to the $G_{p_3}$ part of $L$, this is why our simulator needs $g_3$.

$\mathcal{A}$. It then invokes the challenger and passes on $g$ to it, and gets a challenge $R_U = g^t$. The goal of $\mathcal{B}$ is to makes use of $\mathcal{A}$ to get the discrete log $t$ of $R_U$ with respect to $g$.

$\mathcal{B}$ will engage in the key generation protocol with $\mathcal{A}$ to get a decryption key for the user with $(id, S)$ as follows. It sends $R_U$ to the attacker $\mathcal{A}$ and has to give a zero-knowledge proof of knowledge of the discrete log of $R_U$. The zero-knowledge property of the proof system implies the existence of a simulator $S$ which is able to successfully simulate the view of $\mathcal{A}$ in the protocol (by rewinding $\mathcal{A}$) with all but negligible probability. $\mathcal{B}$ will use the simulator $S$ to simulate the required proof even without of knowledge of $t$. And $\mathcal{B}$ receives $c$ and the primary secret key $sk_{pri}$ as follows: $\langle S,\ \overline{K} = g^{\frac{\alpha}{a+\overline{T}}} g^{\frac{\kappa t}{a+\overline{T}}} v^c R,\ \overline{T} = g_1^{id} r^n \ mod\ n^2,\ \overline{L} = g^c R_0,\ \overline{L'} = g^{ac} R_0',\ \{\overline{K_i} = \mathcal{U}_i^{(a+\overline{T})c} R_i\}_{i \in S} \rangle$. As before, $\mathcal{B}$ checks whether the following equalities hold or not: (1)$e(\overline{L'}, g) = e(\overline{L}, g^a) = e(g^a, (g)^c)$; (2) $e(\overline{K}, g^a g^{\overline{T}}) = e(g, g)^\alpha e(\overline{L'}(\overline{L})^{\overline{T}}, v) e(R_U, g^\kappa)$; (3) $\exists x \in S$ *s.t.* $e(\mathcal{U}_x, \overline{L'}(\overline{L})^{\overline{T}}) = e(\overline{K_x}, g)$.

If any of these checks fail, $\mathcal{B}$ aborts as would an honest user in the key generation protocol.

Now with non-negligible advantage, the attacker $\mathcal{A}$ outputs a decryption key $sk_{id}^*$ such that $\texttt{Trace}(pp, msk, sk_{id}^*) \rightarrow id$, $\texttt{Audit}(pp, sk_{id}, sk_{id}^*) \rightarrow guilty$ and $t_{id}^*$ equals $t_{id}$ (which is unknown to $\mathcal{B}$). The decryption key $sk_{id}^*$ is set as follows: $\langle S,\ K = \overline{K}(g^\mu)^{t_{id}},\ T = \overline{T},\ L = \overline{L},\ L' = \overline{L'},\ R_U,\ t_{id}^*,\ \{K_i = \overline{K_i}\}_{i \in S} \rangle$. Then $\mathcal{B}$ computes $t = c/t_{id}^*$ and outputs $t$ as the discrete log (with respect to $g$) of the challenge $R_U$ and halts.

## C Proof Sketch of Theorem 3

**Proof Sketch.** Suppose there exists a PPT attacker $\mathcal{A}$ that has non-negligible advantage $\epsilon$ in winning the traceability game after making $q'$ key queries, w.l.o.g., assuming $q = q' + 1$, we construct a PPT algorithm that has non-negligible advantage in breaking $q$-SDH assumption or Assumption 2. $\mathcal{B}$ is given an instance of $q$-SDH problem and an instance of Assumption 2 problem as follows[6].

- $\mathcal{B}$ is given an instance of $q$-SDH problem: Let $G$ be a bilinear group of order $N = p_1 p_2 p_3$ (three distinct primes), $G_i$ be the subgroup of order $p_i$ in $G$ (where $1 \leq i \leq 3$), $e : G \times G \rightarrow G_T$ be a bilinear map, $a \in \mathbb{Z}_{p_1}^*$ and $\tilde{g} \in G_{p_1}$. $\mathcal{B}$ is given an instance of $q$-SDH problem $\mathcal{INS}_{SDH} = (G, G_T, N, e, \tilde{g}, \tilde{g}^a, ..., \tilde{g}^{a^q}, p_1, p_2, p_3)$.
- $\mathcal{B}$ is given an instance of Assumption 2 problem: Let $G$ be a bilinear group of order $N = p_1 p_2 p_3$ (three distinct primes), $G_i$ be the subgroup of order $p_i$ in $G$ (where $1 \leq i \leq 3$), $e : G \times G \rightarrow G_T$ be a bilinear map, $\tilde{g}, X_1 \in G_{p_1}$, $X_2, Y_2 \in G_{p_2}$, $X_3, Y_3 \in G_{p_3}$, $\delta \in \{0, 1\}$ and if $\delta = 0$, $T' \in G$, if $\delta = 1$, $T' \in G_{p_1 p_3}$. $\mathcal{B}$ is given an instance of Assumption 2 problem $\mathcal{INS}_{Ass2} = (G, G_T, N, e, \tilde{g}, X_1 X_2, X_3, Y_2 Y_3, T')$.

[6] Note that this two instances are independent from each other.

The goal of $\mathcal{B}$ is to output a bit $\delta' \in \{0,1\}$ to determine $T' \in G$ or $T' \in G_{p_1 p_3}$ for solving the Assumption 2 problem, and a tuple $(T_i, w_i) \in \mathbb{Z}_{p_1} \times G_{p_1}$ satisfying $w_i = \tilde{g}^{1/(a+T_i)}$ for solving the $q$-SDH problem. $\mathcal{B}$ will make use of $\mathcal{A}$ to break at least one of the above assumptions.

Note that the structure of our system is similar to that of [21], and both of the two systems use a Boneh-Boyen-style signature to achieve the unforgeability property of decryption key. Correspondingly, the proof of the DishonestUser game in our system is also similar to the proof of white-box traceability in [21]. And using a similar proof method from [21], it is easy to give a proof that $\mathcal{B}$ will make use of $\mathcal{A}$ to break at least one of the above assumptions in our system. Due to space limitations, we refer the interested reader to the full version of this paper for the proof of this theorem.

## D Proof of Theorem 4

*Proof.* Let the output of an attacker $\mathcal{A}$ be the public parameters $pp$, two different secret keys $sk_{id,S} = \langle S, K, T, L, L', R_U, t_{id}, \{K_i\}_{i\in S}\rangle$ and $\tilde{sk}_{id,S} = \langle S, \tilde{K}, \tilde{T}, \tilde{L}, \tilde{L}', \tilde{R}_U, \tilde{t}_{id}, \{\tilde{K}_i\}_{i\in S}\rangle$, and a ciphertext $ct = \langle C, C_0, C_1, C_2, C_3, \{C_{j,1}, C_{j,2}\}_{j\in[l]}, (A,\rho)\rangle$. $\mathcal{A}$ wins implies that the following conditions (as defined in the key sanity check game in Subsect. 3.2) are all fulfilled.

Conditions (1) – (5):

(1) $\texttt{KeySanityCheck}(pp, sk_{id,S}) \to 1$; (2) $\texttt{KeySanityCheck}(pp, \tilde{sk}_{id,S}) \to 1$;

(3) $\texttt{Decrypt}(pp, sk_{id,S}, ct) \neq \perp$; (4) $\texttt{Decrypt}(pp, \tilde{sk}_{id,S}, ct) \neq \perp$;

(5) $\texttt{Decrypt}(pp, sk_{id,S}, ct) \neq \texttt{Decrypt}(pp, \tilde{sk}_{id,S}, ct)$.

Condition (1) implies

(1) $sk$ is in the form of $(S, K, T, L, L', R_U, t_{id}, \{K_i\}_{i\in S})$ and $T \in \mathbb{Z}^*_{n^2}, K, L, L', R_U, \{K_i\}_{i\in S} \in G$.
(2) $e(L', g) = e(L, g^a)$.
(3) $e(K, g^a g^T) = e(g,g)^\alpha e(L'(L)^T, v) e(R_U, g^\kappa) e((g^a g^T)^{t_{id}}, g^\mu)$.
(4) $\exists x \in S$ *s.t.* $e(\mathcal{U}_x, L'(L)^T) = e(K_x, g)$.

Similarly, condition (2) implies

(1) $sk$ is in the form of $(S, \tilde{K}, \tilde{T}, \tilde{L}, \tilde{L}', \tilde{R}_U, \tilde{t}_{id}, \{\tilde{K}_i\}_{i\in S})$ and $\tilde{T} \in \mathbb{Z}^*_{n^2}$, $\tilde{K}, \tilde{L}, \tilde{L}', \tilde{R}_U, \{\tilde{K}_i\}_{i\in S} \in G$.
(2) $e(\tilde{L}', g) = e(\tilde{L}, g^a)$.
(3) $e(\tilde{K}, g^a g^{\tilde{T}}) = e(g,g)^\alpha e(\tilde{L}'(\tilde{L})^{\tilde{T}}, v) e(\tilde{R}_U, g^\kappa) e((g^a g^{\tilde{T}})^{\tilde{t}_{id}}, g^\mu)$.
(4) $\exists x \in S$ *s.t.* $e(\mathcal{U}_x, \tilde{L}'(\tilde{L})^{\tilde{T}}) = e(\tilde{K}_x, g)$.

From conditions (1) and (3), we have $D = e((C_0)^T C_1, K)(e(C_2, R_u) e(C_3, (g^T g^a)^{t_{id}}))^{-1}$, $E = \Pi_{\rho(j)\in S}(e(C_{j,1}, (L)^T L') e(C_{j,2}, K_{\rho(j)}))^{\omega_j}$, $F = D/E = e(g,g)^{\alpha s}, m = C/F$. And from conditions (2) and (4), we have $\tilde{D} = e((C_0)^{\tilde{T}} C_1, \tilde{K})(e(C_2, \tilde{R}_u) e(C_3, (g^{\tilde{T}} g^a)^{\tilde{t}_{id}}))^{-1}$, $\tilde{E} = \Pi_{\rho(j)\in S}(e(C_{j,1}, (\tilde{L})^{\tilde{T}} \tilde{L}') e(C_{j,2}, \tilde{K}_{\rho(j)}))^{\omega_j}, \tilde{F} = \tilde{D}/\tilde{E} = e(g,g)^{\alpha s}, m = C/\tilde{F}$.

From conditions (1) − (4), we have $F = D/E = e(g,g)^{\alpha s} = \tilde{F} = \tilde{D}/\tilde{E}, m = C/F = C/\tilde{F}$ $(*)$. However, condition (5) implies that $C/F \neq C/\tilde{F}$, where $F = D/E, \tilde{F} = \tilde{D}/\tilde{E}$, which contradicts to $(*)$. Thus $\mathcal{A}$ wins the game only with negligible probability.

## References


1. Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., et al.: A view of cloud computing. Commun. ACM **53**(4), 50–58 (2010)
2. Beimel, A.: Secure schemes for secret sharing and key distribution. Ph.D. thesis, Israel Institute of Technology, Technion, Haifa, Israel (1996)
3. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Brickell, E.F. (ed.) CRYPTO 1992. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (1993)
4. Bethencourt, J., Sahai, A., Waters, B.: Ciphertext-policy attribute-based encryption. In: IEEE Symposium on Security and Privacy. SP 2007, pp. 321–334. IEEE (2007)
5. Boneh, D., Boyen, X.: Short signatures without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 56–73. Springer, Heidelberg (2004)
6. Boneh, D., Franklin, M.: Identity-based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, p. 213. Springer, Heidelberg (2001)
7. Boneh, D., Goh, E.-J., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
8. Chase, M.: Multi-authority attribute based encryption. In: Vadhan, S.P. (ed.) TCC 2007. LNCS, vol. 4392, pp. 515–534. Springer, Heidelberg (2007)
9. Deng, H., Wu, Q., Qin, B., Mao, J., Liu, X., Zhang, L., Shi, W.: Who Is touching my cloud. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014, Part I. LNCS, vol. 8712, pp. 362–379. Springer, Heidelberg (2014)
10. Goyal, V.: Reducing trust in the PKG in identity based cryptosystems. In: Menezes, A. (ed.) CRYPTO 2007. LNCS, vol. 4622, pp. 430–447. Springer, Heidelberg (2007)
11. Goyal, V., Jain, A., Pandey, O., Sahai, A.: Bounded ciphertext policy attribute based encryption. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 579–591. Springer, Heidelberg (2008)
12. Goyal, V., Lu, S., Sahai, A., Waters, B.: Black-box accountable authority identity-based encryption. In: Proceedings of the 15th ACM Conference on Computer and Communications Security, pp. 427–436. ACM (2008)
13. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted data. In: Proceedings of the 13th ACM Conference on Computer and Communications Security, pp. 89–98. ACM (2006)
14. Lewko, A., Okamoto, T., Sahai, A., Takashima, K., Waters, B.: Fully secure functional encryption: attribute-based encryption and (hierarchical) inner product encryption. In: Gilbert, H. (ed.) EUROCRYPT 2010. LNCS, vol. 6110, pp. 62–91. Springer, Heidelberg (2010)
15. Lewko, A., Waters, B.: New proof methods for attribute-based encryption: achieving full security through selective techniques. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 180–198. Springer, Heidelberg (2012)

16. Lewko, A., Waters, B.: Decentralizing attribute-based encryption. In: Paterson, K.G. (ed.) EUROCRYPT 2011. LNCS, vol. 6632, pp. 568–588. Springer, Heidelberg (2011)
17. Li, J., Huang, Q., Chen, X., Chow, S.S.M., Wong, D.S., Xie, D.: Multi-authority ciphertext-policy attribute-based encryption with accountability. In: Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, pp. 386–390. ACM (2011)
18. Li, J., Ren, K., Kim, K.: A2be: Accountable attribute-based encryption for abuse free access control. IACR Cryptology ePrint Arch. **2009**, 118 (2009)
19. Liu, Z., Cao, Z., Huang, Q., Wong, D.S., Yuen, T.H.: Fully secure multi-authority ciphertext-policy attribute-based encryption without random oracles. In: Atluri, V., Diaz, C. (eds.) ESORICS 2011. LNCS, vol. 6879, pp. 278–297. Springer, Heidelberg (2011)
20. Liu, Z., Cao, Z., Wong, D.S.: Blackbox traceable cp-abe: how to catch people leaking their keys by selling decryption devices on ebay. In: Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, pp. 475–486. ACM (2013)
21. Liu, Z., Cao, Z., Wong, D.S.: White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures. IEEE Trans. Inf. Forensics Secur. **8**(1), 76–88 (2013)
22. Ning, J., Cao, Z., Dong, X., Wei, L., Lin, X.: Large universe ciphertext-policy attribute-based encryption with white-box traceability. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014, Part II. LNCS, vol. 8713, pp. 55–72. Springer, Heidelberg (2014)
23. Ning, J., Dong, X., Cao, Z., Wei, L., Lin, X.: White-box traceable ciphertext-policy attribute-based encryption supporting flexible attributes. IEEE Trans. Inf. Forensics Secur. **10**(6), 1274–1288 (2015)
24. Okamoto, T., Takashima, K.: Fully secure functional encryption with general relations from the decisional linear assumption. In: Rabin, T. (ed.) CRYPTO 2010. LNCS, vol. 6223, pp. 191–208. Springer, Heidelberg (2010)
25. Ostrovsky, R., Sahai, A., Waters, B.: Attribute-based encryption with non-monotonic access structures. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, pp. 195–203. ACM (2007)
26. Sahai, A., Seyalioglu, H., Waters, B.: Dynamic credentials and ciphertext delegation for attribute-based encryption. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 199–217. Springer, Heidelberg (2012)
27. Sahai, A., Waters, B.: Fuzzy identity-based encryption. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 457–473. Springer, Heidelberg (2005)
28. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
29. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011)