RAAC: Robust and Auditable Access Control With Multiple Attribute Authorities for Public Cloud Storage

Kaiping Xue, *Senior Member, IEEE*, Yingjie Xue, Jianan Hong, Wei Li, Hao Yue, *Member, IEEE*, David S. L. Wei, *Senior Member, IEEE*, and Peilin Hong

***Abstract*— Data access control is a challenging issue in public cloud storage systems. Ciphertext-policy attribute-based encryption (CP-ABE) has been adopted as a promising technique to provide flexible, fine-grained, and secure data access control for cloud storage with honest-but-curious cloud servers. However, in the existing CP-ABE schemes, the single attribute authority must execute the time-consuming user legitimacy verification and secret key distribution, and hence, it results in a single-point performance bottleneck when a CP-ABE scheme is adopted in a large-scale cloud storage system. Users may be stuck in the waiting queue for a long period to obtain their secret keys, thereby resulting in low efficiency of the system. Although multi-authority access control schemes have been proposed, these schemes still cannot overcome the drawbacks of single-point bottleneck and low efficiency, due to the fact that each of the authorities still independently manages a disjoint attribute set. In this paper, we propose a novel heterogeneous framework to remove the problem of single-point performance bottleneck and provide a more efficient access control scheme with an auditing mechanism. Our framework employs multiple attribute authorities to share the load of user legitimacy verification. Meanwhile, in our scheme, a central authority is introduced to generate secret keys for legitimacy verified users. Unlike other multi-authority access control schemes, each of the authorities in our scheme manages the whole attribute set individually. To enhance security, we also propose an auditing mechanism to detect which attribute authority has incorrectly or maliciously performed the legitimacy verification procedure. Analysis shows that our system not only guarantees the security requirements but also makes great performance improvement on key generation.**

***Index Terms*— Cloud storage, access control, auditing, CP-ABE.**

I. Introduction

CLOUD storage is a promising and important service paradigm in cloud computing [1]–[4]. Benefits of using cloud storage include greater accessibility, higher reliability, rapid deployment and stronger protection, to name just a few. Despite the mentioned benefits, this paradigm also brings forth new challenges on data access control, which is a critical issue to ensure data security. Since cloud storage is operated by cloud service providers, who are usually outside the trusted domain of data owners, the traditional access control methods in the Client/Server model are not suitable in cloud storage environment. The data access control in cloud storage environment has thus become a challenging issue.

To address the issue of data access control in cloud storage, there have been quite a few schemes proposed, among which *Ciphertext-Policy Attribute-Based Encryption (CP-ABE)* is regarded as one of the most promising techniques. A salient feature of CP-ABE is that it grants data owners direct control power based on access policies, to provide flexible, fine-grained and secure access control for cloud storage systems. In CP-ABE schemes, the access control is achieved by using cryptography, where an owner's data is encrypted with an access structure over attributes, and a user's secret key is labelled with his/her own attributes. Only if the attributes associated with the user's secret key satisfy the access structure, can the user decrypt the corresponding ciphertext to obtain the plaintext. So far, the CP-ABE based access control schemes for cloud storage have been developed into two complementary categories, namely, single-authority scenario [5]–[9], and multi-authority scenario [10]–[12].

Although existing CP-ABE access control schemes have a lot of attractive features, they are neither robust nor efficient in key generation. Since there is only one authority in charge of all attributes in single-authority schemes, offline/crash of this authority makes all secret key requests unavailable during that period. The similar problem exists in multi-authority schemes, since each of multiple authorities manages a disjoint attribute set.

Manuscript received June 12, 2016; revised November 16, 2016; accepted December 13, 2016. Date of publication January 2, 2017; date of current version January 30, 2017. This work was supported in part by the National Natural Science Foundation of China under Grant 61379129, in part by the Youth Innovation Promotion Association CAS, and in part by the Fundamental Research Funds for the Central Universities. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Guofei Gu.

K. Xue is with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China, and also with the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China (e-mail: kpxue@ustc.edu.cn).

Y. Xue, J. Hong, W. Li, and P. Hong are with the Department of Electronic Engineering and Information Science, University of Science and Technology of China, Hefei 230027, China (e-mail: xyj1108@ustc.edu.cn; hongjn@ustc.edu.cn; lwz159@ustc.edu.cn; plhong@ustc.edu.cn).

H. Yue is with the Department of Computer Science, San Francisco State University, San Francisco, CA 94132 USA (e-mail: haoyue@sfsu.edu).

D. S. L. Wei is with the Computer and Information Science Department, Fordham University, New York City, NY 10458 USA (e-mail: wei@dsm.fordham.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIFS.2016.2647222

In single-authority schemes, the only authority must verify the legitimacy of users' attributes before generating secret keys for them. As the access control system is associated with data security, and the only credential a user possess is his/her secret key associated with his/her attributes, the process of key issuing must be cautious. However, in the real world, the attributes are diverse. For example, to verify whether a user is able to drive may need an authority to give him/her a test to prove that he/she can drive. Thus he/she can get an attribute key associated with driving ability [13]. To deal with the verification of various attributes, the user may be required to be present to confirm them. Furthermore, the process to verify/assign attributes to users is usually difficult so that it normally employs administrators to manually handle the verification, as [14] has mentioned, that the authenticity of registered data must be achieved by out-of-band (mostly manual) means. To make a careful decision, the unavoidable participation of human beings makes the verification time-consuming, which causes a single-point bottleneck. Especially, for a large system, there are always large numbers of users requesting secret keys. The inefficiency of the authority's service results in single-point performance bottleneck, which will cause system congestion such that users often cannot obtain their secret keys quickly, and have to wait in the system queue. This will significantly reduce the satisfaction of users experience to enjoy real-time services. On the other hand, if there is only one authority that issues secret keys for some particular attributes, and if the verification enforces users' presence, it will bring about the other type of long service delay for users, since the authority maybe too far away from his/her home/workplace. As a result, single-point performance bottleneck problem affects the efficiency of secret key generation service and immensely degrades the utility of the existing schemes to conduct access control in large cloud storage systems. Furthermore, in multi-authority schemes, the same problem also exists due to the fact that multiple authorities separately maintain disjoint attribute subsets and issue secret keys associated with users' attributes within their own administration domain. Each authority performs the verification and secret key generation as a whole in the secret key distribution process, just like what the single authority does in single-authority schemes. Therefore, the single-point performance bottleneck still exists in such multi-authority schemes.

A straightforward idea to remove the single-point bottleneck is to allow multiple authorities to jointly manage the universal attribute set, in such a way that each of them is able to distribute secret keys to users independently. By adopting multiple authorities to share the load, the influence of the single-point bottleneck can be reduced to a certain extent. However, this solution will bring forth threats on security issues. Since there are multiple functionally identical authorities performing the same procedure, it is hard to find the responsible authority if mistakes have been made or malicious behaviors have been implemented in the process of secret key generation and distribution. For example, an authority may falsely distribute secret keys beyond user's legitimate attribute set. Such weak point on security makes this straightforward idea hard to meet the security requirement of access control for public cloud storage. Our recent work, TMACS [15], is a threshold multi-authority CP-ABE access control scheme for public cloud storage, where multiple authorities jointly manage a uniform attribute set. Actually it addresses the single-point bottleneck of performance and security, but introduces some additional overhead. Therefore, in this paper, we present a feasible solution which not only promotes efficiency and robustness, but also guarantees that the new solution is as secure as the original single-authority schemes.

The similar problem has been considered and partly tackled in other related areas, such as *public key infrastructure (PKI)* for e-commerce [16]. To reduce *the certificate authority* (*CA*)'s load, one or more *registration authorities (RAs)* are introduced to perform some of administration tasks on behalf of *CA*. Each *RA* is able to verify a user's legitimacy and determine whether the user is entitled to have a valid certificate. After the verification, it validates the credentials and forwards the certificate request to *CA*. Then, *CA* will generate a certificate for the user. Since the most heavy work of verification is performed by a selected *RA*, the load of *CA* can be largely reduced. However, the security of the scheme with single-*CA*/multi-*RAs* partly depends on the trustiness of multiple *RA*s. In order to achieve traceability, *CA* should store some information to confirm which *RA* has been responsible for verifying the legitimacy of a specific user.

In this paper, inspired by the heterogeneous architecture with single *CA* and multiple *RAs*, we propose a robust and auditable access control scheme (named RAAC) for public cloud storage to promote the performance while keeping the flexibility and fine granularity features of the existing CP-ABE schemes. In our scheme, we seperate the procedure of user legitimacy verification from the secret key generation, and assign these two sub-procedures to two different kinds of authorities. There are multiple authorities (named attribute authorities, *AA*s), each of which is in charge of the whole attribute set and can conduct user legitimacy verification independently. Meanwhile, there is only one global trusted authority (referred as Central Authority, *CA*) in charge of secret key generation and distribution. Before performing a secret key generation and distribution process, one of the *AA*s is selected to verify the legitimacy of the user's attributes and then it generates an intermediate key to send to *CA*. *CA* generates the secret key for the user on the basis of the received intermediate key, with no need of any more verification. In this way, multiple *AA*s can work in parallel to share the load of the time-consuming legitimacy verification and standby for each other so as to remove the single-point bottleneck on performance. Meanwhile, the selected *AA* doesn't take the responsibility of generating final secret keys to users. Instead, it generates intermediate keys that associate with users' attributes and implicitly associate with its own identity, and sends them to *CA*. With the help of intermediate keys, *CA* is able to not only generate secret keys for legitimacy verified users more efficiently but also trace an *AA*'s mistake or malicious behavior to enhance the security.

The main contributions of this work can be summarized as follows.

1) To address the single-point performance bottleneck of key distribution existed in the existing schemes, we propose a robust and efficient heterogeneous framework with single *CA*(Central Authority) and multiple *AA*s (Attribute Authorities) for public cloud storage. The heavy load of user legitimacy verification is shared by multiple *AA*s, each of which manages the universal attribute set and is able to independently complete the user legitimacy verification, while *CA* is only responsible for computational tasks. To the best of our knowledge, this is the first work that proposes the heterogeneous access control framework to address the low efficiency and single-point performance bottleneck for cloud storage.
2) We reconstruct the CP-ABE scheme to fit our proposed framework and propose a robust and high-efficient access control scheme, meanwhile the scheme still preserves the fine granularity, flexibility and security features of CP-ABE.
3) Our scheme includes an auditing mechanism that helps the system trace an *AA*'s misbehavior on user's legitimacy verification.

The rest of this paper is organized as follows. In Section II, we brief some related works about CP-ABE and some existing extension schemes for cloud storage access control. In Section III, technical preliminaries are presented. Following the definition of system model and security assumptions in Section IV, we introduce our proposed heterogeneous framework with single-*CA*/multi-*AA*s and relative access control scheme for public cloud storage in Section V. In Section VI and Section VII, we analyze our proposed scheme in terms of security and performance, respectively. In Section VIII we briefly introduce the construction of a hybrid multi-authority system. Finally, the conclusion is given in Section IX.

II. Related Work

Ciphertext-Policy Attribute-Based Encryption (CP-ABE) has so far been regarded as one of the most promising techniques for data access control in cloud storage systems. This technology offers users flexible, fine-grained and secure access control of outsourced data. It was first formulated by Goyal et al. in [17]. Then the first CP-ABE scheme was proposed by Benthencourt et al. in [18], but this scheme was proved secure only in the generic group model. Subsequently, some cryptographically stronger CP-ABE constructions [19]–[21] were proposed, but these schemes imposed some restrictions that the original CP-ABE does not have. In [22], Waters proposed three efficient and practical CP-ABE schemes under stronger cryptographic assumptions as expressive as [18]. To improve efficiency of this encryption technique, Emura et al. [23] proposed a CP-ABE scheme with a constant ciphertext length. Unlike the above schemes which are only limited to express monotonic access structures, Obtrovsky et al. [24] proposed a more expressive CP-ABE scheme which can support non-monotonic access structures. Recently, Hohenberger and Waters [25] proposed an online/offline ABE technique for CP-ABE which enables the user to do as much precomputation as possible to save online computation. It's a promising technique for resource-limited devices.

In general, there are two categories of CP-ABE schemes classified by the number of participating authorities in key distribution process. One category is the single-authority scheme, the other is multi-authority scheme. In single-authority schemes [5]–[7], [26]–[29], only one authority is involved to manage the universal attribute set, generate and distribute secret keys for all users. In [7] and [26], the authors respectively proposed CP-ABE schemes with efficient attribute revocation capability for data outsourcing systems. Wu et al. [5] proposed a Multi-message Ciphertext-Policy Attribute-Based Encryption(MCP-ABE) which encrypts multiple messages within one ciphertext so as to enforce flexible attribute-based access control on scalable media. The literatures [27]–[29] took the efficiency issue into consideration, but they mainly considered the computation complexity inside the cryptography algorithms rather than interaction protocols between different entities in the real world, such as the procedure of user legitimacy verification. To sum up, in single-authority schemes, the single-point performance bottleneck has not been widely addressed so far.

To meet some scenarios where users' attributes come from multiple authorities, some multi-authority schemes have been proposed. Based on the basic ABE [30] scheme, Chase et al. [31] proposed the first multi-authority scheme which allows multiple independent authorities to monitor attributes and distribute corresponding secret keys, but involves a central authority (*CA*). Subsequently, some multi-authority ABE schemes without *CA* have been proposed, such as [13] and [32]. Since the first construction of CP-ABE [18], a great many multi-authority schemes have been conducted over CP-ABE. Muller et al. [33] proposed the first multi-authority CP-ABE scheme in which a user's secret key was issued by an arbitrary number of attribute authorities and a master authority. Then Lewko et al. [10] proposed a decentralized CP-ABE scheme where the secret keys can be generated fully by multiple authorities without a central authority. Ruj et al. [34] applied Lewko's work [10] for access control in cloud storage systems, and also proposed a revocation method. Lin et al. [32] proposed a decentralized access control scheme based on threshold mechanism. In [11] and [36], the authors proposed two efficient multi-authority CP-ABE schemes for data access control in cloud storage systems, where a central authority is only needed in system initialization phase. Based on the basic multi-authority architecture, some other literatures tried to address the user identity privacy issue [36], [37], policy update [38], and the accountability to prevent key abusing [39], [40]. However, in above multi-authority schemes, multiple authorities separately manage disjoint attribute sets. That is to say, for each attribute, only one authority could issue secret keys associated with it. Therefore, in large-scale systems, the single-point performance bottleneck still exists in multi-authority schemes due to the property that each of the multiple authorities maintains only a disjoint subset of attributes.

Recently, we considered the single-point performance bottleneck of CP-ABE based schemes and devised a threshold multi-authority CP-ABE access control scheme in our another work [15]. Different from other multi-authority schemes, in [15], multiple authorities jointly manage a uniform

attribute set. Taking advantage of (t, n) threshold secret sharing, the master secret key can be shared among multiple authorities, and a legal user can generate his/her secret key by interacting with any t authorities. This scheme actually addressed the single-point bottleneck on both security and performance in CP-ABE based access control in public cloud storage. However, it is not efficient, because a user has to interact with at least t authorities, and thus introduces higher interaction overhead.

In this paper, we present an efficient heterogeneous framework with single *CA*/multiple *AA*s to address the problem of single-point performance bottleneck. The novel idea of our proposed scheme is that the complicated and time-consuming user legitimacy verification is executed only once by one selected *AA*. Furthermore, an auditing mechanism is proposed to ensure the traceability of malicious *AA*s. Thus our scheme can not only remove the single-point performance bottleneck but also be able to provide a robust, high-efficient, and secure access control for public cloud storage.

III. Preliminaries and Definitions

In this section, we first give a brief review of background information on bilinear maps and the security assumptions defined on it. Then we briefly introduce CP-ABE and LSSS which are the constituents in our scheme.

A. Bilinear Maps

Let $\mathbb{G}, \mathbb{G}_T$ be two multiplicative cyclic groups with the same prime order p, and g be a generator of $\mathbb{G}$. A bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ defined on $\mathbb{G}$ has the following three properties:

1) *Bilinearity*: $\forall a, b \in \mathbb{Z}_p$ and $g_1, g_2 \in \mathbb{G}$, we have $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
2) *Non-degeneracy*: $\forall g_1, g_2 \in \mathbb{G}$ such that $e(g_1, g_2) \neq 1$, which means the map does not send all pairs in $\mathbb{G} \times \mathbb{G}$ to the identity in $\mathbb{G}_T$.
3) *Computability*: There is an efficient algorithm to compute $e(g_1, g_2)$ for all $g_1, g_2 \in \mathbb{G}$.

Definition 1 (Decisional q-Parallel Bilinear Diffie-Hellman Exponent Assumption(Decisional q-BDHE)): The decisional q-BDHE problem is that, in a group $\mathbb{G}$ of prime order p, give a, s, b_1, b_2, ..., $b_q \in \mathbb{Z}_p$, if an adversary is given:

$$\vec{y} = (g, g^s, g^a, \ldots, g^{(a^q)}, g^{(a^{q+2})}, \ldots, g^{(a^{2q})}$$
$$\forall_{1 \le j \le q}, g^{s \cdot b_j}, g^{a/b_j}, \ldots, g^{(a^q/b_j)}, g^{(a^{q+2}/b_j)}, \ldots, g^{(a^{2q}/b_j)}$$
$$\forall_{1 \le j, l \le q, l \neq j}, g^{a \cdot s \cdot b_l / b_j}, \ldots, g^{a^q \cdot s \cdot b_l / b_j}),$$

it must remain hard to distinguish $e(g, g)^{a^{q+1}s} \in \mathbb{G}_T$ from a random element R in $\mathbb{G}_T$.

B. Ciphertext-Policy Attribute-Based Encryption (CP-ABE)

Although the definitions and constructions of different CP-ABE schemes are not always consistent, the uses of the access structure in Encrypt and Decrypt algorithms are nearly the same. Here we adopt the definition and construction from [18] and [22].

A CP-ABE scheme consists of four algorithms: Setup, Encrypt, Key Generation (KeyGen), and Decrypt.

Setup$(\lambda, U) \to (PK, MSK)$. The setup algorithm takes the security parameter λ and the attribute universe description U as the input. It outputs the public parameters PK and a master secret key MSK.

Encrypt$(PK, M, \mathbb{A}) \to CT$. The encryption algorithm takes the public parameters PK, a message M, and an access structure $\mathbb{A}$ as input. The algorithm will encrypt M and produce a ciphertext CT such that only a user whose attributes satisfies the access structure will be able to decrypt the message. We will assume that the ciphertext implicitly contains $\mathbb{A}$.

KeyGen$(MSK, S) \to SK$. The key generation algorithm takes the master secret key MSK and a set of attributes S as input. It outputs a secret key SK.

Decrypt$(PK, CT, SK) \to M$. The decryption algorithm takes the public parameters PK, a ciphertext CT which contains an access policy $\mathbb{A}$, and a secret key SK as input, where SK is a secret key for a set S of attributes. If the set S of attributes satisfies the access structure $\mathbb{A}$, the algorithm will decrypt the ciphertext and return a message M.

C. Access Structures and Linear Secret Sharing Schemes

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

Observing the constructions in [17], [18], and [22], an LSSS access structure can be used to denote the access policy $\mathbb{A}$. Following the method defined in [41], any monotonic boolean formula can be converted into an LSSS representation. The description of LSSS is presented as follows.

Definition 3 (Linear Secret Sharing Schemes (LSSS)): A secret-sharing scheme Π over a set of parties $\mathcal{P}$ is called linear (over $\mathbb{Z}_p$) if:

1) *The shares for each party form a vector over $\mathbb{Z}_p$.*
2) *There exists a matrix an $\mathcal{M}$ with l rows and n columns, which is called the sharing-generating matrix for Π. For all $i = 1, \ldots, l$, the i-th row of $\mathcal{M}$ is labeled by a party $\rho(i)$, where ρ is the function associating rows of $\mathcal{M}$ to parties in $\mathcal{P}$. When we consider the vector $\vec{v} = (s, r_2, \ldots, r_n) \in \mathbb{Z}_p^n$, where $r_2, \ldots, r_n$ are randomly chosen and s is the secret to be shared, then $\vec{\lambda} = \mathcal{M} \times \vec{v}^\top$ is the vector of l shares of the secret s according to Π. The share λ_i belongs to the party $\rho(i)$.*

Every linear secret sharing-scheme based on the above definition enjoys the linear reconstruction property, defined as follows: Suppose that Π is an LSSS structure for access structure $\mathbb{A}$. Let $\mathcal{S} \in \mathbb{A}$ be any authorized set, and let $I \subset \{1, 2, \ldots, l\}$ be defined as $I = \{i : \rho(i) \in \mathcal{S}\}$. Then, there exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that, if $\{\lambda_i\}$ are valid shares of any secret s according to Π, then $\Sigma_{i \in I} \omega_i \lambda_i = s$. These constants $\{\omega_i\}$ can be found in time polynomial in the size of the share-generating matrix $\mathcal{M}$. Note that, for any unauthorized set $\mathcal{S} \notin \mathbb{A}$, no such constants $\{\omega_i\}$ exist.

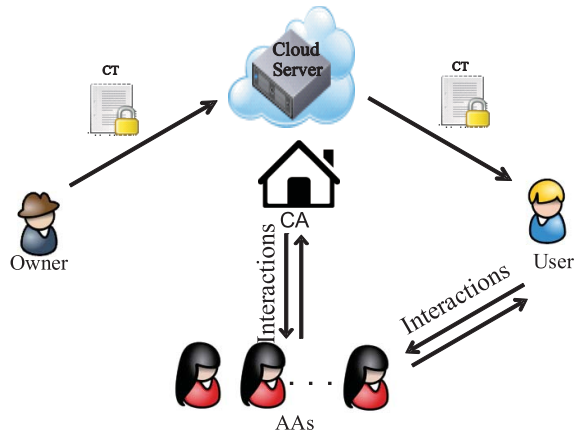


Fig. 1. System model.

IV. System Model and Security Assumptions

In this section, we give the definitions of the system model, the security assumptions and requirements of our public cloud storage access control.

A. System Model

The system model of our design is shown in Fig. 1, which involves five entities: a ***central authority (CA)***, multiple ***attribute authorities (AAs)***, many ***data owners (Owners)***, many ***data consumers (Users)***, and a ***cloud service provider with multiple cloud servers***(here, we mention it as ***cloud server***.).

- ***The central authority (*CA*)*** is the administrator of the entire system. It is responsible for the system construction by setting up the system parameters and generating public key for each attribute of the universal attribute set. In the system initialization phase, it assigns each user a unique Uid and each attribute authority a unique Aid. For a key request from a user, *CA* is responsible for generating secret keys for the user on the basis of the received intermediate key associated with the user's legitimate attributes verified by an *AA*. As an administrator of the entire system, *CA* has the capacity to trace which *AA* has incorrectly or maliciously verified a user and has granted illegitimate attribute sets.
- ***The attribute authorities (*AAs*)*** are responsible for performing user legitimacy verification and generating intermediate keys for legitimacy verified users. Unlike most of the existing multi-authority schemes where each *AA* manages a disjoint attribute set respectively, our proposed scheme involves multiple authorities to share the responsibility of user legitimacy verification and each *AA* can perform this process for any user independently. When an *AA* is selected, it will verify the users' legitimate attributes by manual labor or authentication protocols, and generate an intermediate key associated with the attributes that it has legitimacy-verified. Intermediate key is a new concept to assist *CA* to generate keys.
- ***The data owner (Owner)*** defines the access policy about who can get access to each file, and encrypts the file under the defined policy. First of all, each owner encrypts his/her data with a symmetric encryption algorithm. Then, the owner formulates access policy over an attribute set and encrypts the symmetric key under the policy according to public keys obtained from *CA*. After that, the owner sends the whole encrypted data and the encrypted symmetric key (denoted as ciphertext CT) to the cloud server to be stored in the cloud.
- ***The data consumer (User)*** is assigned a global user identity Uid by *CA*. The user possesses a set of attributes and is equipped with a secret key associated with his/her attribute set. The user can freely get any interested encrypted data from the cloud server. However, the user can decrypt the encrypted data if and only if his/her attribute set satisfies the access policy embedded in the encrypted data.
- ***The cloud server*** provides a public platform for owners to store and share their encrypted data. The cloud server doesn't conduct data access control for owners. The encrypted data stored in the cloud server can be downloaded freely by any user.

B. Security Assumptions and Requirements

In our proposed scheme, the security assumptions of the five roles are given as follows. The cloud server is always online and managed by the cloud provider. Usually, the cloud server and its provider are assumed to be "honest-but-curious", which means that they will correctly execute the tasks assigned to them for profits, but they would try to find out as much secret information as possible based on data owners' inputs and uploaded files. *CA* is the administrator of the entire system, which is always online and can be assumed to be fully trusted. It will not collude with any entity to acquire data contents. *AA*s are responsible for conducting legitimacy verification of users and judging whether the users have the claimed attributes. We assume that *AA* can be compromised and cannot be fully trusted. Furthermore, since the user legitimacy verification is conducted by manual labor, mis-operation caused by carelessness may also happen. Thus, we need an auditing mechanism to trace an *AA*'s misbehavior. Although a user can freely get any encrypted data from the cloud server, he/she cannot decrypt it unless the user has attributes satisfying the access policy embedded inside the data. Therefore, some users may be dishonest and curious, and may collude with each other to gain unauthorized access or try to collude with (or even compromise) any *AA* to obtain the access permission beyond their privileges. Owners have access control over their uploaded data, which are protected by specific access policies they defined.

To guarantee secure access control in public cloud storage, we claim that an access control scheme needs to meet the following four basic security requirements:

- *Data confidentiality*. Data content must be kept confidential to unauthorized users as well as the curious cloud server.
- *Collusion-resistance*. Malicious users colluding with each other would not be able to combine their attributes to

decrypt a ciphertext which each of them cannot decrypt alone.
- AA *accountability*. An auditing mechanism must be devised to ensure that an *AA*'s misbehavior can be detected to prevent *AA*s' abusing their power without being detected.
- *No ultra vires for any* AA. An *AA* should not have unauthorized power to directly generate secret keys for users. This security requirement is newly introduced based on our proposed hierarchical framework.

V. Our Proposed Access Control scheme

This section first gives an overview of our proposed scheme, and then describes the scheme in detail. Our scheme consists of five phases, namely ***System Initialization***, ***Encryption***, ***Key Generation***, ***Decryption***, and ***Auditing & Tracing***.

A. Overview of Our Scheme

To achieve a robust and efficient access control for public cloud storage, we propose a hierarchical framework with single *CA* and multiple *AA*s to remove the problem of single-point performance bottleneck and enhance the system efficiency. In our proposed RAAC scheme, the procedure of key generation is divided into two sub-procedures: 1) the procedure of user legitimacy verification; 2) the procedure of secret key generation and distribution. The user legitimacy verification is assigned to multiple *AA*s, each of which takes responsibility for the universal attribute set and is able to verify all of the user's attributes independently. After the successful verification, this *AA* will generate an intermediate key and send it to *CA*. The procedure of secret key generation and distribution is executed by the *CA* that generates the secret key associated with user's attribute set without any more verification. The secret key is generated using the intermediate key securely transmitted from an *AA* and the master secret key.

In our one-*CA*/multiple-*AA*s construction, *CA* participates in the key generation and distribution for security reasons: To enhance auditability of corrupted *AA*s, one *AA* cannot obtain the system's master secret key in case it can optionally generate secret keys without any supervision. Meanwhile, the introduction of *CA* for key generation and distribution is acceptable, since for a large-scale system, the most time-consuming workload of legitimacy verification is offloaded and shared among the multiple *AA*s, and the computation workload for key generation is very light. The procedure of key generation and distribution would be more efficient than other existing schemes.

To trace an *AA*'s misbehavior in the procedure of user legitimacy verification, we first find the suspected data consumer based on abnormal behavior detection, which is similar to the mechanisms used in [40] and [41]. For a suspected user, our scheme can trace the responsible *AA* who has falsely verified this user's attributes and illegitimately assigned secret keys to him/her.

B. Details of Our Proposed RAAC Scheme

1) System Initialization: Firstly, *CA* chooses two multiplicative cyclic groups $\mathbb{G}$(the parameter g is a generator of $\mathbb{G}$) and $\mathbb{G}_T$ with the same prime order p, and defines a binary map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ on $\mathbb{G}$. *CA* randomly chooses α, β, a and $b \in \mathbb{Z}_p$ as the master secret key. *CA* also randomly generates public keys for each attribute $Att_i, (i = 1, 2, \ldots, U)$: $h_1, h_2, \ldots, h_U \in \mathbb{G}$. Besides, let $H : (0,1)^* \to \mathbb{Z}_p$ be a hash function. The published public key is:

$$PK = \mathbb{G}_T, \mathbb{G}, H, g, g^a, e(g,g)^\alpha, h_1, \ldots, h_U$$

and the master secret key is:

$$MSK = \alpha, \beta, a, b$$

which implicitly exists in the system, and doesn't need to be obtained by any other entity.

Another task for *CA* in this operation is handling *AA*s' and users' registration. Here, *CA* generates a pair of keys (sk_{CA}, vk_{CA}) to sign and verify, in which, vk_{CA} is publicly known by each entity in the system. Each *AA* sends a registration request to *CA* during the ***System Initialization***. For each legal *AA*, *CA* assigns a unique identity $Aid \in \mathbb{Z}_p$, randomly chooses a private key $k_{Aid} \in \mathbb{Z}_p$, and computes its corresponding public key $PK_{Aid} = g^{k_{Aid}}$. Furthermore, *CA* generates a certificate $Cert_{Aid}$ which includes the public key PK_{Aid}, and sends it with the corresponding private key k_{Aid} to the *AA* with the identity Aid. Meanwhile, each user gets his/her Uid, private key k_{Uid} and $Cert_{Uid}$ from *CA*.

2) Encryption: The procedure of ***Encryption*** is performed by the data owner himself/herself. To improve the system's performance, the owner first chooses a random number $\kappa \in \mathbb{G}_T$ as the symmetric key and encrypts the plaintext message M using κ with the symmetric encryption algorithm. The encrypted data can be denoted as $E_\kappa(M)$. Then the owner encrypts the symmetric key κ using CP-ABE under the access policy $\mathbb{A}$ defined by himself/herself. The owner defines an easy expressed monotonic boolean formula. Following the method defined in [41], the owner can turn it to an LSSS access structure, which can be denoted as $(\mathcal{M}, \rho)$. Here, $\mathcal{M}$ is an $l \times n$ matrix, where l is the scale of a specific attribute set associated with a specific access policy and n is a variable that is dependent on the monotonic boolean formula definition and the LSSS turning method. The function ρ maps each row of $\mathcal{M}$ to a specific attribute, marked as $\rho(i) \in \{Att_1, Att_2, \ldots, Att_U\}$. A random secret parameter s is chosen to encrypt the symmetric key κ. To hide the parameter s, a random vector $\vec{v} = (s, y_2, y_3, \ldots, y_n) \in \mathbb{Z}_p^n$ is selected, where $y_2, y_3, \ldots, y_n$ are randomly chosen and used to share the parameter s. Each $\lambda_i = \mathcal{M}_i \vec{v}^\top$ is computed for $i = 1, 2, \ldots, l$, where $\mathcal{M}_i$ denotes the i-th row of the matrix $\mathcal{M}$. Owner randomly selects $r_1, r_2, \ldots, r_l \in \mathbb{Z}_p$ and uses the public key generated by *CA* to compute:

$$(C = \kappa e(g,g)^{\alpha s},\ C' = g^s,$$
$$\forall i = 1\ to\ l,\ C_i = (g^a)^{\lambda_i} \cdot h_{\rho(i)}{}^{-r_i},\ D_i = g^{r_i}).$$

The ciphertext CT can be denoted as the format shown in Fig. 2. Finally, the owner sends CT to the cloud server.

3) Key Generation and Distribution: This procedure is totally different from those existing CP-ABE schemes. It involves the given user, a selected *AA* and *CA*. We divide the procedure into the following 4 steps.

Description	$E_\kappa(M)$	$C, C', \{C_i, D_i\}_{i=1\ to\ l}$

Fig. 2. Format of owners' data in the cloud server.

- ***STEP 1:*** $U_j \rightarrow AA_i$. When a user U_j with the identity Uid_j makes a secret key request, the user selects an $AA(AA_i$ with the identity $Aid_i)$ by a certain scheduling algorithm and sends the $Cert_{Uid}$ to show the validity of his/her identity, along with some proofs to show that he/she has the attribute set that he/she claims to have.
- ***STEP 2:*** $AA_i \rightarrow CA$. The user legitimacy verification process may involve manual labor or verification protocols performed by AA_i. After successful verification, AA_i obtains the current timestamp value TS, computes $t_1 = H(Uid_j||TS||0)$ and $t_2 = H(Uid_j||TS||1)$, and generates an intermediate key IC_{Aid_i,Uid_j} as follows:

$$IC_{Aid_i,Uid_j} = \{K_x = h_x^{k_{Aid_i}t_1}, J_x = h_x^{t_2}\}_{\forall x \in S_j}$$

where S_j is the verified legitimate attribute set for the user with the identity Uid_j. Finally this AA securely sends the following message to CA:

$$\{Uid_j, Aid_i, S_j, IC_{Aid_i,Uid_j}, TS\}$$

- ***STEP 3&STEP 4:*** $CA \rightarrow AA_i \rightarrow U_j$. After receiving the message from the AA, CA first uses Aid_i to obtain the corresponding stored public key PK_{Aid_i}. Then CA checks whether the transmission delay is within the allowed time interval $\triangle T$. We assume that the current time is T'. If $T' - TS > \triangle T$, CA stops here and sends REJ to the AA. Otherwise, CA continues to compute $t_1 = H(Uid_j||TS||0)$, $t_2 = H(Uid_j||TS||1)$, and makes sure t_1 and t_2 haven't yet been re-used from the same user. This can prevent AA's collusion attack (We will discuss the collusion attack in Section VI.). CA continues to use its master secret key MSK to generate a secret key SK_j for the user as follows:

$$\begin{aligned}
K &= g^\alpha (PK_{Aid_i})^{a\beta t_1} g^{a\alpha t_2} = g^\alpha (g^{k_{Aid_i}})^{a\beta t_1} g^{a\alpha t_2},\\
L &= (PK_{Aid_i})^{\beta t_1} g^{\alpha t_2} = (g^{k_{Aid_i}})^{\beta t_1} g^{\alpha t_2},\\
\forall x &\in S_j, K'_x = K_x^\beta \cdot g^{b(t_1+t_2)} = h_x^{k_{Aid_i}\beta t_1} \cdot g^{b(t_1+t_2)},\\
K''_x &= J_x^\alpha \cdot g^{-b(t_1+t_2)} = h_x^{\alpha t_2} \cdot g^{-b(t_1+t_2)}.
\end{aligned}$$

To simplify the formulas, a parameter d is introduced:

$$d = k_{Aid_i}\beta t_1 + \alpha t_2 \tag{1}$$

Therefore, the formulas of the user's secret key SK_j can be simply denoted as:

$$\begin{aligned}
K &= g^\alpha \cdot g^{ad},\ L = g^d,\\
\forall x \in S_j : K'_x &= K_x^\beta \cdot g^{b(t_1+t_2)} = h_x^{k_{Aid_i}\beta t_1} \cdot g^{b(t_1+t_2)},\\
K''_x &= J_x^\alpha \cdot g^{-b(t_1+t_2)} = h_x^{\alpha t_2} \cdot g^{-b(t_1+t_2)}.
\end{aligned}$$

With the relay of AA_i, CA securely sends SK_j and TS to the user.

4) Decryption: The procedure of ***Decryption*** is performed by the user. A user can freely query and download any interested encrypted data from the public cloud storage. However, he/she cannot decrypt data unless his/her attribute set satisfies the access structure embedded in the ciphertext. For the user U, let $\mathcal{M}_U$ be a sub-matrix of $\mathcal{M}$, where each row of $\mathcal{M}_U$ corresponds to a specific attribute in U's attribute set S_U. Let $I \subset \{1, 2, \ldots, l\}$ denote $\{i : \rho(i) \in S_U\}$, and $\mathcal{M}_i$ denote the i-th row of $\mathcal{M}$.

If the user U's attribute set S_U satisfies the access structure $(\mathcal{M}, \rho)$, the vector $\vec{e} = (1, 0, \ldots, 0)$ is in the span of matrix $\mathcal{M}_U$, which means an appropriate parameter $\{\omega_i \in \mathbb{Z}_p\}_{i\in I}$ can be found to satisfy $\vec{e} = (\omega_1, \omega_2, \ldots, \omega_{|I|})M_U$.

The parameter $\{\omega_i\}_{i\in I}$ can further help the user to find the hidden secret parameter s:

$$\begin{aligned}
s &= (1, 0, \ldots, 0) \cdot (s, y_2, \ldots, y_n)^\top\\
&= \vec{e} \cdot \vec{v}^\top = (\omega_1, \omega_2, \ldots, \omega_{|I|})M_U \cdot \vec{v}^\top\\
&= (\omega_1, \omega_2, \ldots, \omega_{|I|}) \cdot \vec{\lambda_I}^\top = \sum_{i\in I} \omega_i \cdot \lambda_i.
\end{aligned} \tag{2}$$

where, $\vec{\lambda_I}$ denotes the sub-vector of $(\lambda_1, \lambda_2, \ldots, \lambda_l)$.

For any $x \in S_U$, the user computes

$$\begin{aligned}
KK_x &= K'_x \cdot K''_x\\
&= \left(h_x^{k_{Aid_i}\beta t_1} \cdot g^{b(t_1+t_2)}\right) \cdot \left(h_x^{\alpha t_2} \cdot g^{-b(t_1+t_2)}\right)\\
&= h_x^d,
\end{aligned} \tag{3}$$

and stores it. Using the parameter $\{\omega_i\}_{i\in I}$, the user further computes:

$$\begin{aligned}
C_U &= \frac{e(C', K)}{\prod_{i\in I}(e(C_i, L) \cdot e(D_i, KK_{\rho(i)}))^{\omega_i}}\\
&= \frac{e(g^s, g^\alpha \cdot g^{ad})}{\prod_{i\in I}(e((g^a)^{\lambda_i} \cdot h_{\rho(i)}{}^{-r_i}, g^d) \cdot e(g^{r_i}, h_{\rho(i)}{}^d))^{\omega_i}}\\
&= \frac{e(g, g)^{\alpha\cdot s} \cdot e(g, g)^{a\cdot s\cdot d}}{\prod_{i\in I} e(g, g)^{d\cdot a\cdot\lambda_i\cdot\omega_i}}\\
&= \frac{e(g, g)^{\alpha\cdot s} \cdot e(g, g)^{a\cdot s\cdot d}}{e(g, g)^{d\cdot a\cdot\sum_{i\in I}(\lambda_i\cdot\omega_i)}} = e(g, g)^{\alpha s}.
\end{aligned} \tag{4}$$

The user can further compute the symmetric key κ as follows:

$$\kappa = C/C_U = C/e(g, g)^{\alpha s}. \tag{5}$$

With the computed symmetric key κ, the user can further decrypt the ciphertext CT to obtain the final plaintext data M.

5) Auditing & Tracing: Each AA may generate an intermediate key for any attribute set associated with a specific user, and then CA can generate the secret key for this user without any more verification. However, AAs can be compromised and cannot be fully trusted. Meanwhile, the user legitimacy verification is conducted by manual labor, and therefore AAs may maliciously or incorrectly generate an intermediate key for an unverified attribute set. A malicious user will try any possible means to gain the secret key associated with the

specific attribute set to obtain the data access permission. Under this assumption, the user would often show abnormal behaviors. Usually, we need to hold the accountability of *AA*s to prevent the compromised or misbehaved ones from freely generating secret keys for malicious users. Like user accountability addressed in [40], [41], and [43], we assume that we have appropriate techniques to detect users' abnormal behaviors.

The procedure of ***Auditing & Tracing*** is periodically performed or event-triggered by *CA* to mandatorily ask a suspected user to securely submit K'_x of a given attribute, L and TS in his/her gained secret key. In order to continue to obtain data, users have to cooperate to perform the process correctly. However, in order to deceive *CA*, a suspected user still has the motivation to submit a secret key component that doesn't belong to him/her. Thus, to implement an effective tracing, *CA* must confirm the received secret key components really belong to the given user. Based on the reasons mentioned above, the tracing method should be executed as the following two sub-procedures.

- ***Secret key ownership confirming***. This procedure is executed to confirm that the received secret key component really belongs to the user who has submitted it. We assume that the user is U_j with the identity Uid_j. *CA* randomly selects a suspected attribute x in S_j, and asks U_j to securely submit his/her secret key components K'_x, L and TS. Then *CA* computes $t_1 = H(Uid_j||TS||0)$, $t_2 = H(Uid_j||TS||1)$ and $K_x^{\triangle} = h_x^{\alpha t2} \cdot g^{-b(t_1+t_2)}$, and confirms whether the following equation holds:

$$e(h_x, L) = e(g, K'_x K_x^{\triangle}).$$

  If it holds, *CA* will further continue to execute the next sub-procedure. Otherwise, it indicates that the suspected user doesn't correctly submitted his/her own secret key components and the user will receive a severe punishment, such as kicking the user out of the system.
- **AA *Tracing***. This procedure is executed to trace and confirm which *AA* has generated the suspected user's secret key. *CA* takes its master secret key MSK to recover the public key associated with a specific *AA* as follows:

$$PK = (L \cdot g^{-\alpha t_2})^{1/\beta t_1} = g^{k_{Aid_i} \beta t_1 / \beta t_1} = g^{k_{Aid_i}}. \quad (6)$$

  CA uses PK as an index to search its storage for the responsible *AA*. If some *AA* with the identity Aid_i owns a public key that is equal to PK, it means that *AA* has maliciously or incorrectly verified the legitimacy of this user. The found *AA* should implement security enhancement or be kicked out of the system as a severe punishment.

VI. Security Analysis

A. Data Confidentiality

To prove the data confidentiality of RAAC, we use the following theorem:

Theorem 1: When the decisional q-parallel BDHE assumption holds, no adversary can use a polynomial-time algorithm to selectively break RAAC with a challenge matrix of size $l^* \times n^*$, where $l^*, n^* \leq q$.

Proof: When $k_{Aid_i}\beta t_1 + \alpha t_2$ is simplified as d, and K'_x and K''_x are combined by a multiplication operation, the user's secret key can be simplified as:

$$K = g^{\alpha} \cdot g^{ad}, \quad L = g^d, \ \forall x \in S_j, \ KK_x = K'_x \cdot K''_x = h_x^d,$$

where d can be seen as a random number based on different t_1 and t_2. Then the proof of *Theorem* 1 is similar to that in [22]. Interested readers are referred to read [22] for more details about the proof. □

If the cloud server is curious about the plaintext, it would not have any advantages compared with outside malicious users, as the cloud server doesn't participate in any user's secret key generation. All it does is storing the ciphertext, and it thus find no way to obtain the plaintext.

B. User Collusion Resistance

Like the technique proposed in [22] which could prevent user collusion attack, each user in our system is assigned a global unique identity Uid, and each time when performing the procedure of key generation and distribution, all the secret key components are associated with a unique value d, which a user is not able to compute. Thus, it is impossible for two or more users to collude and decrypt the ciphertext.

Moreover, it's important to note that, in order to introduce the auditing and tracing mechanism, we divide the original KK_x into two parts, namely K'_x and K''_x. If there is no $g^{b(t_1+t_2)}$ in K'_x and no $g^{-b(t_1+t_2)}$ in K''_x, the malicious user can easily gain $h_x^{k_{Aid_i}\beta}$ and h_x^{α}. Then, this internal attacker can gather the two secret key components about the other attributes different from his/her own set. If the attacker obtains K and L from the same *AA* with gathered information, he/she can further use his/her own t_1 and t_2 to compute $K'_x = (h_x^{k_{Aid_i}\beta})^{t_1}$ and $K''_x = (h_x^{\alpha})^{t_2}$. In this way, colluded with other users, the malicious user can easily obtain additional access privilege that he/she is not supposed to have. However, because b and g^b are kept secure, and there is $g^{b(t_1+t_2)}$ in K'_x and $g^{-b(t_1+t_2)}$ in K''_x, malicious users find no way to realize the above mentioned malicious action.

C. No Ultra Vires for AA

As *AA*s cannot be fully trusted, it's possible that two or more *AA*s collude with each other to gain more information in the system. If two *AA*s collude to respectively generate intermediate keys for the same user at the same time, they can have the same t_1 and t_2. We assume that these two colluded *AA*s are AA_{i1} with Aid_{i1} and AA_{i2} with Aid_{i2}, and the user is U_j with the identity Uid_j. In STEP 3 of the procedure of key generation and distribution, if AA_{i1} is chosen, the secret key can be computed as follows:

$$K_1 = g^{\alpha} \cdot g^{a(k_{Aid_{i1}}\beta t_1 + \alpha t_2)}, \quad (7a)$$

$$L_1 = g^{k_{Aid_{i1}}\beta t_1 + \alpha t_2}, \quad (7b)$$

$$\forall x \in S_j, KK_{x1} = h_x^{k_{Aid_{i1}}\beta t_1 + \alpha t_2}, \quad (7c)$$

and if AA_{i2} is chosen, the secret key can be computed as follows:

$$K_2 = g^{\alpha} \cdot g^{a(k_{Aid_{i2}}\beta t_1 + \alpha t_2)}, \tag{8a}$$

$$L_2 = g^{k_{Aid_{i2}}\beta t_1 + \alpha t_2}, \tag{8b}$$

$$\forall x \in S_j, KK_{x2} = h_x^{k_{Aid_{i2}}\beta t_1 + \alpha t_2}, \tag{8c}$$

If *CA* has no verification to check whether TS is in an allowed time interval, meanwhile, t_1 and t_2 are different and haven't yet been reused in the same time, these two secret keys can both be issued. Combining these two secret keys using the division operation, AA_{i1} and AA_{i2} can collude to obtain as follows:

$$K_1/K_2 = g^{a(k_{Aid_{i1}} - k_{Aid_{i2}})\beta t_1}, \tag{9a}$$

$$L_1/L_2 = g^{(k_{Aid_{i1}} - k_{Aid_{i2}})\beta t_1}, \tag{9b}$$

$$\forall x \in S_j, KK_{x1}/KK_{x2} = h_x^{(k_{Aid_{i1}} - k_{Aid_{i2}})\beta t_1}, \tag{9c}$$

Furthermore, based on Equation 9b and Equation 7b, the colluded *AA*s can compute g^{α} as follows:

$$\left(L_1 / \left(L_1/L_2\right)^{\frac{k_{Aid_{i1}}}{k_{Aid_{i1}} - k_{Aid_{i2}}}} \right)^{\frac{1}{t_2}}.$$

Based on the above method, colluded *AA*s can obtain g^{α}. Also, based on similar methods, colluded *AA*s can obtain $g^{a\beta}$, $g^{a\alpha}$, h_x^{α} and $h_x^{\beta} (\forall x \in S)$. Now the malicious *AA* can masquerade as another *AA* with a random private key k, and issues any key for a user with any attribute set. Thus, the two colluded *AA*s can illegally obtain *CA*'s privilege. However, based on the analysis above, if *CA* maintains the state of used t_1 and t_2 for the same user in a given time interval, the secret keys from different *AA*s will have different t_1 and t_2, and the above attack method can be thwarted.

D. Assurance of AA*s' Auditablity*

In the heterogeneous single-*CA*/multi-*AA*s framework, we must ensure that each *AA* can be audited so that when a user's abnormal behavior happens, we can trace which *AA* has verified this user's legitimacy and issued intermediate key for him/her. In our scheme, we enforce this assurance by the construction of the intermediate key. For user U_j, when he/she is legitimacy verified by an *AA*(e.g., AA_i with the identity Aid_i), the intermediate key $\{K_x = h_x^{k_{Aid_i} t_1}, J_x = h_x^{t_2}\}_{\forall x \in S_j}$ contains both attributes that AA_i has verified and AA_i's identity implicitly. Here, k_{Aid_i} is a private key securely owned by AA_i, and other *AA*s can only generate this intermediate key with his/her own private key.

To generate a secret key, using Aid_i as an index to get the public key PK_{Aid_i}, *CA* generates K and L to match the K'_x, $x \in S_j$, where $K'_x = K_x^{\beta} \cdot g^{b(t_1+t_2)}$, $x \in S_j$. Thus, the issued secret key implicitly contains AA_i's private key k_{Aid_i}, which can be regarded as a signature when *CA* conducts auditing.

If AA_i tries to get rid of the responsibility by sending $Aid_j, i \neq j$ to CA, the CA will generate K and L of SK according to the elements associated with PK_{Aid_j}. The generated K and L are related to k_{Aid_j}, but K'_x, which is generated on the intermediate key, is related to k_{Aid_i}. The secret key made by their combination cannot be used to decrypt any ciphertexts, unless AA_i knows k_{Aid_j} and generates an intermediate key with k_{Aid_j}. Unless AA_j is compromised, the private key k_{Aid_j} would not be leaked to AA_i. Therefore, no AA can evade its responsibility by faking to be someone else.

VII. Performance Analysis

As we have mentioned, in reality, the tedious procedure of user legitimacy verification is much more complicated than secret key generation. In our scheme, the load of legitimacy verification is shared among multiple *AA*s, while a much lighter computational task is assigned to the single *CA*. Thus, the efficiency of key distribution is improved. More Specifically, multiple *AA*s are standby for the legitimacy verification in the system. When there is a key request, an idle *AA* is selected by a scheduling algorithm to perform the verification and other *AA*s are standby to serve the subsequent user requests.

We give the theoretical performance analysis as the following steps. Firstly, we model our system in queueing theory, and then we analyze the state probabilities to obtain the two important factors, the mean failure probability and the average waiting time for users. Finally, to show the significant performance improvement of our proposed RAAC, we compares it with single-*AA* system. It's important to note that, the comparison between RAAC and multi-authority systems [11], [31] is similar, since each authority independently manages a disjoint attribute subset. When a user requests secret keys with regard to one certain attribute subset, he/she has to go to the only and exclusive authority that issues secret keys with that attribute subset. The queue condition is just the same as the one in single- authority schemes.

A. Modeling in Queuing Theory

For simplicity, we assume there is a central coordinator which assigns users' key requests to *AA*s. The coordinator maintains each *AA*'s state with the boolean value of $0/1$, where state 0 indicates that the *AA* is available to conduct verification, and state 1 indicates the *AA* is occupied and is not available right now. Each time the coordinator assigns a key request to an *AA* with the state 0. If all *AA*s are busy, the new users who are requesting the secret keys will wait in a queue to be served. The coordinator can adopt *First Come First Service (FCFS)* algorithm to serve the arriving users.

It's important to note that some other strategies can also be adopted in our architecture, such as a user arriving at a nearest *AA* according to his/her knowledge and decision. Thus, each *AA* may separately maintain a queue of its own. However, this model may not achieve load balance as some *AA*s may be unoccupied while other *AA*s are always busy in serving users' requests. Therefore, we introduce a central coordinator and adopt a single arrival queue as our strategy. The queueing model of our system is shown in Fig. 3, and can be treated as a Markov process.

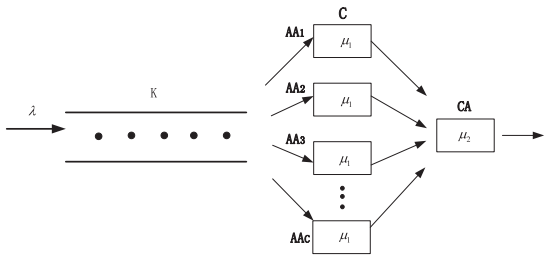


Fig. 3. The queue model with single-*CA*/multi-*AA*s.

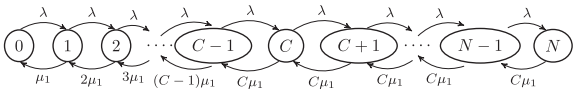


Fig. 4. $M/M/C/N/\infty$ state transition diagram.

The central coordinator is deployed at the entrance of the system to monitor each *AA*'s state (occupied/unoccupied) and assign each arriving users to an unoccupied *AA*. Furthermore, we model our system as follows. On *AA*s' side, the queueing model can be described as $M/M/C/N/\infty$, where C is the number of *AA*s, N is the capacity of our system and $N = C + K$ (K is the queue length that indicates the maximum number of the queued users.). Here, the first M describes that arrivals of key requests follow a Poisson process in the system, and the second M means the verification service times are exponentially distributed. ∞ means the source of key requests is infinite.

When there are N users in the system, other new arrivals of users' requests will be rejected. This property can ensure that a user will not wait in the queue for an irrationally long time. On *CA*'s side, the queueing model can be described as $M/M/1$.

The following assumptions are made to describe our system.

1) Assumption 1. The instant user request arrival event constitutes a stationary Poisson process with the parameter λ.
2) Assumption 2. For each *AA*, the service time of different individual users are independent and identically distributed exponential random variables, in which the mean value is $1/\mu_1$.
3) Assumption 3. For *CA*, the service time of individual users are independent and identically distributed exponential random variables, in which the mean value is $1/\mu_2$.

B. State Probabilities

*AA*s' side is the most influential part of our system's performance, so we mainly analyze its state probabilities.

The state transition of the queueing model with multiple *AA*s is shown in Fig. 4. Arrivals occur at rate λ according to a Poisson process and move the process from state i to $i+1$. For each *AA*, the service time of different individual users are independent, and follow negative exponential distribution with parameter μ_1. Therefore we can get:

$$\lambda_n = \begin{cases} \lambda, & n \in [0, N-1] \\ 0, & n \geq N \end{cases} \tag{10}$$

and

$$\mu_{1n} = \begin{cases} n\mu_1, & n \in [0, C) \\ C\mu_1, & n \in [C, N-1] \end{cases} \tag{11}$$

Since our queueing system is Markovian, the state probabilities can be described by a set of Chapman-Kolmogorov difference equations in the steady state using standard techniques. Let $\rho = \lambda/(\mu_1)$ and $\rho_C = \rho/C$, we obtain the steady-state probability distribution as follows:

$$p_n = \begin{cases} \dfrac{\rho^n}{n!} p_0, & n \in [0, C) \\ \dfrac{\rho^n}{C! C^{n-C}} p_0, & n \in [C, N] \end{cases} \tag{12}$$

where

$$p_0 = \left(\sum_{n=0}^{C-1} \frac{\rho^n}{n!} + \frac{C^C}{C!} \sum_{n=C}^{N-1} \rho_C^n \right)^{-1} \tag{13}$$

Then we can get the mean queue length at multi-*AA*s' side as:

$$\begin{aligned} L_q &= \sum_{j=C}^{N+C-1} (j-C) p_j \\ &= \sum_{j=C}^{N+C-1} j \cdot p_j - \sum_{j=C}^{N+C-1} C \cdot p_j \\ &= \sum_{j=0}^{N+C-1} j \cdot p_j - \sum_{j=0}^{C-1} j \cdot p_j - \sum_{j=C}^{N+C-1} C \cdot p_j \\ &= L - \sum_{j=0}^{C-1} j \cdot p_j - C \cdot (1 - \sum_{j=0}^{C-1} p_j) \\ &= L - C - \sum_{j=0}^{C-1} (j-C) \cdot p_j \\ &= L - C - p_0 \cdot \sum_{j=0}^{C-1} \frac{(j-C) \cdot \rho^n}{j!} \end{aligned} \tag{14}$$

Meanwhile, because the system can only accommodate N users at anytime, we can compute the effective arrival rate as follows:

$$\lambda_{eff} = \lambda(1 - p_N) \tag{15}$$

By Little's law, we can compute the average sojourn time and the average waiting time for users at multi-*AA*s' side as follows:

$$W = L/\lambda_e, W_q = L_q/\lambda_e = W - 1/\mu_1 \tag{16}$$

To sum up with the average waiting time at the single-*CA*'s side, the average waiting time W_q' of users is:

$$W_q' = W_q + \frac{\frac{\lambda_{eff}}{\mu_2}}{\mu_2 \cdot (1 - \frac{\lambda_{eff}}{\mu_2})} \tag{17}$$

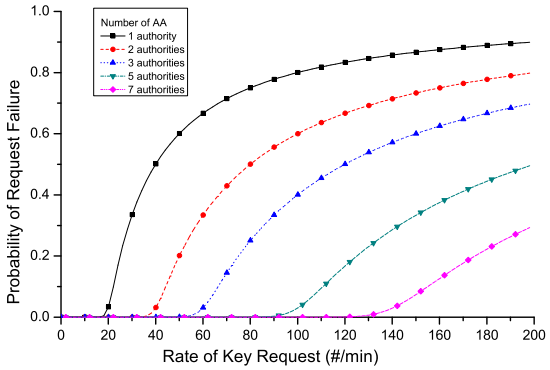


Fig. 5. The failure rate in RAAC with $\mu_1 = 20/min$, $\mu_2 = 200/min$ and $K = 30$.

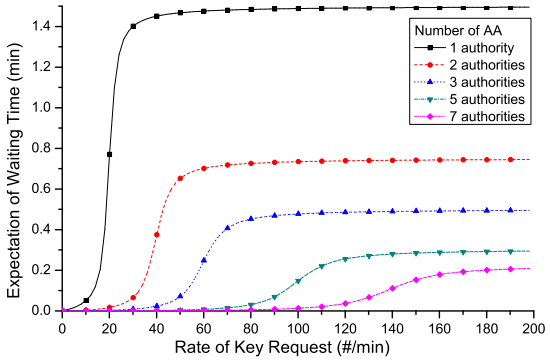


Fig. 6. The average waiting time in RAAC with $\mu_1 = 20/min$, $\mu_2 = 200/min$ and $K = 30$.

C. Numerical Evaluation

As we described above, when the queue model with multi-*AA*s is filled with N users, which means that K users are waiting in the queue and all C *AA*s are occupied, the newly arriving users are rejected. This means those new users would fail to get the secret keys. We analyze the probability of failure to show how to lower this failure rate with more *AA*s. Meanwhile, we analyze the average waiting time W_q' of users in our system.

Based on the emulation of the scheme in [18], the average time of generating a secret key for an attribute is about 35ms (on 64-bit AMD 3.7 GHz workstation). Furthermore, we assume that users possess 10 attributes on average and the verification takes tenfold amount of time of that of the key generation. The parameters are set as: $\mu_1 = 20/min$, $\mu_2 = 200/min$, and $K = 30$. The performance analysis in terms of the average failure rate and the average waiting time is shown in Fig. 5 and Fig. 6, respectively.

Fig. 5 shows the failure rate versus the arrival rate and the number of *AA*s. From Fig. 5, we can see that when the average failure rate of single-authority scheme is less than 5%, it can only support an arrival rate of less than 20/min. With increasing the number of *AA*s, the system can greatly increase its service capacity with the support of a greater arrival rate at the same failure rate. If we employ 7 *AA*s, the system can support the arrival rate of up to 150/min, with the failure rate of less than 5%. It is easy to infer that we can build our system based on the observation of key request rate, and then use an appropriate number of *AA*s to provide high equality service.

TABLE I
STORAGE OVERHEAD

Entity	*CA*	*AA*	Owner	User
[22]	Not exist	$2\|p\|$	$(U+4)\|p\|$	$(N_{Uid}+2)\|p\|+\|C_u\|$
[11]	$2\|p\|$	$3\|p\|$	$(U+3N_A+5)\|p\|$	$(N_{Uid}+3k_{max}+1)\|p\|$ $+\|C_u\|$
[15]	$2\|p\|$	$\|p\|+\|C_a\|$	$(U+7)\|p\|$	$(N_{Uid}+2)\|p\|+\|C_u\|$
Ours	$(2N_A+5)\|p\|$	$\|p\|+\|C_a\|$	$(U+6)\|p\|$	$(2N_{Uid}+2)\|p\|$ $+\|C_u\|+\|TS\|$

Fig. 6 shows the average waiting time versus the arrival rate and the number of *AA*s when $\mu_1 = 20/min$, $\mu_2 = 200/min, K = 30$. From the figure, we can see that the average waiting time increases rapidly with the increase of arrival rate when the arrival rates are low. But later the average waiting time will become steady because newly arrival users will be rejected by the system due to the limit length of waiting queue. More specifically, with single *AA*, the average waiting time increases rapidly and reaches 1.5 min, which is unbearable.

Whereas, with 7 *AA*s, the average waiting time is about $15s$. Moreover, from Fig. 5, with the arrival rate less than 150, the failure rate is less than 5%. Although using more working *AA*s brings larger configuration cost, by combining the failure rate and the average waiting time, we can assure that the configuration of multiple *AA*s can provide secret key generation service with high quality as well as low cost.

D. Storage, Communication and Computation Overhead Analysis

In this section, we conduct performance analysis in terms of storage, communication and computation overhead in each process, among our proposed RAAC, Waters' scheme [22], DAC-MACS [11] and TMACS [15]. Waters' scheme involves only one *AA* in charge of key generation and distribution. In DAC-MACS, each of the multiple *AA*s manages a disjoint attribute set, and in TMACS, all *AA*s manage the universal attribute set as our RAAC.

For clarity of description, we have the following definitions: Let $|p|$ be the size of element in the groups with prime order p, and U be the number of attributes. N_u denotes the number of users in the system. N_{Uid} denotes the average number of attributes owned by users, and N_c denotes the average number of attributes inside ciphertexts. Let N_A be the number of *AA*s. $|C_u|$ denotes the average size of a user's certificate, and $|C_a|$ denotes the average size of an *AA*'s certificate. In addition, for DAC-MACS, U_{Aid_i} ($\sum_{i=1}^{N_A} U_{Aid_i} = U$) denotes the number of attributes the *AA* (with the Aid_i) manages, and N_{Uid,Aid_i} ($N_{Uid,Aid_i} \leq N_{Uid}$) means the average number of attributes the user has in *AA* with the Aid_i. k_{max} denotes that the user may have attribute sets from at most k_{max} ($k_{max} \leq N_A$) *AA*s on average. $N_{A,c}$ denotes the average number of *AA*s related to ciphertexts.

TABLE II
COMMUNICATION OVERHEAD

Process	System Initialization			Key Generation &Distribution			Encryption/Decryption	Auditing&Tracing
Entity	*CA*	*AA*	User	*CA*	*AA*	User	Owner/User	*CA*/ User
[22]	Not exist	$N_u\|C_u\|$	$\|C_u\|$	Not exist	$(N_{Uid}+2)\|p\|$ $+\|C_u\|$	$(N_{Uid}+2)\|p\|$ $+\|C_u\|$	$(2N_c+2)\|p\|$	N/A
[11]	$(3N_u+N_A)\|p\|$ $+N_u\|C_u\|$	$\|p\|$	$3\|p\|+\|C_u\|$	N/A	$(N_{Uid,Aid_i}+3)\|p\|$ $+\|C_u\|$	$(N_{Uid}+3k_{max})\|p\|$ $+k_{max}\|C_u\|$	$(3N_c+N_{A,c}+2)\|p\|$	N/A
[15]	$(N_u+N_A)\|p\|$ $+N_u\|C_u\|+N_A\|C_a\|$	$(2N_A-1)\|p\|$ $+\|C_a\|$	$\|p\|+\|C_u\|$	N/A	$(N_{Uid}+2)\|p\|$ $+\|C_u\|$	$(N_{Uid}+2)t\|p\|$ $+t\|C_u\|$	$(2N_c+2)\|p\|$	N/A
Ours	$(N_u+N_A)\|p\|$ $+N_u\|C_u\|+N_A\|C_a\|$	$\|p\|+\|C_a\|$	$\|p\|+\|C_u\|$	$(4N_{Uid}+5)\|p\|+\|TS\|$	$(7N_{Uid}+6)\|p\|$ $+2\|TS\|+\|C_u\|$	$(2N_{Uid}+2)\|p\|$ $+\|TS\|+\|C_u\|$	$(2N_c+2)\|p\|$	$2\|p\|+\|TS\|$

TABLE III
COMPUTATION OVERHEAD

Phase	System Initialization		Key Generation&Distribution			Encryption	Decryption	Auditing	
Entity	*CA*	*AA*	*CA*	*AA*	User	Owner	User	CA	User
[22]	Not exist	$O(U)$	Not exist	$O(N_{Uid})$	$O(1)$	$O(N_c)$	$O(N_c)$	N/A	N/A
[11]	$O(N_u+N_A)$	$O(U_{Aid_i})$	N/A	$O(N_{Uid,Aid_i})$	$O(1)$	$O(N_c+N_{A,c})$	$O(1)^*$	N/A	N/A
[15]	$O(U+N_u+N_A+t)$	$O(N_A)$	N/A	$O(N_{Uid})$	$O(tN_{Uid})$ (Only once)	$O(N_c)$	$O(N_c)$	N/A	N/A
Ours	$O(U+N_u+N_A)$	$O(1)$	$O(N_{Uid})$	$O(N_{Uid})$	$O(N_{Uid})$ (Only once)	$O(N_c)$	$O(N_c)$	$O(1)$	$O(1)$

* There are the computation overhead of $O(N_c+N_{A,c})$ on Cloud Server.

1) Storage Overhead: The storage overhead analysis on each entity is shown in TABLE I. Compared with other schemes, in RAAC, *CA* has an extra overhead of $2N_A|p|$, which is caused by the storage of public keys PK_{Aid} and Aid of each *AA*. That extra storage is used for key generation and auditing. Since N_A will not be very large, the additional cost is affordable. From the perspective of *AA*, TMACS and RAAC both store one more certificate than Waters' scheme, since TMACS and RAAC explicitly take the distribution of *AA*'s certificate into account when *AA* registers at *CA*. To be noted, for a fair comparison, we do not count in the storage of attribute version keys in DAC-MACS. In RAAC, each user's storage overhead is larger, mainly because the user needs to store a copy of K'_x for potential auditing. The additional storage will not consume too much, thus we think it is acceptable for users.

2) Communication Overhead: Table II shows comparison result about communication overhead on *CA*, each *AA*, *owner*, and *user* of RAAC, Waters' scheme, DAC-MACS and TMACS. During *System Initialization* phase, all multi-authority schemes introduce more communication overhead in *CA*, but it is a necessary sacrifice for *AA*'s registration. Besides the registration overhead, RAAC does not introduce much extra cost, compared with other multi-authority schemes. In *Key Generation&Distribution* phase of RAAC, the communication overhead on *CA* and *AA* is obviously larger than others due to the fact that *CA* must participate in the key generation and *AA* must communicate with *CA*. The extra overhead is introduced to guarantee the auditability of *AA*s, which is worth. Furthermore, from the view of users, RAAC gets a better performance than TMACS in *Key Generation&Distribution* phase because the latter one enforces users to communicate with t authorities. Our RAAC also shows an advantage over DAC-MACS in *Encryption* and *Decryption* phase, because the ciphertext length of RAAC is smaller.

3) Computation Overhead: The computation overhead is analyzed as shown in Table III. As we have mentioned in Section VII-D2, in *System Initialization* phase, multi-authority settings inevitably introduce some overhead for *AA* registration, which is acceptable. In *Key Generation&Distribution* phase, after receiving secret keys from *AA*s, users in TMACS and RAAC need to recompute their secret keys to decrypt. However, the recomputation can be done once and then users can store them. To decrypt a ciphertext, RAAC is as efficient as Waters' scheme. In *Decryption* phase, it seems that DAC-MACS is more efficient for users. However, the reason is that DAC-MACS adopts the outsourcing technique to let the cloud server do most decryption for users. If we look at the total computation overhead of the cloud server and the user, RAAC is at least as efficient as DAC-MACS.

VIII. DISCUSSION OF A HYBRID MODEL CONSTRUCTION

This paper has presented RAAC scheme based on a single-authority algorithm, where the authority is replaced by one *CA* and multiple *AA*s (we rename the combination as *CA/AA*s unit). Practically, RAAC can also be built in many traditional multi-authority settings, e.g., [11], [31], where the attribute authorities (referred to as *TAAs* to distinguish *AA*s of traditional multi-authority settings from *AA*s in *CA/AA*s unit of RAAC) manage disjoint attribute subsets. In what follows, we show how our proposed scheme works with DAC-MACS [11].

At a high level, the *CA/AA*s unit acts as an individual *TAA* to manage a disjoint attribute subsets. Besides, there is a root central authority to play the role of *CA* in DAC-MACS (denoted as *RCA* for distinction). Fig. 7 briefly depicts the hybrid architecture. The secret key of user j from a certain CA/AAs unit k ($SK_{j,k}$) is:

$$K_{j,k} = g^{\alpha_k} \cdot g^{au_j} \cdot g^{at_{j,k}/\beta_k}, L_{j,k} = g^{\beta_k t_{j,k}}, R_{j,k} = g^{at_{j,k}},$$
$$\forall x_k \in S_{j,k} : K_{j,x_k} = g^{\beta_k \gamma_k t_{j,k}} \cdot H(x_k)^{\gamma_k \beta_k u_j},$$

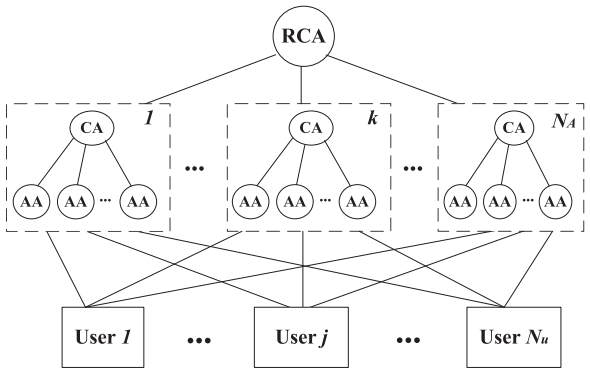


Fig. 7. A hybrid architecture.

where α_k, β_k and γ_k are master secret keys of *CA*. It is noted that the components for decryption outsource and attribute revocation, z_j and υ_{x_k}, respectively, are omitted in our discussion. The remaining will discuss how we generate the same secret key format as DAC-MACS with *CA/AA*s unit, while preserve the auditability of *AA*s.

To output $SK_{j,k}$, an *AA* in *CA/AA*s k generates an intermediate key as:

$$IC_{Aid_i,Uid_j,k} = \{K_{x_k} = h_{x_k}^{k_{Aid_i} t_1}, J_{x_k} = h_{x_k}^{t_2}\} \forall x_k \in S_{j,k},$$

where, $h_{x_k} = g \cdot H(x_k)^{u_j}$ acts the same role of h_x in RAAC. After receiving the intermediate key, *CA* computes:

$$\forall x_k \in S_{j,k},$$
$$K'_{x_k} = \left(h_{x_k}^{k_{Aid_i}\theta_k t_1} \cdot g^{\phi_k(t_1+t_2)}\right)^{\gamma_k\beta_k},$$
$$K''_{x_k} = \left(h_{x_k}^{\varphi_k t_2} \cdot H(x_k)^{u_j\left(1-(k_{Aid_i}\theta_k t_1+\varphi_k t_2)\right)} g^{-\phi_k(t_1+t_2)}\right)^{\gamma_k\beta_k},$$

where, φ_k, θ_k and ϕ_k are the master secret keys maintained by that *CA* in *CA/AA*s unit k). Then, the user recomputes secret key for each attribute in the set:

$$\forall x_k \in S_{j,k} : K_{j,x_k} = K'_{x_k} \cdot K''_{x_k} = g^{\beta_k\gamma_k d_k} \cdot H(x_k)^{\gamma_k\beta_k u_j}.$$

where, $d_k = (k_{Aid_i}\theta_k t_1 + \varphi_k t_2)$, which acts as $t_{j,k}$ in DAC-MACS. Other parts of secret key $K_{j,k}$, $L_{j,k}$, $R_{j,k}$ are computed by *CA* in the same way as DAC-MACS does.

The above output $SK_{j,k}$ has the same format as DAC-MACS, which shows the construction can realize a traditional multi-authority structure. Furthermore, as $g \cdot H(x_k)^{u_j}$ plays the same role as h_x for every attribute x in *CA*'s management domain, and $L_{j,k}$ plays the similar role of L in RAAC, the *Auditing&Tracing* mechanism can be preserved.

IX. Conclusion

In this paper, we proposed a new framework, named RAAC, to eliminate the single-point performance bottleneck of the existing CP-ABE schemes. By effectively reformulating CP-ABE cryptographic technique into our novel framework, our proposed scheme provides a fine-grained, robust and efficient access control with one-*CA*/multi-*AA*s for public cloud storage. Our scheme employs multiple *AA*s to share the load of the time-consuming legitimacy verification and standby for serving new arrivals of users' requests.

We also proposed an auditing method to trace an attribute authority's potential misbehavior. We conducted detailed security and performance analysis to verify that our scheme is secure and efficient. The security analysis shows that our scheme could effectively resist to individual and colluded malicious users, as well as the honest-but-curious cloud servers. Besides, with the proposed auditing & tracing scheme, no *AA* could deny its misbehaved key distribution. Further performance analysis based on queuing theory showed the superiority of our scheme over the traditional CP-ABE based access control schemes for public cloud storage.

References

[1] P. Mell and T. Grance, "The NIST definition of cloud computing," Nat. Inst. Standards Technol., Gaithersburg, MD, USA, Tech. Rep. 800-145, 2011.

[2] Z. Fu, K. Ren, J. Shu, X. Sun, and F. Huang, "Enabling personalized search over encrypted outsourced data with efficiency improvement," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 9, pp. 2546–2559, Sep. 2016.

[3] Z. Fu, X. Sun, S. Ji, and G. Xie, "Towards efficient content-aware search over encrypted outsourced data in cloud," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2016, pp. 1–9.

[4] K. Xue and P. Hong, "A dynamic secure group sharing framework in public cloud computing," *IEEE Trans. Cloud Comput.*, vol. 2, no. 4, pp. 459–470, Oct. 2014.

[5] Y. Wu, Z. Wei, and R. H. Deng, "Attribute-based access to scalable media in cloud-assisted content sharing networks," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 778–788, Jun. 2013.

[6] J. Hur, "Improving security and efficiency in attribute-based data sharing," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 10, pp. 2271–2282, Oct. 2013.

[7] J. Hur and D. K. Noh, "Attribute-based access control with efficient revocation in data outsourcing systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 7, pp. 1214–1221, Jul. 2011.

[8] J. Hong, K. Xue, W. Li, and Y. Xue, "TAFC: Time and attribute factors combined access control on time-sensitive data in public cloud," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2015, pp. 1–6.

[9] Y. Xue, J. Hong, W. Li, K. Xue, and P. Hong, "LABAC: A location-aware attribute-based access control scheme for cloud storage," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2016, pp. 1–6.

[10] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Advances in Cryptology—EUROCRYPT*. Berlin, Germany: Springer, 2011, pp. 568–588.

[11] K. Yang, X. Jia, K. Ren, and B. Zhang, "DAC-MACS: Effective data access control for multi-authority cloud storage systems," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr. 2013, pp. 2895–2903.

[12] J. Chen and H. Ma, "Efficient decentralized attribute-based access control for cloud storage with user revocation," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2014, pp. 3782–3787.

[13] M. Chase and S. S. M. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *Proc. 16th ACM Conf. Comput. Commun. Secur. (CCS)*, 2009, pp. 121–130.

[14] M. Lippert, E. G. Karatsiolis, A. Wiesmaier, and J. Buchmann, "Directory based registration in public key infrastructures," in *Proc. 4th Int. Workshop Appl. PKI (IWAP)*, 2005, pp. 17–32.

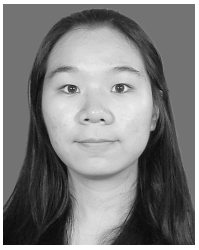
[15] W. Li, K. Xue, Y. Xue, and J. Hong, "TMACS: A robust and verifiable threshold multi-authority access control system in public cloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 27, no. 5, pp. 1484–1496, May 2016.

[16] S. Chokhani, W. Ford, R. Sabett, C. Merrill, and S. Wu, *Internet x.509 Public Key Infrastructure Certificate Policy and Certification Practices Framework*, document IETF RFC, RFC3647, 2003.

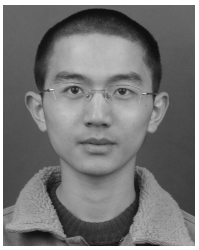
[17] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur. (CCS)*, 2006, pp. 89–98.

[18] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy (S&P)*, May 2007, pp. 321–334.
[19] V. Goyal, A. Jain, O. Pandey, and A. Sahai, "Bounded ciphertext policy attribute based encryption," in *Automata, Languages and Programming*. Berlin, Germany: Springer, 2008, pp. 579–591.
[20] L. Cheung and C. Newport, "Provably secure ciphertext policy ABE," in *Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS)*, 2007, pp. 456–465.
[21] A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters, "Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption," in *Advances in Cryptology—EUROCRYPT*. Berlin, Germany: Springer, 2010, pp. 62–91.
[22] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Proc. Int. Workshop Public Key Cryptogr.*, 2011, pp. 53–70.
[23] K. Emura, A. Miyaji, A. Nomura, K. Omote, and M. Soshi, "A ciphertext-policy attribute-based encryption scheme with constant ciphertext length," in *Information Security Practice and Experience*. Berlin, Germany: Springer, 2009, pp. 13–23.
[24] R. Ostrovsky, A. Sahai, and B. Waters, "Attribute-based encryption with non-monotonic access structures," in *Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS)*, 2007, pp. 195–203.
[25] S. Hohenberger and B. Waters, "Online/offline attribute-based encryption," in *Public-Key Cryptography—PKC*. Berlin, Germany: Springer, 2014, pp. 293–310.
[26] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proc. 5th ACM Symp. Inf., Comput. Commun. Secur. (ASIACCS)*, 2010, pp. 261–270.
[27] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of ABE ciphertexts," in *Proc. USENIX Secur. Symp.*, 2011, p. 34.
[28] J. Shao, R. Lu, and X. Lin, "Fine-grained data sharing in cloud computing for mobile devices," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, Apr./May 2015, pp. 2677–2685.
[29] J. Li, X. Huang, J. Li, X. Chen, and Y. Xiang, "Securely outsourcing attribute-based encryption with checkability," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 8, pp. 2201–2210, Aug. 2014.
[30] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology—EUROCRYPT*. Berlin, Germany: Springer, 2005, pp. 457–473.
[31] M. Chase, "Multi-authority attribute based encryption," in *Proc. 4th Theory Cryptogr. Conf. (TCC)*, 2007, pp. 515–534.
[32] H. Lin, Z. Cao, X. Liang, and J. Shao, "Secure threshold multi authority attribute based encryption without a central authority," *Inf. Sci.*, vol. 180, no. 13, pp. 2618–2632, Jul. 2010.
[33] S. Müller, S. Katzenbeisser, and C. Eckert, "Distributed attribute-based encryption," in *Information Security and Cryptology—ICISC*. Berlin, Germany: Springer, 2009, pp. 20–36.
[34] S. Ruj, A. Nayak, and I. Stojmenovic, "DACC: Distributed access control in clouds," in *Proc. 10th Int. Conf. Trust, Secur. Privacy Comput. Commun. (TrustCom)*, Nov. 2011, pp. 91–98.
[35] K. Yang and X. Jia, "Expressive, efficient, and revocable data access control for multi-authority cloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 7, pp. 1735–1744, Jul. 2013.
[36] J. Han, W. Susilo, Y. Mu, J. Zhou, and M. H. A. Au, "Improving privacy and security in decentralized ciphertext-policy attribute-based encryption," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 3, pp. 665–678, Mar. 2015.
[37] T. Jung, X. Li, Z. Wan, and M. Wan, "Control cloud data access privilege and anonymity with fully anonymous attribute-based encryption," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 1, pp. 190–199, Jan. 2014.
[38] K. Yang, X. Jia, and K. Ren, "Secure and verifiable policy update outsourcing for big data access control in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 12, pp. 3461–3470, Dec. 2015.
[39] J. Li, K. Ren, B. Zhu, and Z. Wan, "Privacy-aware attribute-based encryption with user accountability," in *Information Security*. Berlin, Germany: Springer, 2009, pp. 347–362.
[40] Z. Liu and Z. Cao, "On efficiently transferring the linear secret-sharing scheme matrix in ciphertext-policy attribute-based encryption," IACR Cryptol. ePrint Arch., Tech. Rep. 2010/374, 2010, [Online]. Available: http://eprint.iacr.org/
[41] J. Li, K. Ren, and K. Kim, "A^{2}BE: Accountable attribute-based encryption for abuse free access control," IACR Cryptol. ePrint Arch., Tech. Rep. 2009/118, 2009, [Online]. Available: http://eprint.iacr.org/
[42] J. Li, K. Ren, and K. Kim, "A^{2}BE: Accountable attribute-based encryption for abuse free access control," *IACR Cryptol. ePrint Arch.*, vol. 2009, p. 118, 2009.

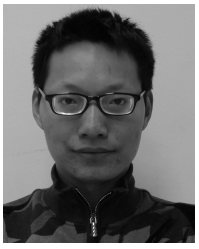
Kaiping Xue (M'09–SM'15) received the B.S. degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2003, and the Ph.D. degree from the Department of Electronic Engineering and Information Science (EEIS), USTC, in 2007. He is currently an Associate Professor with the Department of Information Security and the Department of EEIS, USTC. His research interests include next-generation Internet, distributed networks, and network security.



Yingjie Xue received the B.S. degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2015. She is currently pursuing the degree in communication and information system with the Department of Electronic Engineering and Information Science, USTC. Her research interests include network security and cryptography.



Jianan Hong received the B.S. degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2012. He is currently pursuing the Ph.D. degree in information security with the Department of Electronic Engineering and Information Science, USTC. His research interests include secure cloud computing and mobile network security.



Wei Li received the B.S. degree from the Department of Information Security, University of Science and Technology of China (USTC), in 2014. He is currently pursuing the degree in communication and information system with the Department of Electronic Engineering and Information Science, USTC. His research interests include network security protocol design and analysis.

Hao Yue (M'13) received the B.Eng. degree in telecommunication engineering from Xidian University, Xi'an, China, in 2009, and the Ph.D. degree in electrical and computer engineering from the University of Florida, Gainesville, FL, USA, in 2015. He is currently an Assistant Processor with the Department of Computer Science, San Francisco State University, San Francisco, CA, USA. His research interests include cyber-physical systems, cybersecurity, wireless networking, and mobile computing.

David S. L. Wei (SM'07) received the Ph.D. degree in computer and information science from the University of Pennsylvania in 1991. From 1993 to 1997, he was with the Faculty of Computer Science and Engineering, University of Aizu, Japan, as an Associate Professor and then a Professor. He is currently a Professor with the Computer and Information Science Department at Fordham University. He has authored or co-authored over 100 technical papers in the areas of distributed and parallel processing, wireless networks and mobile computing, optical networks, peer-to-peer communications, cognitive radio networks, big data, and cloud computing in various archival journals and conference proceedings. His research interests include cloud computing, big data, IoT, and cognitive radio networks. He served on the program committee and was a Session Chair of several reputed international conferences. He was a lead Guest Editor of IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS for the Special Issue on Mobile Computing and Networking, a lead Guest Editor of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS for the Special Issue on Networking Challenges in Cloud Computing Systems and Applications, a Guest Editor of IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS for the Special Issue on Peer-to-Peer Communications and Applications, and a lead Guest Editor of IEEE TRANSACTIONS ON CLOUD COMPUTING for the Special Issue on Cloud Security. He was the Chair of the Intelligent Transportation Forum of Globecom 2010, the General Chair of the Intelligent Transportation Workshop of ICC 2011, and the Chair of the Cloud Security Forum and Intelligent Transportation Forum of Globecom 2011. He is currently an Associate Editor of IEEE TRANSACTIONS ON CLOUD COMPUTING, an Associate Editor of the *Journal of Circuits, Systems and Computers*, and also a Guest Editor of IEEE TRANSACTIONS ON BIG DATA for the Special Issue on Trustworthiness in Big Data and Cloud Computing Systems.

Peilin Hong received the B.S. and M.S. degrees from the Department of Electronic Engineering and Information Science (EEIS), University of Science and Technology of China (USTC), in 1983 and 1986, respectively. She is currently a Professor and an Advisor for Ph.D. candidates in the Department of EEIS, USTC. She has authored two books and over 150 academic papers in several journals and conference proceedings. Her research interests include next-generation Internet, policy control, IP QoS, and information security.