SEM-ACSIT: Secure and Efficient Multi-Authority Access Control for IoT Cloud Storage

Shuming Xiong, Qiang Ni, *Senior Member, IEEE,* Liangmin Wang, *Member, IEEE,* and Qian Wang

***Abstract*—Data access control in a cloud storage system is regarded as a promising technique for enhanced efficiency and security utilizing ciphertext-policy attribute-based encryption (CP-ABE) approach. However, due to large number of data users as well as limited resources and heterogeneity of data devices in Internet of Things (IoT), existing access control schemes for the cloud storage are not effectively applicable to IoT applications. In this paper, we construct a new CP-ABE based storage model for data storing and secure access in a cloud for IoT applications. Our new framework introduces an attribute authority management (AAM) module in the cloud storage system functioned as an agent that provides a user-friendly access control and highly reduces the storage overhead of public keys. Then, we propose a novel secure and efficient multi-authority access control scheme of the cloud storage system for IoT, namely SEM-ACSIT, which obtains both backward security and forward security when an attribute of a user is revoked. By exploiting encryption outsourcing, simplified key structuring and the AAM module, computational overhead of a user is immensely decreased. Moreover, a user access control list (UACL) in the cloud server is constructed newly to support authorization access for a specific user. The analysis and simulation results demonstrate that our SEM-ACSIT scheme achieves powerful security with less computational overhead and lower storage cost than existing schemes.**

***Index Terms*—Cloud storage, access control, multi-authority, attribute-based encryption, Internet of Things (IoT).**

I. Introduction

The Internet of Things (IoT) technology has achieved large development in many aspects such as signal sensing, wireless communication, data transferring and processing, and is applied widely in various fields [1–3], especially in the industry environment [4]. For example, smart city is a highly important field to exploit the IoT technology because low carbon consumption and high quality of life are more emphasized than ever before for urban inhabitants in recent years. Furthermore, IoT enables to exploit public resources more conveniently and to reduce the cost of public administration through building smart city [5]. More and more different types of environment-conscious devices, for example, smart sensors, RFID readers, cameras, mobile phones, intelligent vehicles, roadside terminals, etc, are distributed in many territories and carried with persons respectively. On the other hand, from the point of view of trades there are many instrumental devices to implement remote metering, such as in water utilities, power service and natural gas provision. These devices which are affiliated with smart city and connected by IoT can persistently collect a large amount of data and impose a new challenge to data processing in IoT. Therefore, it is necessary to supply enough storage space to store the big-sized data [6]. In addition, there can be some sensitive data that should be kept privacy [7] in IoT applications. As a result, security of these sensitive data sourced from some smart city applications based on IoT needs to be guaranteed.

The cloud computing is an increasingly paramount computing paradigm that envisions a promising future, in which a user can conveniently enjoy computability anywhere and anytime by the Internet. The cloud storage service is also exploited by increasingly growing number of applications, and it provides consumers with data storage and access control services [8], which can effectively solve the problems sourced from big data mentioned earlier. For instance, in a prospective smart city transportation system connected by IoT and assisted by cloud storage respectively, the transportation data are collected from the vehicles and roadside traffic monitoring terminals and automatically stored in a cloud storing system. Then, a traffic policeman, a vehicle administrative officer and even some vehicle insurance companies or automobile manufacturers can inquire the driving information of a person anywhere by using mobile devices or some other network terminals. It offers big convenience for a traffic policeman and an automobile service provider to be able to remotely monitor and even control traffic flow by IoT. However, a driver also concerns about his/her privacy seriously. Thus, some sensitive information relating to individuals should be stored in confidentiality. If a curious internal staff or a deliberate external attacker can access individual information of a driver without authorized permission, it will pose a potential threat to security of the data possessed by an owner. But the owner may be willing to share his/her data with some authorized people. Furthermore, he/she may want to share different data with different people. Then access control to big-sized data like in smart city applications based on IoT emerges as a challenge in order to cater for heterogeneous requirements for security and convenience.

Ciphertext-policy attribute-based encryption (CP-ABE) [9] is deemed as a promising technology for access control in a cloud storage system for IoT applications because of flexibility of access control policy. A data user (DU) can get shared

This work is supported by National Natural Science Foundation of China (U1736216), Jiangsu Government Scholarship for Overseas Study, Humanities and Social Science Research Youth Fund Project of Ministry of Education of China (17YJC910008) and project 2016B050502001.

S. Xiong, L. Wang, and Q. Wang are with the School of Computer Science and Communication Engineering, Jiangsu University, and also with Jiangsu Key Laboratory of Security Tech. for Industrial Cyberspace, Zhenjiang, China, 212013 (e-mail: xsm@ujs.edu.cn; wanglm@ujs.edu.cn; 251256145@qq.com).

Q. Ni is with the School of Computing and Communications, InfoLab21, Lancaster University, U.K., LA1 4WA (e-mail: q.ni@lancaster.ac.uk).

access to data provided by a data owner (DO) if there exist some attributes of a DU to satisfy the access policy issued by the DO. Usually there is an authority responsible for attribute management and key distribution in a variety of attribute-based encryption schemes. Extensive researches [10–14] have been done for a system with single authority based on CP-ABE approach. However, a data user may hold several attributes allocated from multiple authorities which are managed by different sectors and a data owner may share data with the users administrated by some different authorities in a cloud storage system. For example, in the smart transportation system of a city, data regarding to personal driving may be shared only with a user who holds two attributes of both "reparation" and "insurance" issued by a garage and an automobile insurance company respectively, or shared with a user possessing two attributes of both "traffic-guidance" and "accident-treatment" only issued by a traffic policeman team. Some multi-authority CP-ABE approaches, such as [15–17] are proposed to conveniently implement data access control between a data owner and some data users.

However, these approaches are designed to adapt to a common situation and cannot be directly applied to a cloud storage system. Therefore, in order to apply the multi-authority CP-ABE approach to implement data access control in a cloud storage system, two schemes [18, 19] are further proposed. They implement distributed key management relating to attributes of a user and realize data sharing among a data owner and some data users through exploiting an attribute access control policy. However, they are devised mainly to adapt to a universal cloud storage system, and some features of IoT such as device heterogeneity, resource limitation and large number of users are not considered. In addition, many resource-limited users in IoT face overlarge cost when exploiting the existing data access control schemes for the cloud storage system, which highly deteriorates performance of an individual data device and an entire IoT system. Meanwhile, the data access control scheme in [18] does not possess forward security when revoking a user, and that in DAC-MACS [19] does not have backward security when an attribute authority revokes an attribute of a user [20]. Our further analysis shows that the scheme in [19] lacks forward security when revoking an attribute of a user, i.e. the revoked user can still decrypt the previously-made ciphertext that may be decrypted only when a user possesses the revoked attribute, which we will deal with. Furthermore, big overheads in storage and computation of [19] may incur a challenge to a broad range of resource-limited data devices in many IoT applications.

The big-volumed data generated in IoT pose another challenge to devise an efficient and convenient data access control system. And the dynamic change of a data user role will make it more serious. Therefore, in this work we are inspired by the secure cloud storage system presented in [19] to devise a new secure and efficient multi-authority access control scheme of a cloud storage system for IoT (SEM-ACSIT) based on CP-ABE. The main contributions of this paper are summarized as follows:

- Aiming to the heterogeneity features of IoT applications, we propose a new user-friendly data access control framework for a cloud storage system, which provides a unified user access interface by using an attribute authority management (AAM) module.
- We construct a novel secure and efficient data access scheme based on CP-ABE in our proposed framework to realize data sharing for the cloud storage system in IoT with multiple attribute authorities, which can ensure both backward and forward security when revoking an attribute of a data use.
- Our SEM-ACSIT scheme not only outsources decryption to the cloud server, but also dumps public key storage of both attribute authorities and users to an AAM module, which can reduce computation cost and user storage in a system level.
- For a specific user, we construct a user access control list (UACL) in the cloud server to enable the specified user to get direct and effective data access by adding an authorization record in the UACL, enhancing flexibility of shared data access.

This paper is organized as follows. Section II gives a detailed review of the related work. Section III briefly introduces some preliminary knowledge related to our scheme. Design of the proposed scheme is presented in detail in Section IV. Section V provides performance analysis and experiment results. Section VI concludes this paper.

II. Related work

Several kinds of access control schemes have been proposed for different user requirements, among which a model based on attribute-based encryption (ABE) [21] attracts big interest for securing cloud storage. ABE mechanisms can be divided into two main categories, namely, key-policy attribute-based encryption (KP-ABE) [22] and CP-ABE [9]. In a cloud storage system, many factors such as computation overhead, communication cost, management of attribute keys and security requirements impose large challenge on design of access control scheme and the model based on CP-ABE is commonly exploited.

Initially, the research work in ABE application is mainly related to a model of single authority. Hur et al. [10] proposed a scheme which exploits CP-ABE and selective group key distribution in each attribute group. The scheme requires a trusted attribute authority to administrate all of the attributes in the system and distribute secret keys to users. As a result, the authority becomes a possible vulnerability and further a performance bottleneck of the data access control system. In addition, their scheme incurs heavy computing overhead. Green et al. [11] proposed two kinds of ABE schemes that outsource the decryption to a server in order to highly eliminate the overhead of users. In their schemes, the traditional secret key is divided into a user secret key and a transformation key which is sent to the server and exploited to generate a constant-size temporary El Gamal-style ciphertext with the server unable to learn content of the ciphertext.

Recently, there are some meaningful research efforts for multi-authority based on CP-ABE. Ruj et al. [18] proposed a distributed access control scheme (DACC) for a multi-authority system that contains several key distribution centers

to distribute attribute keys to data users and data owners. The scheme avoids storing multiple encrypted copies of the same data and also constructs an attribute revocation mechanism to revoke a user. However, their scheme does not possess forward security when revoking a user. Due to large calculation cost of attribute revocation Liu et al. [23] proposed a multi-authority attribute based encryption scheme with attribute revocation. The scheme adds the user revocation list into ciphertext to implement user revocation. Through decryption outsourcing and proxy re-encryption technology it satisfies security requirements, meanwhile reducing the computation overhead. From point of view of privacy preservation Zhou et al. [24] proposed a TR-MABE scheme for e-healthcare application, namely a white-box traceable and revocable multi-authority attribute-based encryption. The scheme can efficiently attain multi-level privacy preservation without extra signatures. Nomura et al. [25] proposed a multi-authority attribute-based encryption scheme with attribute revocation for cloud storage. It can revoke an attribute without updating a user's secret key.

Yang et al. [19] proposed an effective data access control for multi-authority cloud storage systems (DAC-MACS) based on CP-ABE and outsourced primary computation of the decryption to a cloud server by generating a decryption token. Moreover, they designed an efficient attribute revocation method that incurs less cost of communication and revocation computation. But, computing overhead and cost of attribute revocation in DAC-MACS scheme are still the burden for a resource-limited user in IoT. Wu et al. [26] analyzed the two attacks in multi-authority cloud storage system with DAC-MACS, including eavesdropping users secret key updates and intercepting ciphertext update key. And, they proposed a new extensive DAC-MACS scheme (NEDAC-MACS) for multi-authority cloud storage system against the two attacks. The scheme can guarantee more security and its performance is similar to that of DAC-MACS. Considering low efficiency in multi-authority access control system Xue et al. [27] proposed a robust and auditable access control scheme for a public cloud storage with multiple attribute authorities. It introduces a central authority to generate secret keys for a legal user. In addition, the scheme can detect which attribute authority has maliciously carried out the legitimacy authentication through an auditing mechanism. It makes big performance advantages when key generating on the basis of guaranteeing the security requirements. Aiming to access control for encrypted cloud storage Xue et al. [28] proposed a scheme based on CP-ABE to secure cloud data storage against economic denial of sustainability (EDoS) attacks. At the same time, it provides resource consumption accounting to control service cost through bloom filter and probabilistic check. The scheme is secure and efficient when applied in a real-world environment. Due to not complete trustworthiness for a cloud server Wei et al. [29] proposed a secure and cost-effective access control scheme for multi-authority cloud storage system based on CP-ABE. The cloud storage access model supports scalable user revocation and dynamic ciphertext updating. It is demonstrated that the scheme is secure in the random oracle model and has good practicality. Yahiatene et al. [30] proposed a new framework to control access to the private data on online social networks. The framework is based on cloud storage and exploits a distributed multi-authority ABE model. Owing to an integrated access control mechanism the scheme can ensure data security and enable data access only to an authorized user. In order to support users to gain access permission through collaboration Xue et al. [31] proposed a collaborative access control scheme based on attribute-based encryption model. In the scheme a data owner can designate some selected users to construct a group to collaborate for accessing shared data. Further, it considers to resist collusion attack when an adversary combines some attributes from different users. The scheme can guarantee data security and is performance-efficient according to storage and computation overheads.

III. SCHEME PRELIMINARY

A. Bilinear Map

Let G and G_T be two multiplicative cyclic groups with a prime order p. Let g be a generator of G. The map $e : G \times G \rightarrow G_T$ is a bilinear map if it has the following properties:

1) Bilinearity: For all $u \in G$, $v \in G$ and $a, b \in \mathbb{Z}_q$, $e(u^a, v^b) = e(u, v)^{ab}$.
2) Non-degeneracy: $e(g, g) \neq 1$.
3) Computability: For all $u \in G$, $v \in G$, $e(u, v)$ is computable efficiently.

B. Access Structure

Let $P = \{P_1, P_2, \ldots, P_n\}$ be a set of parties. An access structure [13] is a collection $\mathbb{A}$ of nonempty subsets of P, i.e., $\mathbb{A} \subseteq 2^{\{P_1, P_2, \ldots, P_n\}} \setminus \{\emptyset\}$. If a collection $\mathbb{A}$ is monotone, then $\forall X, Y$: if $X \in \mathbb{A}$ and $X \subseteq Y$, then $Y \in \mathbb{A}$. A monotone access structure means that $\mathbb{A}$ is monotone. In our research, a party associates with an attribute of a user, and we assume that the access structure is monotone.

An example of attribute access structure in the form of a binary tree is shown in Fig. 1. If an owner defines an access policy and a corresponding boolean expression is $A \wedge B \vee (C \wedge (D \vee E))$, then an access structure of the policy is $R = \{\{A, B\}, \{C, D\}, \{C, E\}\}$. Therefore, both user1 and user2 with the sets of $\{A, B\}$ and $\{C, D\}$ respectively satisfy requirements of the access structure, but user3 with an attribute set $\{C\}$ does not satisfy the access structure.

IV. CONSTRUCTION OF SEM-ACSIT SCHEME

In this section, we describe our proposed SEM-ACSIT scheme in detail.

A. System Model and Security Assumptions

There are six kinds of entities in our cloud storage system for IoT applications with multiple authorities. These entities include a global certificate authority (CA), some attribute authorities (AAs), an AAM module, a cloud server (CServer), many DOs and DUs, as shown in Fig. 2.

In multi-authority storage system for IoT applications, being a globally trusted certificate authority the CA is originally responsible for establishing the system and accepting registration request from all users, owners (as some special users) and

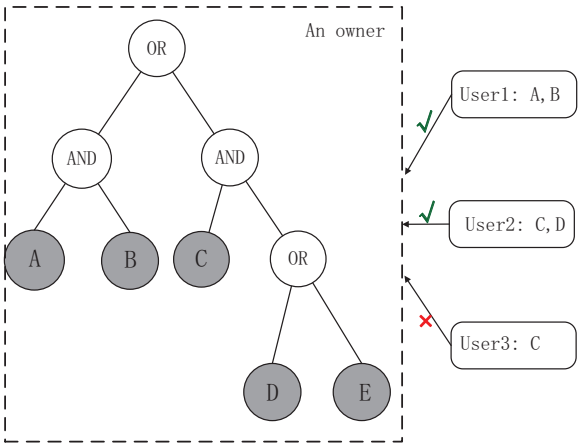

Fig. 1. Access structure of attributes by an owner.

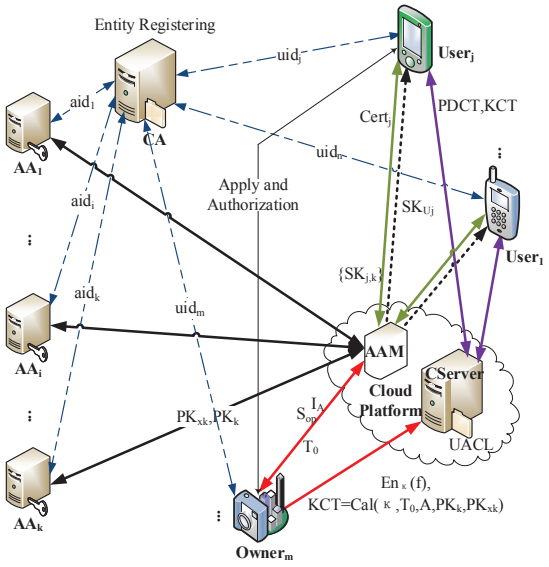

Fig. 2. Access control model of a cloud storage system in IoT applications with multiple authorities.

AAs in the system. The AAM supported by a cloud platform is functioned as a bridge between some AAs and many users (including DOs and DUs) in order to improve data access efficiency of users, especially in resource-limited scenarios of IoT. Each of AAs is an independent attribute authority which is mainly in charge of dynamically distributing and revoking attributes of users according to their roles in its domain. Furthermore, every AA is also responsible for generating and distributing some different types of keys associated with a users attributes. The CServer provides data storage and access control services to users and manages the UACL for all shared files, which contributes to support user-specific access control. A DO encrypts shared data according to an access policy and stores the corresponding ciphertext into the CServer. A DU will be able to decrypt the encrypted data if his/her attributes are matched to an access policy of the data owner.

We make following security assumptions for the multi-authority storage system of IoT applications:

1) The CA is trusted in the system. As a system constructor, it will accept the registration of AAs and data users. But, the CA is not responsible for managing attributes of the users and corresponding attribute key generation.
2) Each of AAs is also trusted, but it can be corrupted by an adversary.
3) Both CServer and AAM in the cloud platform are curious but honest. They are curious about the content of encrypted data or a received message, but they will correctly execute the tasks assigned by each AA, a user and an owner.
4) The users are dishonest and may collude with other users or AAs to obtain unauthorized access to shared data.
5) A decisional q-parallel bilinear Diffie-Hellman exponent (q-parallel BDHE) assumption is made. We review the decisional q-parallel BDHE problem followed as in [13]. Let $a, s, b_1, \ldots, b_q \in \mathbb{Z}_p$ be chosen at random and g be a generator of a group G. An adversary must distinguish $e(g,g)^{a^{q+1}s} \in G_T$ from a random element R in G_T if it is given

$$\vec{y} = \Big(g, g^a, g^{a^2}, \ldots, g^{a^q}, , g^{a^{q+2}}, \ldots, g^{a^{2q}}, g^s$$
$$\forall_{1\le j\le q} : g^{a/b_j}, \ldots, , g^{a^{q+2}/b_j}, \ldots, g^{a^{2q}/b_j}, g^{sb_j}$$
$$\forall_{1\le j,k\le q, k\ne q} : g^{asb_k/b_j}, \ldots, g^{a^q sb_k/b_j}\Big).$$

An algorithm $\mathscr{B}$ that outputs $z \in \{0,1\}$ has advantage ϵ in solving q-parallel BDHE in G if

$$\Big|Pr[\mathscr{B}(\vec{y}, T = e(g,g)^{a^{q+1}s}) = 0] - Pr[\mathscr{B}(\vec{y}, T = R) = 0]\Big| \ge \epsilon.$$

We define that the decisional q-parallel BDHE assumption holds if no polynomial time algorithm has a non-negligible advantage in solving the q-parallel BDHE problem.

B. Overview of SEM-ACSIT

There are a few multi-authority data access control schemes proposed for a cloud storage system, such as [18, 19], which are able to realize security of data sharing and attribute revocation. However, these schemes cannot be directly applied to the secure storage system for IoT applications because of different security requirements. And high overheads sourced from both system-level storage and user-level computation in those schemes are additionally negative factors to apply them in IoT applications. Thus, to design a secure and efficient data access control scheme for IoT application storage systems is still a big challenge.

IoT is widely enabled by the latest progress in smart sensors, WSNs, RFID, cloud platform and some other communication technologies. For many IoT applications, there exists a large quantity of data sharing between data providers and data consumers with limited resources. Number of these data devices is enormous and their performance is heterogeneous. Therefore, one challenge in designing a data access control scheme for the IoT storage system is how to realize secure data

access and reduce overhead of the system simultaneously. In SEM-ACSIT, we adopt the key framework proposed by [19], therefore every attribute has an attribute public key and an attribute private key. Moreover, each of AAs possesses both a public key and a secret key relating to unique authority identification and every user has a set of keys relating to the attributes distributed by some AAs, which provides a base to resist collusion attack between an adversary and some other users or attribute authorities. However, different from DAC-MACS [19], SEM-ACSIT simplifies key organization and optimizes key storage in order to improve computing efficiency and lessen storage redundancy relating to attribute keys. To exploit powerful performance of a cloud computing platform, SEM-ACSIT introduces an attribute authority management AAM into the cloud platform that can effectively manage resources in the storage system. The AAM stores public keys of all attributes, public keys of all authorities and hash values of all attributes, which reduces key storage redundancy of data owners in [19]. Furthermore, it provides a friendly-user data access control interface by which a user is able to switch from a one-to-many communication mode to a one-to-one mode and highly decrease total number of communications to all AAs. Finally, the AAM also undertakes some intensive encryption computation to lower computational overheads of a data owner. The SEM-ACSIT scheme has good applicability and practicality due to its definite framework and less user computation overheads, and can be applied in such fields as smart traffic, community management and so on.

In some IoT applications, roles of many network data devices are frequently changed, for example in such a clustering-based scenario as LBC-DDU [32]. Role change of sensor nodes inevitably induces many attributes to change. Thus, another challenge is how to efficiently treat the frequent attribute revocation of many data users in the access control system for IoT applications. To solve this attribute revocation problem, a corresponding AA_k that manages the revoked attribute problem updates the two keys relating to it. Then, each of non-revoked users updates his/her secret key set, public key of the revoked attribute, and many pieces of cipertext relating to it are renewed locally by all owners and the CServer respectively. However, frequent attribute revocation will lead to a large number of communications for all owners, which augments overheads of a system. In addition, due to ciphertext updating by the CServer the existing scheme cannot resist collusion attack between a revoked user and the cloud server when the revoked user illegally obtains the ciphertext update key [20]. Therefore, in SEM-ACSIT we apply an agent AAM of authorities located in the cloud platform to store all attribute public keys. It is enough to locally update an attribute public key in the AAM module when an attribute is revoked from a user, which is able to highly reduce communication overheads for resource-limited data owners even in the case of frequent attribute revocation in IoT applications. In order to solve security problem when revoking an attribute from a user, our proposed scheme processes ciphertext update through the AA_k managing the revoked attribute as opposed to by the CServer in [19].

The SEM-ACSIT scheme also provides a specified user access control service by exploiting a UACL structure to improve flexibility of data access authorization in IoT applications. A DU, while not satisfying the access policy, can still be able to access the shared file if the DU obtains an access authorization from a DO directly.

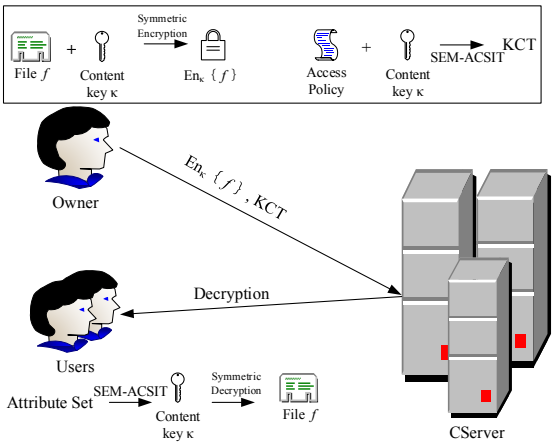

Fig. 3. Data sharing process between the owner and each user.

We briefly describe a process of data sharing as shown in Fig. 3. Initially, a DO wants to upload his/her data file f to the CServer, and he/she will encrypt the file using a symmetric algorithm with a content key κ to obtain the ciphertext $En_\kappa(f)$. Then, the DO defines an access policy, such as "(reparation AND insurance) OR traffic-guidance" in a smart city transportation system built through the IoT technology, and encrypts the content key κ using SEM-ACSIT scheme with the defined access policy to produce a piece of key ciphertext (KCT). To encrypt the content key rather than to directly encrypt the content is able to highly reduce computational overhead and communication cost for a cloud storage system. If some users want to access the file f, they must try to decrypt KCT through our scheme and obtain κ first. Only if attributes of a DU satisfy the access policy embedded into a piece of ciphertext, could the user decrypt the key ciphertext. Then, the user can access the file successfully because he/she can decrypt to obtain f using the symmetric key κ.

A structure of the file storage in a cloud storage system is shown in Fig. 4, where ID is a unique identity for file access control and τ is the time of uploading a file. In the structure, there are such two kinds of ciphertext as key ciphertext KCT and data ciphertext $En_\kappa\{data\}$, which is corresponding to the symmetric key as well as data themselves respectively. $UACL$ is created by the CServer for each shared file when an owner uploads the encrypted file to the CServer. It consists of a triple record like $\langle RNum, UValue, ATime\rangle$. $RNum$ is a value selected randomly and unique to each of records. $UValue$ is a value which a data owner stores into the record. If a data user obtains an access authentication from a DO directly, the user can get the corresponding $UValue$ by which he/she can decrypt the ciphertext readily. $ATime$ is the current time of adding the record.

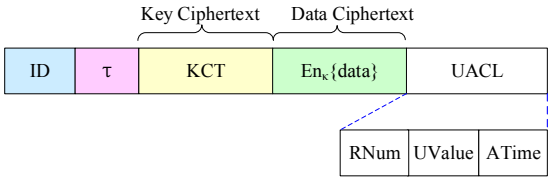

Fig. 4. Structure of encrypted file stored in the CServer.

C. Construction of SEM-ACSIT

We let S_U and S_A be a set of users (an owner representing a special user in the period of registering) and a set of attribute authorities in a cloud storage system for IoT respectively. Let G and G_T be two multiplicative groups with the same prime order p and $e : G \times G \longrightarrow G_T$ be a bilinear map. Let g be a generator of G and define a hash function $F : \{0,1\}^* \longrightarrow G$ which maps an attribute into the group G so that security of the system is in the random oracle. The proposed SEM-ACSIT scheme consists of eight phases, including system initialization, AA joining, user joining, user secret key generation, data encryption, ciphertext decryption, attribute revocation, and authenticated access for a specified user as well.

1) ***Initialization of a Cloud Storage System****:* A cloud storage system for IoT applications is established initially by the CA. It executes an algorithm $CASetup(\lambda) \longrightarrow (SP, SK_{CA}, VK_{CA})$ in this period and the input parameter λ is a security parameter. Specifically, the CA first randomly chooses a number $a \in \mathbb{Z}_p$ as a parameter of the cloud storage system and generates the system parameter $SP = \{g, g^a, G, G_T, F\}$. Then, the CA generates a pair of secret key and verification key $\{SK_{CA}, VK_{CA}\}$, such as by exploiting RSA algorithm in order to authenticate validity of a user when obtaining secret keys from an attribute authority AA_k. Those prospective AAs and users must register to the CA and obtain some security parameters just when they join the cloud storage system for IoT applications.

2) ***AA Joining****:* Every AA should register itself to the CA when joining the cloud storage system for IoT applications. If an AA is a legal attribute authority, the CA first allocates a global authority identity aid, namely k, to the AA. Then, the CA sends the system parameter SP and its verified key VK_{CA} to the AA. After receiving these security parameters, each $AA_k (k \in S_A)$ executes an algorithm $AASetup(SP, k) \longrightarrow (PK_k, SK_k, \{PK_{x_k}, VK_{x_k}\})$. It chooses $\alpha_k, \beta_k \in \mathbb{Z}_p$ randomly as the AA_k authority secret key $SK_k = \{\alpha_k, \beta_k\}$. In addition, AA_k randomly chooses $v_{x_k} \in \mathbb{Z}_p$ for each attribute $x_k \in S_{A_k}$ managed by itself as a private version key VK_{x_k} and computes $PK_{x_k} = (g^{v_{x_k}})^{1/\beta_k}$ as an attribute public key, where S_{A_k} is a set of attributes managed by AA_k. Furthermore, AA_k calculates the authority public key as shown in formula (19):

$$PK_k = \{e(g,g)^{1/\alpha_k}, g^{\beta_k}\} \tag{1}$$

Then, a set of attribute public keys $\{PK_{x_k}\}$ and a set of hash values $\{F(x_k)\}$ corresponding to every attribute x_k in the AA_k should be stored together with the authority public key PK_k into the two index tables of AAM as shown in Table I and II.

TABLE I
ATTRIBUTES AND ITS PUBLIC KEYS AS WELL AS HASH VALUES

Attribute	Attribute Parameter	Authority
x_{11}	$(PK_{x_{11}}, F(x_{11}))$	AA_1
x_{12}	$(PK_{x_{12}}, F(x_{12}))$	AA_1
x_{13}	$(PK_{x_{13}}, F(x_{13}))$	AA_1
x_{21}	$(PK_{x_{21}}, F(x_{21}))$	AA_2
...	...	...

TABLE II
PUBLIC KEYS OF ATTRIBUTE AUTHORITIES

Authority	Authority Parameter
AA_1	PK_1
AA_2	PK_2
...	...

3) ***User Joining****:* Each of users (including owners) should first register themselves to the CA when they join a cloud storage system for IoT applications. The CA will assign a globally unique user identity uid to a user if the user is legal in the system. For a user with uid, the CA further generates a global public key $GPK_{uid} = g^{u_{uid}}$ and a global secret key $GSK_{uid} = z_{uid}$ by choosing two random numbers $u_{uid}, z_{uid} \in \mathbb{Z}_p$ respectively. In addition, the CA produces a certificate $Cert(uid)$ containing an entry of $Sign_{SK_{CA}}(uid, u_{uid}, g^{z_{uid}})$. Finally, the CA gives GPK_{uid}, GSK_{uid} and $Cert(uid)$ to the user uid.

4) ***User Secret Key Generation by AAs****:* In order to obtain secret keys from all AAs, a user $U_j (j \in S_U)$ first sends $Cert(j)$ to the AAM, and then the AAM transparently sends it to the all AAs, by which the cloud storage system can effectively unify and simplify a user access interface to attribute authorities. Every $AA_k (k \in S_A)$ can verify $Cert(j)$ by exploiting verification key VK_{CA} of the CA to obtain $\{j, u_j, g^{z_j}\}$. If the user U_j is illegal, generation request of secret key will be refused. Otherwise, the AA_k allocates a set of attributes $SA_{j,k}$ to the user U_j according to its role in the access control system. Then, the attribute authority AA_k runs an algorithm $SKeyGen(SA_{j,k}, SK_k, \{PK_{x_k}\}, SP, Cert(j)) \longrightarrow SK_{j,k}$ to generate the secret key $SK_{j,k}$ of the user U_j as formula (2):

$$\begin{aligned} SK_{j,k} &= \{K_{j,k} = g^{a \cdot u_j} g^{z_j (1/\alpha_k - 1)}, \\ &\forall x_k \in SA_{j,k} : K_{j,x_k} = g^{v_{x_k}((u_j+1)/\beta_k^2 + 1/\beta_k)} \cdot F(x_k)^{u_j/\beta_k}\} \end{aligned} \tag{2}$$

Here, $j \in S_U$ and $k \in S_A$. Finally, the AA_k returns the secret key $SK_{j,k}$ to the AAM, and then the AAM will further send all secret keys obtained from all AAs to the user U_j collectively.

5) ***Data Encryption by an Owner****:* A data owner DO first encrypts the file f with a content key κ using a symmetric encryption method to generate data ciphertext $En_\kappa(f)$. Let $S_{OP} = \{x_{oi}\}_{oi=1 \ to \ l}$ be a set of attributes involved in the access structure $\mathbb{A}$ like in Fig. 1, where l is the number of attributes in an access control policy. Furthermore, let I_A

denote a set of the AAs who manage an attribute set S_{OP}. Then, the owner sends I_A and S_{OP} sets to the AAM. In order to reduce computational overhead of the owner, the AAM executes an algorithm $MCTGen(\{PK_k\}_{k\in I_A})\longrightarrow T_0$ to generate the provisional ciphertext as formula (3):

$$T_0 = \prod_{k\in I_A} e(g,g)^{1/\alpha_k - 1} \tag{3}$$

When computing T_0, the AAM gets PK_k by enquiring Table II and sends it back to the data owner together with $\{PK_k\}$, $\{PK_{x_k}\}$ and $\{F(x_k)\}$ as well. Then, the data owner takes as inputs T_0, SP, κ, $\{PK_k\}_{k\in I_A}$, $\{PK_{x_k}\}_{x_k\in S_{A_k}}$ and the access structure $\mathbb{A} = (\mathbf{M}, \rho)$ to execute Encrypt algorithm to generate key ciphertext KCT, where S_{A_k} denotes a set of attributes of the attribute authority AA_k relating to the access structure $\mathbb{A}$. Each of the attributes involved in $\mathbb{A}$ is managed by only one AA, and $\mathbf{M}$ is an $l \times n$ access matrix obtained from an access tree associated with the access policy, where l is the number of attributes as mentioned earlier and n is related to specific logic expression of the access policy. The function ρ maps a row number of matrix $\mathbf{M}$ to an attribute x_{oi} of the access policy. In addition, the data owner randomly chooses $s \in \mathbb{Z}_p$ and a vector $\vec{v} = (s, y_2, ..., y_n) \in \mathbb{Z}_p^n$ with s as its first entry, where random values $y_2, ..., y_n$ are employed to share the secret s. For each row of matrix $\mathbf{M}$, it also randomly chooses $r_1, r_2, ..., r_l \in \mathbb{Z}_p$ to compute the key ciphertext. For $i = 1\ to\ l$, the data owner first computes

$$\lambda_i = \vec{v} \cdot \mathbf{M}_i \tag{4}$$

, where $\mathbf{M}_i$ is a row vector corresponding to the ith row of the access matrix $\mathbf{M}$. Then, the DO computes the key ciphertext KCT according to formula (5) and uploads it as well as $En_\kappa(f)$ to the CServer.

$$\begin{aligned} &KCT = \langle \mathbf{M}, \rho, KC = \kappa \cdot (T_0)^s, C_0 = g^s, \\ &\forall i = 1\ to\ l: \\ &C_{i,1} = g^{a\lambda_i}(g^{v_{\rho(i)}/\beta_k} F(\rho(i)))^{r_i}, \\ &C_{i,2} = g^{-\beta_k r_i}, C_{i,3} = g^{-r_i}, \\ &\rho(i) \in S_{A_k}, k \in I_A\rangle \end{aligned} \tag{5}$$

6) ***Ciphertext Decryption by the Cloud and User****:* In the cloud storage system for IoT applications, any legal user can obtain any interested ciphertext from the CServer. However, only if attributes of a user meet the access policy, the user will be able to decrypt to get the content key and exploit it to further obtain the file. In order to reduce the computational overhead of a user in IoT, the ciphertext decryption phase consists of two steps. Initially, against the highly intensive computation cost on a user the CServer decrypts the key ciphertext to obtain a partially decrypted ciphertext $PDCT$. Then, the user further decrypts the key ciphertext to recover a content key with a small cost based on the $PDCT$.

- $PDCT\ Generation$
  To improve key ciphertext decryption efficiency of a user the CServer undertakes a large number of computation. A user $U_j (j \in S_U)$ will send its global public key GPK_j, a set of secret keys $\{SK_{j,k}\}_{k\in S_A}$ and its attribute set I_j to the CServer and require it to compute a partially decrypted ciphertext $PDCT$. The CServer first gets attribute public keys $\{PK_{x_k}\}$ relating to the user U_j from the AAM and then computes a common subset of attributes $P_{jM} = \{\rho(i) : i \in L\} \cap I_j$, where L is a set of row numbers in the access matrix $\mathbf{M}$. For these attributes in set P_{jM}, the CServer further checks to find if there exists an index subset I_c of rows of $\mathbf{M}$, such that the vector $(1, 0, \cdots, 0)$ is a linear combination of these rows corresponding to the index subset I_c in matrix $\mathbf{M}$. If not, the decryption is terminated. Otherwise, the CServer proceeds as follows. Let N_A be the number of attribute authorities corresponding to an index set I_A. The CServer calculates a set of constants $\{\xi_i \in \mathbb{Z}_p\}_{i\in I_c}$ such that $\sum_{i\in I_c} \xi_i \mathbf{M}_i = (1, 0, \cdots, 0)$, i.e., a set $\{\lambda_i\}$ is valid shares of the secret $s = \sum_{i\in I_c} \xi_i \lambda_i$. Then, the CServer executes an algorithm $CKGen(KCT, GPK_j, \{SK_{j,k}\}_{k\in S_A}, \{PK_{x_k}\}) \longrightarrow PDCT$ to obtain $PDCT$ and sends it to the user U_j. The $PDCT$ is calculated as follows:
  a) For each attribute authority index $k \in I_A$, to calculate

  $$CT1 = \prod_{k\in I_A} e(C_0, K_{j,k}) \tag{6}$$

  b) For each attribute index $i \in I_c$ relating to the access matrix $\mathbf{M}$, to calculate

  $$\begin{aligned} CT2 = \prod_{i\in I_c} [&e(C_{i,1}, GPK_j) \cdot e(C_{i,2}, K_{j,\rho(i)}) \\ &\cdot e(C_{i,2}C_{i,3}, PK_{\rho(i)}^{-1})]^{\xi_i N_A} \end{aligned} \tag{7}$$

  c) The Cserver computes $PDCT$ as following

  $$PDCT = CT1/CT2 \tag{8}$$

- $Ciphertext\ Decryption$
  A user U_j can execute an algorithm $Decrypt(KC, PDCT, GSK_j) \longrightarrow f$ to get the content key κ and the plaintext f after receiving $PDCT$ from the CServer.

  $$\kappa = KC/(PDCT^{1/GSK_j}) \tag{9}$$

  The user U_j can use κ to decrypt the encrypted file to obtain plaintext as

  $$f = Dec_\kappa(En_\kappa(f)) \tag{10}$$

7) ***Attribute Revocation by the*** AA_k ***and Some Users****:* If an attribute $\tilde{x}_k$ of a user U_φ is revoked from an AA_k, What is the most important for the AA_k is that it will update the version key and public key associated with the revoked attribute $\tilde{x}_k$. The AA_k can execute $UKeyBase(SK_k, \{GPK_j\}_{j\in S_u, j\neq\varphi}, VK_{\tilde{x}_k}) \longrightarrow (SKU_{j,\tilde{x}_k}, CTU_{\tilde{x}_k})$ algorithm which takes as inputs the authority secret key SK_k, the current attribute version key $VK_{\tilde{x}_k}$, namely $v_{\tilde{x}_k}$, and global public keys of some users $\{GPK_j\}_{j\in S_u, j\neq\varphi}$. Initially, the attribute authority AA_k chooses a new random value $v'_{\tilde{x}_k} \in \mathbb{Z}_p$ as the new $VK'_{\tilde{x}_k}$

and obtains the attribute update key, namely $AUK_{\tilde{x}_k} = (v'_{\tilde{x}_k} - v_{\tilde{x}_k})/\beta_k$. The algorithm can output the secret key update $SKU_{j,\tilde{x}_k} = (GPK_j \cdot g^{\beta_k+1})^{AUK_{\tilde{x}_k}/\beta_k}$ and the ciphertext update $CTU_{\tilde{x}_k} = -AUK_{\tilde{x}_k}/\beta_k$. Then, the AA_k updates the public attribute key of the revoked attribute $\tilde{x}_k$ as shown in formula (11) and sends it to the AAM who then updates the corresponding item in Table I.

$$PK'_{\tilde{x}_k} = PK_{\tilde{x}_k} \cdot g^{AUK_{\tilde{x}_k}} \tag{11}$$

Finally, the AA_k sends $SKU_{j,\tilde{x}_k}$ through the AAM to the all non-revoked users. We will then be in the following two attribute revocation phases which consist of key update and ciphertext update.

- $Non-Revoked\ Users\ Updating\ Secret\ Keys$
  When receiving the $SKU_{j,\tilde{x}_k}$, a non-revoked user U_j will execute an algorithm $SKUpdate(SK_{j,k}, SKU_{j,\tilde{x}_k})$ to get $NK_{j,\tilde{x}_k} = K_{j,\tilde{x}_k} \cdot SKU_{j,\tilde{x}_k}$ and then construct a new secret key as

$$\begin{aligned} SK'_{j,k} =& \{K'_{j,k} = K_{j,k}, \\ & \forall x_k \in SA_{j,k}, x_k \neq \tilde{x}_k : K'_{j,x_k} = K_{j,x_k}, \\ & \forall x_k \in SA_{j,k}, x_k == \tilde{x}_k : K'_{j,\tilde{x}_k} = NK_{j,\tilde{x}_k}\} \end{aligned} \tag{12}$$

  These secret key updates $SKU_{j,\tilde{x}_k}$ are different for all non-revoked users, thanks to the unique u_j. Therefore, the revoked user U_φ fails to use update keys of other non-revoked users to update its secret key, guaranteeing security of the shared data. In addition, there is a component of a constant factor 1 in exponent of g of expression K_{j,x_k} in formula (2), which contributes that the revoked user U_φ cannot update its secret key by colluding with any other non-revoked users. Thus, the scheme further guarantees backward security of the data access.
- $AA_k\ Updating\ Ciphertext$
  For an attribute authority AA_k managing some attributes in the IoT storage system, we consider to take full advantage of its capacity and extend its functions to update the ciphertext when revoking an attribute of a user in order to enhance security of the shared data. The AA_k first obtains those ciphertext relating to a revoked attribute $\tilde{x}_k$ from the CServer who can check the access policy embedded into the ciphertext and find out which ciphertext to be updated. After receiving the ciphertext, the AA_k will get some necessary components $CT_{\tilde{x}_k} = \{C_{i,1}, C_{i,2}\}$ associated with the revoked attribute $\tilde{x}_k$ in the ciphertext. Then, it will run an algorithm $CTUpdate(CT_{\tilde{x}_k}, CTU_{\tilde{x}_k}) \longrightarrow \hat{C}$ to produce the new ciphertext components $\hat{C}$ and send it to the CServer. The AA_k updates the ciphertext associated with the revoked attribute $\tilde{x}_k$ through formula (13).

$$\begin{aligned} KCT_{UPD} =& \langle \forall i = 1\ to\ l : \\ & C_{i,1} = g^{a\lambda_i}(g^{v_{\rho(i)}/\beta_{k'}} F(\rho(i)))^{r_i}, \\ & C_{i,2} = g^{-\beta_{k'} r_i}, \rho(i) \in S_{A_{k'}}, k' \in I_A \\ & if\ \rho(i) == \tilde{x}_k : \\ & C'_{i,1} = \hat{C} = C_{i,1} \cdot (C_{i,2})^{CTU_{\tilde{x}_k}} \rangle \end{aligned} \tag{13}$$

  Note that the AA_k just needs to update only a component $C_{i,1}$ which is associated with the revoked attribute $\tilde{x}_k$. This can highly improve efficiency of the attribute revocation. Through ciphertext updating provided by the AA_k, the proposed scheme can guarantee forward security of the access control system that a newly joined user is able to decrypt the ciphertext if he/she possesses sufficient attributes satisfying the access policy. In addition, the new ciphertext updating scheme can also resist collusion attack between a revoked user and the non-revoked users or some AAs (not including the AA_k in charge of the revoked attribute) because the ciphertext update key $CTU_{\tilde{x}_k}$ is possessed only by the AA_k, which enhances backward security.

*8) **Authenticated Access for a Specified User**:* In a scenario, such as transportation system of a smart city based on IoT, a data owner may want to give the access permission to a user U_j who holds an attribute of "reparation" that does not satisfy the access policy "insurance AND traffic-guidance" of the owner. However, if the data owner changes the access policy as "(insurance AND traffic-guidance) OR reparation", the data sharing range will be domain-expanded, which the data owner does not intend to grant. Consequently, we need to implement a flexible authenticated access for a specified user. The access control steps are as follows.

An owner-specified user U_j sends a specific request $Req = g^{\mu_{U_j}}$ to an owner, where the user U_j randomly chooses an exponent $\mu_{U_j} \in \mathbb{Z}_p$. If the owner agrees access request, he/she will add a record into the $UACL$ field in the CServer and set $UValue = \kappa \cdot e(g^a, Req)^{\mu_{owner}}$ which will be exploited by the user U_j, where the owner chooses $\mu_{owner} \in \mathbb{Z}_p$ randomly. In addition, the CServer returns $RNum$ to the owner. Then, the owner sends $Ans = g^{\mu_{owner}}$ and $RNum$ to the user U_j who can compute content key according to formula (14) and further decrypt to get the file successfully.

$$\kappa = UValue/e(Ans, g^a)^{\mu_{U_j}} \tag{14}$$

We note that the number of records in the UACL is a system parameter whose value is set as a constant value Lu. Therefore, when a buffer of the UACL is full, the owner notifies the oldest user of an invalid $RNum$ according to a FIFO rule in order to obtain the vacancy to provide an access authorization to a new user.

V. PERFORMANCE ANALYSIS AND EXPERIMENTS

In this section, we initially analyze security of the proposed SEM-ACSIT scheme. Then we analyze performance of a cloud

storage system for different schemes. Finally, we carry out some computational overhead comparisons through simulation experiments.

A. Security Analysis and Proof

We will demonstrate that only authorized users can decrypt ciphertext stored in the CServer. Some users also cannot collude to decrypt the ciphetext that they are not able to decrypt individually. The proposed scheme can guarantee both forward security and backward security when an attribute of a user is revoked. We can draw the following security conclusions.

$Theorem$ 1. In the SEM-ACSIT scheme just authorized users can decrypt the shared data.

$Proof$: According to CP-ABE, a user is able to decrypt the ciphertext if and only if attributes of the user satisfy an access policy embedded into the ciphertext. It follows that an access matrix $\mathbf{M}$ constructed from the access policy has a subset of rows corresponding to attributes of the user if and only if there exists a set of linear constants $\{\xi_i \in \mathbb{Z}_p\}$, such that $\sum_{i\in I_c} \xi_i \mathbf{M}_i = (1, 0, \cdots, 0)$. From formulas (6) and (7), we have respectively

$$\begin{aligned} CT1 &= \prod_{k\in I_A} e(C_0, K_{j,k}) \\ &= e(g,g)^{a\cdot U_j s N_A} \cdot \prod_{k\in I_A} e(g,g)^{(1/\alpha_k - 1) s z_j} \end{aligned} \tag{15}$$

$$\begin{aligned} CT2 = \prod_{i\in I_c} [e(C_{i,1}, GPK_j) \cdot e(C_{i,2}, K_{j,\rho(i)}) \cdot \\ e(C_{i,2}C_{i,3}, PK_{\rho(i)}^{-1})]^{\xi_i N_A} \\ = e(g,g)^{U_j a N_A \sum_{i\in I_c} \lambda_i \xi_i} \end{aligned} \tag{16}$$

Thus, we have $PDCT$ as follows

$$\begin{aligned} PDCT &= CT1/CT2 \\ &= \prod_{k\in I_A} e(g,g)^{s z_j (1/\alpha_k - 1)} \end{aligned} \tag{17}$$

There exist $\lambda_i = \vec{v}\cdot\mathbf{M}_i$ and $\vec{v}\cdot(1,0,0,\cdots,0) = s$. Thus,

$$\begin{aligned} &KC/(PDCT^{1/GSK_j}) \\ &= \kappa \cdot (\prod_{k\in I_A} e(g,g)^{1/\alpha_k - 1})^s / (\prod_{k\in I_A} e(g,g)^{s z_j (1/\alpha_k - 1)})^{1/z_j} \\ &= \kappa \end{aligned} \tag{18}$$

Finally, the user executes symmetric algorithm through content key κ to obtain plaintext.

For an unauthorized user, there does not exist a subset of rows in an access matrix $\mathbf{M}$ corresponding to attributes of the user, such that $\sum_{i\in I_c} \xi_i \mathbf{M}_i = (1, 0, \cdots, 0)$. Hence, the content key κ cannot be calculated out. ■

$Theorem$ 2. The SEM-ACSIT scheme is resistant to collusion attack between a user and some other users.

$Proof$: In SEM-ACSIT scheme, every legal user is granted for a globally unique identity uid by the certificate authority CA. The secret keys of a user issued by each of AAs are correlated with the uid of the user. Therefore, it is impossible for some users who cannot individually decrypt ciphertext to exchange components of the secret keys to collusively decrypt the ciphertext. On the other hand, suppose that there exists a subset of attributes from colluding users, such that $\Sigma_{i\in I_c} \xi_i \mathbf{M}_i = (1, 0, \cdots, 0)$. However, in computing $CT2$ according to formula (7), different users have different GPK_j and $K_{j,\rho(i)}$, and one of authority secret keys β_k is just only possessed by the AA_k exclusively. Therefore, even if the colluding users combine their attributes together, they still cannot decrypt the ciphertext.

Besides, in a process of the specified user authorizing, even if an illegal user can obtain $RNum$ of other users, he/she still cannot decrypt the ciphertext without the corresponding parameter $\mu_{U_{uid}}$. Hence, our scheme can guarantee the data security. ■

$Theorem$ 3. When revoking an attribute of a user U_φ, the SEM-ACSIT scheme can hold both forward security and backward security.

$Proof$: In SEM-ACSIT scheme, secret keys of all non-revoked users and all pieces of ciphertext in the CServer relating to a revoked attribute will be updated when this attribute of a user U_φ is revoked. Even if the revoked user U_φ can collude with some non-revoked users, he/she still cannot update a set of secret keys since the item $g^{v_{x_k}/\beta_k^2}$ in a user's secret keys from the AA_k can prevent the revoked user U_φ from updating the secret keys through updated keys of collusive non-revoked users. Therefore, the revoked user that has old-version secret keys cannot decrypt the newly generated ciphertext that can be decrypted just through the new-version secret keys, which guarantees backward security of the scheme.

Furthermore, in our SEM-ACSIT scheme ciphertext updating is implemented by the AA_k which manages the revoked attribute and does efforts to keep the ciphertext update private. And the mechanism can resist collusion attack rooted in a cloud if the cloud is responsible for updating ciphertext. Therefore, our scheme can overcome vulnerability problem when revoking an attribute of a user. If a newly-joined user has matching attributes to an access policy embedded into the previously generated ciphertext, he/she still can decrypt the ciphertext, which guarantees forward security of the SEM-ACSIT scheme. ■

$Theorem$ 4. If the decisional q-parallel BDHE assumption holds, then no polynomial time adversary can selectively break SEM-ACSIT scheme with a challenge matrix of size $l^* \times n^*$, where $n^* \leq q - 1$.

$Proof$: We suppose there is an adversary $\mathscr{A}$ with non-negligible advantage $\epsilon = Adv_A$ in the selective security game against our construction. Furthermore, assume that the adversary chooses a challenge matrix $\mathbf{M}^*$ with dimension at most q columns. In the security game, the adversary can query for any private keys that cannot be used for decryption together with any keys it can obtain from the corrupted AAs. We can devise a simulator $\mathscr{B}$ that deals with the decisional q-parallel BDHE problem with non-negligible advantage. Due to limitations of the space we give the detailed theory proof for security in Appendix A. ■

TABLE III
COMPREHENSIVE COMPARISON FOR OUR SCHEME WITH OTHER TWO SCHEMES

Scheme	Authority	Computation Cost		Revocation	Revocation	Ciphertext	Global	Revocation Security		Support of
		Enc.	Dec.♣	Message($\|p\|$)	Controller	Updater	Center	Backward	Forward	Specified user
DACC [18]	Multiple	$O(t_c)$	$O(t_u)$	$O(n_{c,\tilde{x}} \cdot n_{non,\tilde{x}})$	Owner	Owner	No	Yes	No	No
DAC-MACS [19]	Multiple	$O(t_c + t_k)$	$O(1)$	$O(n_{non,\tilde{x}}) + O(n_{owner})$	AA‡	CServer	Yes	No [20]	No	No
SEM-ACSIT	Multiple	$O(t_c)$	$O(1)$	$O(n_{non,\tilde{x}} + d_c) + O(1)$	AA†	AA	Yes	Yes	Yes	Yes

♣: just decryption on a user. ‡: including both non-revoked users and the CServer. †: including only non-revoked users. $|p|$: size of group element.
t_c: number of attributes in an access policy imbedded in the ciphertext. t_k: number of attribute authorities involved in the ciphertext.
n_{owner}: number of all owners in an access control system. t_u: number of attributes in a user.
d_c: number of the ciphertext relating to a revoked attribute $\tilde{x}$ in the CServer.
$n_{c,\tilde{x}}$: number of the ciphertext containing a revoked attribute $\tilde{x}$. $n_{non,\tilde{x}}$: number of non-revoked users holding an attribute $\tilde{x}$.

TABLE IV
COMPARISON OF STORAGE OVERHEAD FOR TWO SCHEMES

Entity	DAC-MACS [19]	SEM-ACSIT
AA_k	$(n_{a,k}+3)\|p\|$	$(n_{a,k}+2)\|p\|$
AAM	/	$(2NN_A + \sum_{k=1}^{NN_A} n_{a,k})\|p\|$
Owner	$(3NN_A + 1 + \sum_{k=1}^{NN_A} n_{a,k})\|p\|$	$\|p\|$
User	$(3NN_A + 1 + \sum_{k=1}^{NN_A} n_{a,k,uid})\|p\|$	$(NN_A + 1 + \sum_{k=1}^{NN_A} n_{a,k,uid})\|p\|$
CServer	$(3t_c + 2 + N_A)\|p\|$	$(3t_c + 2 + L_u)\|p\|$

$n_{a,k}$: number of all attributes managed by AA_k.
NN_A: number of all AAs in a cloud storage system.
$n_{a,k,uid}$: number of attributes assigned to a user uid from AA_k.
N_A: number of attribute authorities corresponding to a set I_A.
t_c: number of attributes in an access policy imbedded in the ciphertext.
L_u: buffer length of the UACL. $|p|$: size of group element.

TABLE V
COMMUNICATION COST COMPARISON IN SOME OPERATIONS

Operation	DAC-MACS [19]	SEM-ACSIT
Data Encryption	$(3t_c + N_A + 2)\|p\|$	$(4t_c + 2N_A + 3)\|p\|$
Public Key Update	$n_{non,\tilde{x}}\|p\|$	$\|p\|$
Ciphertext Update	$\|p\|$	$n_{c,\tilde{x}}(2n_{l,\tilde{x}} + 1)\|p\|$

t_c: number of attributes in an access policy imbedded in the ciphertext.
N_A: number of attribute authorities corresponding to set I_A.
$|p|$: size of group element.
$n_{non,\tilde{x}}$: number of non-revoked users holding an attribute $\tilde{x}$.
$n_{c,\tilde{x}}$: number of the ciphertext containing a revoked attribute $\tilde{x}$.
$n_{l,\tilde{x}}$: average number of attributes in the ciphertext containing $\tilde{x}$.

B. Comprehensive Analysis

We compare proposed SEM-ACSIT scheme with existing schemes, including DACC [18] and DAC-MACS [19], as shown in Table III for data access control in a cloud storage system of IoT with multiple authorities. Among these schemes, both DAC-MACS and SEM-ACSIT have a certificate authority CA that is responsible for managing join of a user or an AA. From the Table III, we can conclude that computational cost of decryption is constant $O(1)$ for SEM-ACSIT similar to DAC-MACS, which is due to decryption outsourcing of these two schemes. However, decryption cost in DACC scheme is determined by the total number t_u of attributes of a user. Therefore, the decryption complexity is $O(t_u)$. Compared with DAC-MACS, SEM-ACSIT reduces encryption overhead by $O(t_k)$. There are two reasons. The first one is that SEM-ACSIT simplifies a key framework and the encryption algorithm. The other is that some heavy pairing computation is transferred to the AAM. Reduction in encryption computation overhead will bring significant benefit to resource-limited IoT data devices.

Moreover, in SEM-ACSIT ciphertext updating is done by the AA_k managing a revoked attribute when the attribute of a user is revoked. Although it increases number of communication messages by $O(d_c)$ as the AA_k is necessary to send updated ciphertext back to the CServer, SEM-ACSIT can guarantee backward security, where d_c is the total number of ciphertext to be updated. In addition, due to storing of attribute public keys in the AAM agent SEM-ACSIT just needs to update them locally, which highly reduces number of communications of the public keys. However, all data owners must access public board of the AA_k to get the newly-updated attribute public key in DAC-MACS, and thus communication overhead is $O(n_{owner})$ that is enormous for a scenario with a large number of data owners, especially in the case of frequent attribute revocation in IoT. Through UACL, SEM-ACSIT scheme provides access authorization for specific users compared with other two schemes, augmenting elasticity of data access. It can be shown from Table III that for DACC decryption cost is overlarge and forward security cannot be guaranteed when a user is revoked. And DAC-MACS lacks security when revoking a user attribute.

In the following section, we will just compare SEM-ACSIT with DAC-MACS because they have the same key frames and these two schemes commonly have the CA.

*1) **Storage Overhead**:* The storage overhead is a critical issue of attribute-based access control scheme in a cloud storage system for resource-limited IoT applications. Comparison of storage overhead with DAC-MACS scheme is shown in Table IV.

In these two schemes storage overhead on each AA_k is resulted from key storage of each attribute managed by the AA_k and that of the AA_k itself. The storage overhead of our scheme is similar to that of DAC-MACS. In SEM-ACSIT, the newly-introduced AAM module has storage overhead of $(2NN_A + \sum_{k=1}^{NN_A} n_{a,k})|p|$ to store authority public keys of all AAs and public keys of all attributes in Tables I and II. Although the AAM adds an extra storage overhead, it can substantially reduce storage overheads of all owners in a system level, which plays an important role in the IoT applications. The two kinds of public keys mainly contribute to storage overhead for an owner. From Table IV we can conclude that for an owner storage overhead of SEM-ACSIT is constant. However, that of DAC-MACS is determined by

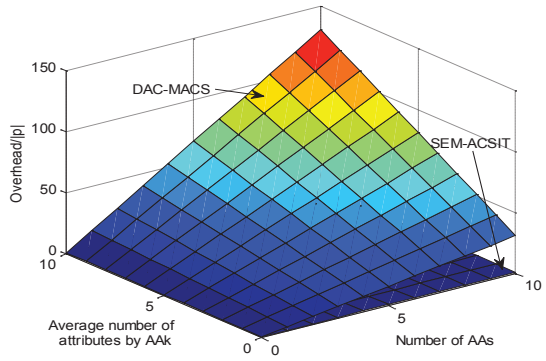

Fig. 5. Storage overhead comparison of an owner for the two schemes.

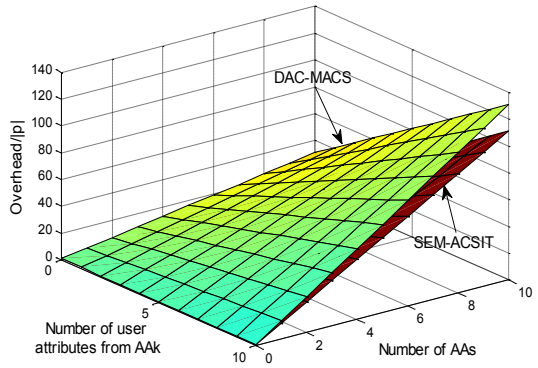

Fig. 6. Storage overhead comparison of a user for the two schemes..

the total number of the AAs and the number of attributes in the AA_k. Consequently, storage overheads of DAC-MACS are far more than those of SEM-ACSIT, especially due to having large redundancy of public keys storage. The storage overhead of a user in both DAC-MACS and SEM-ACSIT schemes includes a global secret key issued by the CA, attribute-irrelevant and attribute-relevant secret keys issued by all AAs. Due to simplified secret key structure, user storage overhead of SEM-ACSIT scheme is $2 * NN_A$ less than that of DAC-MACS scheme. Note that in the CServer the key ciphertext is primary storage overhead considered only, not including the encrypted data because there are the same overheads for the data ciphertext in these two schemes. Compared with DAC-MACS scheme, storage overhead of our scheme is mainly determined by buffer length L_u of the UACL as opposed to the number N_A of attribute authorities involved in the ciphertext. In summary, Fig. 5 and Fig. 6 present storage overhead comparisons of the two schemes for an owner and a user as changed with number of the AAs and average number of attributes managed by the AA_k respectively.

*2) **Communication Cost**:* The communication cost of SEM-ACSIT is similar to that of DAC-MACS in several phases of data access, including initialization phase, secret key generation phase for a user and so forth. However, there exist some differences between the two schemes, especially in both encryption phase and attribute revocation phase. We compare communication cost of the two schemes, as shown in Table V.

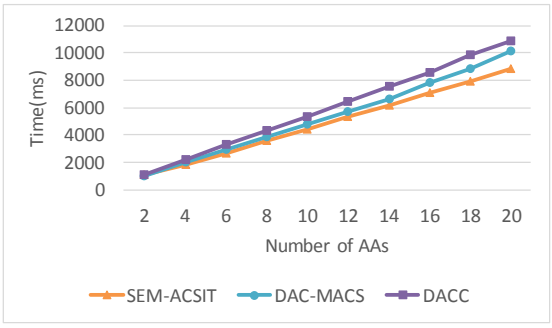

Fig. 7. Encryption time comparison on an owner with varying AAs number.

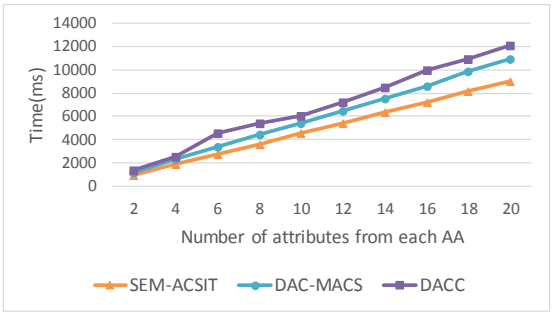

Fig. 8. Encryption time comparison on an owner with varying attribute number from each AA.

Because public keys of all authorities and attributes are stored in the AAM module of SEM-ACSIT scheme, communication cost of our scheme to encrypt data and update ciphertext is a little more than that of DAC-MACS scheme. Due to the large number of data owners in a cloud storage system for IoT applications, DAC-MACS incurs a heavy communication cost for updating the attribute public keys for non-revoked users. However, the AAM in our scheme is functioned as an agent of all AAs and in charge of distributing updated public keys, which can highly reduce the communication cost of AAs, especially in the frequent attribute revocation scenarios. Our SEM-ACSIT scheme guarantees both backward security and forward security at the cost of increasing communication cost by $O(n_{c,\tilde{x}} * n_{l,\tilde{x}})$ as compared to DAC-MACS scheme, which, however, still has big significance because of basic security requirements.

C. Simulation Experiments

In this section, we will compare SEM-ACSIT scheme with DAC-MACS scheme [19] and DACC scheme [18] in encryption and decryption efficiency. We conduct a large number of simulation experiments on a Windows system with the JDK1.8 developing environment and an Intel Core i3-3240 CPU at 3.3GHz and 16.0 GB RAM. In addition, we adopt Java pairing-based cryptography library version 1.2.1 to simulate the three schemes. All simulation results are averaged over 40 simulation tests.

In the experiments, we adopt the same parameters as DAC-MACS scheme due to the similar framework, for example, $|p| = 160$ and compare computation time of both encryption

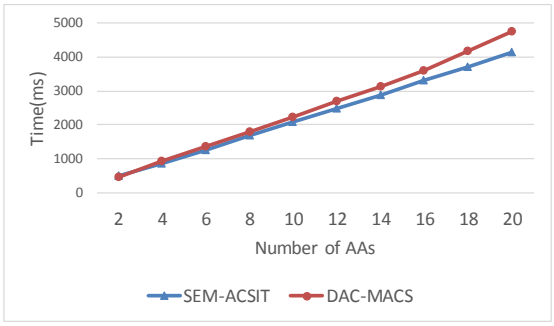

Fig. 9. Decryption time comparison on the CServer with varying AAs number.

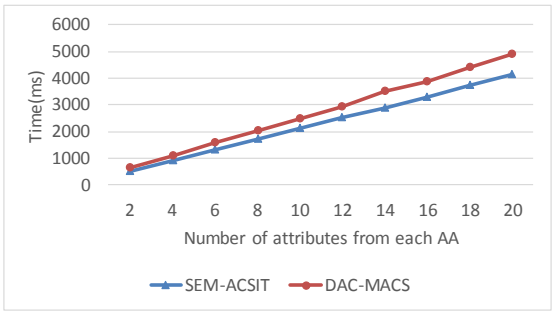

Fig. 10. Decryption time comparison on the CServer with varying attribute number from each AA.

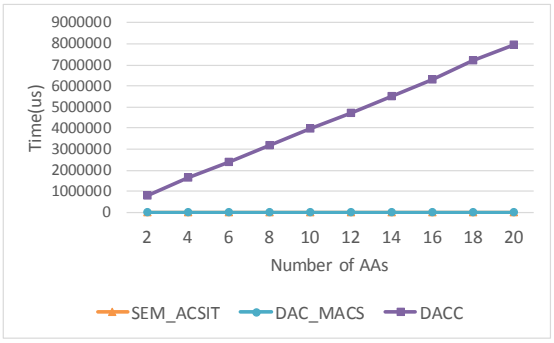

Fig. 11. Decryption time comparison on a user with varying AAs number.

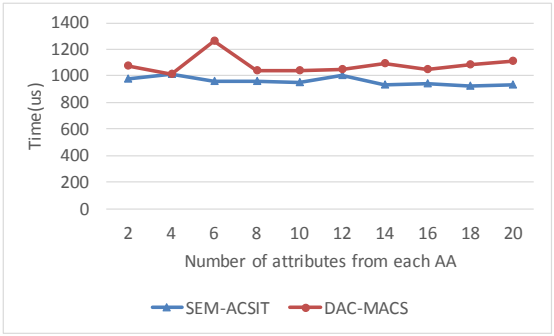

Fig. 12. Decryption time comparison on a user with varying attribute number from each AA.

and decryption in such two different conditions as the number of attribute authorities and the number of attributes managed by each authority. We present the comparisons of encryption time on an owner, decryption time on the CServer and that on a user respectively versus an increasing number of AAs, as shown in Fig. 7, Fig. 9 and Fig. 11, where the number of attributes on each AA is set to be a mean of 10. Moreover, the comparisons of encryption time on an owner, decryption time on the CServer and that on a user respectively, versus a growing number of attributes on each AA, are shown in Fig. 8, Fig. 10 and Fig. 12, where the number of AAs is also set to be a mean of 10.

As shown in Fig. 7, encryption computation overhead of an owner in the three schemes grows linearly with an increasing number of the AAs and encryption computation overhead of SEM-ACSIT is less than that of DAC-MACS and DACC. Fig. 8 demonstrates that encryption computation overhead of an owner in the three schemes is also linear to grow with an increasing number of attributes on each AA and the encryption computation overhead of our scheme is also less than that of DAC-MACS and DACC. The reason of reduction in encryption overhead is that our scheme simplifies an encryption algorithm and a portion of encryption computation is transferred to the AAM. Our scheme is N_A scalar multiplications less than DAC-MACS for each of encryptions. We do some decryption performance comparisons between our scheme and DAC-MACS since the DACC scheme has no cloud decryption outsourcing. Similarly, because of less computation on the CServer, the decryption overhead of our scheme is reduced, as shown in Fig. 9 and Fig. 10. According to formula (6) and (7) our scheme reduces the number of paring operations to compute $e(g, g)$ by N_A times compared with DAC-MACS when decrypting every ciphertext. Whereas, the pairing operation is the most expensive calculation in an access control system. From Fig. 11 and Fig. 12, we can demonstrate that decryption computation overheads of a user in both SEM-ACSIT and DAC-MACS are low since the two schemes outsource a large number of decryption computation to the cloud server and thus the user only needs to perform a few computation. However, the decryption computation overhead of a user in DACC is highly bigger than those in our scheme and DAC-MACS since all decryption computation is done totally by the user for DACC scheme. Besides, for the two schemes of SEM-ACSIT and DAC-MACS computational overhead of a user is irrelevant to the number of AAs and the number of attributes in each AA. Finally, we note that SEM-ACSIT scheme incurs similar updating computation cost to DAC-MACS scheme on a user since computational overhead for the ciphertext updating in DAC-MACS is shifted from the cloud server to the AA_k in SEM-ACSIT to guarantee security.

VI. CONCLUSION

A secure access to shared data stored in a cloud server is becoming a considerable demand for IoT applications. There are many limitations to design an effective and secure data access scheme, such as performance of a cloud server, heterogeneity of data devices, large number of data users, and different requirements for security in IoT. These factors motivated us to propose SEM-ACSIT, an efficient and secure

access control system for IoT applications, where storage overhead of the system is reduced significantly, and both forward security and backward security are guaranteed when an attribute of a user is revoked. In addition, a specific user can be supported to be authorized and access to the shared data with big flexibility. Results from analysis and simulation experiments show that SEM-ACSIT clearly improves storage efficiency with reduced computational overhead, and provides strong data sharing security in a cloud storage system for IoT applications as compared to the existing schemes.

APPENDIX A
PROOF OF THEOREM 4

We show the detailed proof of Theorem 4 here as follows.

$Init$: The simulator $\mathscr{B}$ takes in a q-parallel BDHE challenge $\vec{y}, T$. The adversary gives the algorithm the challenge access structure $(\mathbf{M}^*, \rho^*)$, where $\mathbf{M}^*$ has n^* columns.

$Setup$: The simulator runs two algorithms of CASetup and AASetup respectively and provides g to the adversary. Accordingly, the adversary chooses an authority set $S_{A'} \subset s_A$, which have been corrupted, and reveals these corrupted authorites to the simulator. For every uncorrupted authority $AA_k (k \in S_A - S'_A)$, the simulator chooses two randoms $\alpha_k, \beta_k \in \mathbb{Z}_p$.

Then, we despict how the simulator programs a random oracle $F(x)$, where x is an attribute in the system. For each x the simulator chooses a random value $z_x \in \mathbb{Z}_p$. If there does not exist an i in the access strcture (M^*, ρ^*) such that $\rho^*(i) = x$, then let $F(x) = g^{z_x}$. Otherwise let X denote a set of indices i, and let

$$F(x) = g^{z_x} \prod_{i \in X} g^{aM^*_{i,1}/b_i} g^{a^2 M^*_{i,2}/b_i} \dots g^{a^{n^*} M^*_{i,n^*}/b_i}.$$

Note that due to randomization of the g^{z_x} value the oracle $F(x)$ is distributed randomly. Obviously, by using the $\alpha_k, \beta_k \in \mathbb{Z}_p$ mentioned above the simulator can generate a public key PK_k of each uncorrupted authority AA_k as follows.

$$PK_k = \{e(g,g)^{1/\alpha_k}, g^{\beta_k}\}$$

Similarly, a public key PK_{x_k} associated with an attribute x_k for the authority AA_K can also be generated by randomly choosing a version number v_{x_k} as $PK_{x_k} = (g^{v_{x_k}})^{1/\beta_k}$. The adversary is assigned a user identity u_{id} by the simulator. And the simulator further chooses two random numbers $u'_{uid}, z_{uid} \in \mathbb{Z}_p$ to set $GSK_{uid} = z_{uid}$. Then it implicitly sets $u_{uid} = u'_{uid} - (a^{a+1}/z_{uid})$ by setting

$$GPK_{uid} = g^{u'_{uid}} (g^{a^{q+1}})^{-1/z_{uid}}$$

Then the simulator sends the public/secret key pair (GPK_{uid}, GSK_{uid}) to the adversary.

$Phase1$:In this phase, the simulator answers some private key queries and update key queries from the adversary. Suppose the simulator is given key queries by the adversary through submitting a tuple (uid, S_k), where S_k is a set of attributes belonging to an uncorrupted authority AA_k. Another assumption is that S_k does not satisfy $\mathbf{M}^*$ in combination with any keys which can be obtained from corrupted authorities.

Therefore, the simulator can construct $K_{uid,k}$ as

$$K_{uid,k} = g^{a u'_{uid}} (g^{a^{q+2}})^{-1/z_{uid}} g^{z_{uid}(1/\alpha_k - 1)}.$$

Now, we calculate $K_{uid,x_k} (\forall x_k \in SA_{j,k})$. First, we consider the attribute $x_k \in SA_{j,k}$ for which there is no i in the access structure such that $\rho^*(i) = x_k$. For these attribute x_k, we can simply let

$$K_{uid,x_k} = g^{v_{x_k}((u'_{uid} - a^{q+1}/z_j + 1)/\beta_k^2 + 1/\beta_k)} \times F(x_k)^{u'_{uid}/\beta_k} (g^{a^{q+1}})^{-z_x/\beta_k z_{uid}}.$$

Another task is to create attribute keys for some attribute x_k, where x_k is used in the access structure, namely there exists $\rho^*(i) = x_k$. The simulator creates K_{uid,x_k} as follows.

$$K_{uid,x_k} = g^{v_{x_k}((u'_{uid} - a^{q+1}/z_j + 1)/\beta_k^2 + 1/\beta_k)} \times F(x_k)^{u'_{uid}/\beta_k} (g^{a^{q+1}})^{-z_x/\beta_k z_{uid}} \times \prod_{i \in X} \prod_{j = 1 \cdots n^*} (g^{a^{j+q+1}/b_i})^{\frac{-M^*_{i,j}}{\beta_k z_{uid}}}$$

$Challenge$:In this phase, we build the challenge ciphertext. The adversary gives two messages $m0, m1$ to the simulator. The simulator flips a coin b. It creates

$$KC = m_b T \prod_{k \in I_A} e(g,g)^{(1/\alpha_k - 1)s}$$

and $C_0 = g^s$. The hard part is to simulate the value $C_{i,1}$ since this contains some terms that must be cancelled out. However, the simulator can choose the secret splitting, such that these items can be dealt with. Intuitively, the simulator will choose some random values $y'_2, y'_3, \cdots, y'_{n^*} \in \mathbb{Z}_p$ and share the secret using the vector.

$$\vec{v} = (s, sa + y'_2, sa^2 + y'_3, \dots, sa^{n^*-1} + y'_{n^*}) \in \mathbb{Z}_p^{n^*}.$$

It also chooses random numbers $r'_1, r'_2, \dots, r'_{l^*}$. For $i = 1, 2, \dots, l^*$, let R_i be a set of all $k \neq i$ such that $\rho(i)^* = \rho(k)^*$. That means the set of all other row indices corresponds to the same attribute as row i. We can generate the challenge ciphertext components as

$$C_{i,2} = (g^{r'_i} g^{s b_i})^{-\beta_k}, C_{i,3} = (g^{r'_i} g^{s b_i})^{-1}.$$

From the vector $\vec{v}$, we can also construct the share of the secret as

$$\lambda_i = \vec{v} \cdot \mathbf{M}^*_i \quad s \cdot \mathbf{M}^*_{i,1} + \sum_{j=2,\dots,n^*} (sa^{j-1} + y'_j) \mathbf{M}^*_{i,j}.$$

Then, the challenge ciphertext component $C_{i,1}$ can be simulated as

$$C_{i,1} = (g^{v_{\rho(i)}/\beta_k} F(\rho(i)))^{-r'_i} (\prod_{j=2,\dots,n^*} (g^a)^{\mathbf{M}^*_{i,j} y'_j}) \times (g^{b_i s})^{v_{\rho^*(i)}/\beta_k + z_{\rho^*(i)}} (\prod_{k \in R_i} \prod_{j=1,\dots,n^*} (g^{a^j s b_i / b_k})^{\mathbf{M}^*_{kj}}).$$

$Phase2$: Same as Phase 1.

$Guess$: The adversary will ultimately output a guess b' of b. The simulator then outputs 0 to indicate that $T = e(g,g)^{a^{q+1} s}$

if $b == b'$; otherwise, it outputs 1 to show that it believes T is a random group element in G_T. When T is a tuple, the simulator $\mathscr{B}$ gives a perfect simulation. Therefore we have that

$$Pr[\mathscr{B}(\vec{y}, T = e(g,g)^{a^{q+1}s}) = 0] = 1/2 + Adv_A.$$

When T is a random group element, the message m_b is completely hidden from the adversary and we get at $Pr[\mathscr{B}(\vec{y}, T = R) = 0] = 1/2$. Thus, $\mathscr{B}$ can play the decisional q-parallel BDHE game with non-negligible advantage.

REFERENCES

[1] D. Zhang, S. J. Zhao, L. T. Yang, M. Chen, Y. Wang, and H. Liu, "Nextme: Localization using cellular traces in internet of things," *IEEE Trans. Ind. Informat.*, vol. 11, no. 2, pp. 302–312, 2015.

[2] S. Bera, S. Misra, and J. Rodrigues, "Cloud computing applications for smart grid: A survey," *IEEE Trans. on Parallel and Distributed Systems*, vol. 26, no. 5, pp. 1477–1494, 2015.

[3] S. Xiong, Q. Ni, X. Wang, and Y. Su, "A connectivity enhancement scheme based on link transformation in IoT sensing networks," *IEEE Internet of Things Journal*, vol. 4, no. 6, pp. 2297–2308, 2017.

[4] Q. Chi, H. Yan, C. Zhang, Z. Pang, and L. Xu, "A reconfigurable smart sensor interface for industrial wsn in iot environment," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1417–1425, 2014.

[5] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[6] D. Lee, J. Choi, and H. Shin, "A scalable and flexible repository for big sensor data," *IEEE Sensors Journal*, vol. 15, no. 12, pp. 7284–7294, 2015.

[7] K. Liang, W. Susilo, and J. K. Liu, "Privacy-preserving ciphertext multi-sharing control for big data storage," *IEEE Trans. on Information Forensics and Security*, vol. 10, no. 8, pp. 1578–1589, 2015.

[8] L. Jiang, L. Xu, H. Cai, Z. Jiang, F. Bu, and B. Xu, "An iot-oriented data storage framework in cloud computing platform," *IEEE Trans. Ind. Informat.*, vol. 10, no. 2, pp. 1443–1451, 2014.

[9] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. on Security and Privacy(S&P'07)*, 2007, pp. 321–334.

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

[11] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of abe ciphertexts," in *Proc. 20th USENIX Security Symp (SEC'11)*, 2011, pp. 1–16.

[12] X. Dong, J. D. Yu, Y. Luo, Y. Y. Chen, G. T. Xue, and M. L. Li, "Achieving an effective, scalable and privacy-preserving data sharing service in cloud computing," *Computers and Security*, vol. 42, no. 5, pp. 151–164, 2014.

[13] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *International Workshop on Public Key Cryptography (PKC'11)*, 2011, pp. 53–70.

[14] J. Hur, "Improving security and efficiency in attribute-based data sharing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 10, pp. 2271–2282, 2013.

[15] M. Chase, "Multi-authority attribute based encryption," in *Proc. 4th Theory of Cryptography Conference (TCC07)*, 2007, pp. 515–534.

[16] M. Chase and S. S. M. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *Proc. 16th ACM conference on Computer and Communications Security (CCS09)*, 2009, pp. 121–130.

[17] A. B. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *Proc. 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT11)*, 2011, pp. 568–588.

[18] S. Ruj, A. Nayak, and I. Stojmenovic, "Dacc: Distributed access control in clouds," in *Proc. 10th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom11)*, 2011, pp. 91–98.

[19] K. Yang, X. H. Jia, K. Ren, B. Zhang, and R. Xie, "Dac-macs: Effective data access control for multiauthority cloud storage systems," *IEEE Transactions on Information Forensics and Security*, vol. 8, no. 11, pp. 1790–1801, 2013.

[20] J. A. Hong, K. P. Xue, and W. Li, "Comments on dac-macs: Effective data access control for multiauthority cloud storage systems/security analysis of attribute revocation in multiauthority data access control for cloud storage systems," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1315–1317, 2015.

[21] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT05)*, 2005, pp. 457–473.

[22] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conference on Computer and Communications Security (CCS06)*, 2006, pp. 89–98.

[23] Z. Liu, X. Zhu, and S. Zhang, "Multi-authority attribute based encryption with attribute revocation," in *Proc. 17th IEEE International Conference on Computational Science and Engineering (CSE14)*, 2014, pp. 1872–1876.

[24] J. Zhou, Z. Cao, X. Dong, and X. Lin, "Tr-mabe: White-box traceable and revocable multi-authority attribute-based encryption and its applications to multi-level privacy-preserving e-healthcare cloud computing systems," in *Proc. 10th IEEE Conference on Computer Communications (INFOCOM'15)*, 2015, pp. 2398–2406.

[25] K. Nomura, M. Mohri, Y. Shiraishi, and M. Morii, "Attribute revocable multi-authority attribute-based encryption with forward secrecy for cloud storage," *Ieice Transactions on Information and Systems*, vol. 100, no. 10, pp. 2420–2431, 2017.

[26] X. Wu, J. Rui, and B. Bhargava, "On the security of data access control for multiauthority cloud storage systems," *IEEE Transactions on Services Computing*, vol. 10, no. 2, pp. 258–272, 2017.

[27] K. Xue, Y. Xue, J. Hong, W. Li, H. Yue, D. S. L. Wei, and P. Hong, "Raac: Robust and auditable access control with multiple attribute authorities for public cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 953–967, 2017.

[28] K. Xue, W. Chen, W. Li, J. Hong, and P. Hong, "Combining data owner-side and cloud-side access control for encrypted cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 13, no. 8, pp. 2062–2074, 2018.

[29] J. Wei, W. Liu, and X. Hu, "Secure and efficient attribute-based access control for multiauthority cloud storage," *IEEE Systems Journal*, vol. 12, no. 2, pp. 1731–1742, 2018.

[30] Y. Yahiatene, D. E. Menacer, M. A. Riahla, A. Rachedi, and T. B. Tebibel, "Towards a distributed abe based approach to protect privacy on online social networks," in *Proc. 17th 2019 IEEE Wireless Communications and Networking Conference (WCNC'19)*, 2019, pp. 1–7.

[31] Y. Xue, K. Xue, N. Gai, J. Hong, D. S. L. Wei, and P. Hong, "An attribute-based controlled collaborative access control scheme for public cloud storage," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 11, pp. 2927–2942, 2019.

[32] M. Zhao, Y. Y. Yang, and C. Wang, "Mobile data gathering with load balanced clustering and dual data uploading in wireless sensor networks," *IEEE Transactions on Mobile Computing*, vol. 14, no. 4, pp. 770–785, 2015.

Shuming Xiong received the B.E., M.E., and Ph.D. degrees in 1998, 2003, and 2011, respectively, from Jiangsu University, Zhenjiang, China. Shuming Xiong is an Associate Professor at the School of Computer Science and Communication Engineering, Jiangsu University.

His research interests mainly include data security,connectivity control in WSNs, IoT and other wireless networks,etc. He has published 20+ papers in WSNs and IoT.

Qiang Ni is a Professor and the Head of Communication Systems Research Group, School of Computing and Communications, InfoLab21, Lancaster University, U.K.

His research interests lie in the area of future generation communications and networking, including green communications, 5G, energy harvesting, WSNs, cognitive radio, ultra-dense networks, SDN, IoTs and vehicular networks. He has published 200+ papers in these areas. He is a Voting Member of IEEE 1932.1 standard. He was an IEEE 802.11 Wireless Standard Working Group Voting Member and a contributor to various IEEE Wireless Standards.

Liangmin Wang received the B.S. degree in computational mathematics from Jilin University, Changchun, China, in 1999 and the Ph.D. degree in cryptology from Xidian University, Xian, China, in 2007.

He is currently a Full Professor with the School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China. His current research interests include security protocols and Internet of Things.

Qian Wang received the B.E. and M.E. degrees in 2013 and 2016 respectively, from Jiangsu University, Zhenjiang, China. His research interests include data security,cloud computing, etc.