Peer-to-Peer Netw. Appl. (2018) 11:766–777
DOI 10.1007/s12083-017-0562-8

Secure, efficient and revocable data sharing scheme for vehicular fogs

Kai Fan[1] · **Junxiong Wang**[1] · **Xin Wang**[1] · **Hui Li**[1] · **Yintang Yang**[2]

Received: 31 July 2016 / Accepted: 18 April 2017 / Published online: 6 May 2017

Abstract With the rapid development of vehicular networks, the problem of data sharing in vehicular networks has attached much attention. However, existing data access control schemes in cloud computing cannot be applied to the scenario of vehicular networks, because cloud computing paradigm cannot satisfy the rigorous requirement posed by latency-sensitive mobile application. Fog Computing is a paradigm that extends Cloud computing and services to the edge of the network. The vehicular fog is the ideal platform to achieve data sharing in vehicular networks. In this paper, we propose a revocable data sharing scheme for vehicular fogs. We construct a new multi-authority ciphertext policy attribute-based encryption (CP-ABE) scheme with efficient decryption to realize data access control in vehicular network system, and design an efficient user and attribute revocation method for it. The analysis and the simulation results show that our scheme is secure and highly efficient.

Keywords Vehicular fogs · Attribute-based encryption · Revocation · Security · Efficiency

This article is part of the Topical Collection: *Special Issue on Fog Computing on Wheels*
Guest Editors: Hongzi Zhu, Tom H. Luan, Mianxiong Dong, and Peng Cheng

✉ Kai Fan
kfan@mail.xidian.edu.cn

Junxiong Wang
lihui@mail.xidian.edu.cn

Xin Wang
wangjx921210@163.com

Hui Li
xwang2011@stu.xidian.edu.cn

Yintang Yang
ytyang@xidian.edu.cn

[1] State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an 710071, China

[2] Key Laboratory of Ministry of Education for Wide Band-Gap Semicon, Materials and Devices, Xidian University, Xi'an 710071, China

1 Introduction

During the last decade, with the development of science and technologies, such as cloud computing and cellular networks, vehicular networks and associated applications have been developed dramatically. Especially vehicular networks, which are recognized as a significant component of the future intelligent transportation systems [1, 2], support various mobile services ranging from the content-sharing applications to the information-spreading services [3]. As a consequence of this trend, the problem of data sharing has attached much attention [4]. Therefore, it is necessary to develop a secure and efficient data sharing method.

However, existing data access control schemes in cloud computing cannot be applied to the scenario of vehicular networks, because cloud computing paradigm cannot satisfy the requirements of mobility support, low latency and geographical distribution in addition to location awareness. Since mobile cloud computing uses the client–server communication model [5], it is costly and time consuming when uploading real-time data. Moreover, mobile cloud computing also requires a high quality of network connections with remote infrastructures [6]. As a result, the requirement of efficient and convenient communication and computational is a challenging issue in the design of data sharing schemes for vehicular networks.

To address the above problem, the concept of Fog Computing [7] is proposed. It is also proposed as a technology

to secure the cloud environment. Similar to the cloud, it can also provide storage, computation, and application services to end-users. However, Fog can be distinguished from Cloud by its proximity to end-users, the dense geographical distribution and its support for mobility [8]. The Fog Computing paradigm is well positioned for real time big data analytics, supports densely distributed data collection points, and provides advantages in advertising, personal computing and other applications.

1.1 Related work

Fog Computing is a network between the underlying networks and the clouds. It extends the traditional cloud computing mode to the edge of the network from the network center, which is more widely used in all kinds of services [8]. Compared with the Cloud, which is more centralized, Fog Computing targets the services and applications with widely distributed deployments. Since the Fog is localized, it provides low-latency communication and more context awareness [9]. The Fog suits applications with low-latency communication, video streaming, gaming, AR, etc.

With the concept of Fog Computing, researchers have studied some useful and interesting applications based on it. The application and deployment of Connected Vehicle is enriched by connectivity and interactions: vehicles-to-vehicles, vehicle to access points, and access point to access point. It becomes an ideal platform by mobility, low latency, and supports for real-time interactions of Fog.

Another work [10] points out that Fog Computing brings cloud resources close to the underlying devices and Internet of Things, which makes it ideal for latency-sensitive services. They present the idea of utilizing the services of fog for offloading and preprocessing purposes to provide a quick way of notifying the relevant emergency-dealing department.

K. Hong et al. proposed the concept of mobile Fog [11]. Mobile Fog is a high level programming model consisting of a set of event handlers and functions for future Internet applications. They also proposed using parallel resources to mitigate large speed of mobile and taking several predictions for each time step. B. Ottenwalder et al. presented a placement and migration method for Cloud and Fog resources providers [12]. It ensures application-defined end-to-end latency restrictions and reduces the network utilization by planning the migration ahead of time. But this work does not optimize workload mobility or the size of control information.

In the mobile Cloud concept [13], pervasive mobile devices share their heterogeneous resources and support services. Neighboring nodes in a local network form a group called a local Cloud. Nodes share their resources with other nodes in the same local Cloud. A local resource coordinator serving as Fog device is elected from the nodes in each local Cloud. Moreover, some data sharing schemes [14–17] for mobile networks are proposed.

To achieve secure and efficient data sharing for Vehicular Fogs, we use the method of CP-ABE (Ciphertext Policy Attribute-Based Encryption), which is regarded as one of the most suitable technologies for data access control in cloud storage systems.

There are two types of CP-ABE systems: single-authority CP-ABE where all attributes are managed by a single authority, and multi-authority CP-ABE where attributes are from different domains and managed by different authorities. However, in many applications, users may hold attributes issued by multiple authorities and data owners may also share the data using access policy defined over attributes from different authorities. Therefore, multi-authority CP-ABE is more appropriate for data access control of cloud storage systems.

However, the revocation issue is difficult in CP-ABE systems. Since there are many users in a cloud storage environment, users may change frequently. Moreover, users' attributes can be changed dynamically. A user may be entitled some new attributes or revoked some current attributes. And his permission of data access should be changed accordingly.

To deal with the challenging revocation issue in CP-ABE systems, several user revocable CP-ABE schemes have been proposed. Ostrovsky et al. [18] first proposed a fine-grained user revocation scheme based on CP-ABE that supports negative clause. However, the size of the users' keys and the ciphertext are larger. In addition, two user revocation schemes are proposed respectively by Rafaeli and Boyen [19, 20], but both of them have some shortcomings. Attrapadung et al. came up with three user-revocable ABE scheme. However, they are not applicable in the outsourcing environment. Later, Liang et al. [21] also proposed a CP-ABE scheme with efficient user revocation. In this scheme, once a user leaves from a single user set, he will not be able to access the data again. However, this construction has low efficiency in practice.

To achieve revocation on attribute level, several attribute revocation schemes are proposed by setting expiration time on each attribute. However, this method cannot achieve immediate attribute revocation, and the key updating phase can be the bottleneck of the system. Some other re-encryption-based attribute revocation schemes are proposed by relying on a trusted server. We know that the cloud server cannot be fully trusted by data owners, thus traditional attribute revocation methods are no longer suitable for cloud storage systems. With the purpose of reducing the computation overhead of data service manager, Xie et al. [22] proposed a new CP- ABE construction with efficient user and attribute revocation. Compared with Hur's [23], in the key update phase, the computation load of the data service manager will be reduced by half.

1.2 Our contribution

In this paper, we propose a revocable data sharing scheme for vehicular fogs. Here we construct a new

multi-authority CP-ABE scheme with efficient decryption to realize data access control in vehicular network system, and design an efficient user and attribute revocation method for it.

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

1) We construct a secure and efficient data sharing scheme for Vehicular Fogs. Any vehicle in vehicular network system can share their resources with others.

2) We propose a new multi-authority CP-ABE scheme with efficient decryption, while still keeping the CP-ABE system secure against the collusion attack. The main computation of decryption is outsourced to the cloud server. We design an efficient user and attribute revocation method for multi-authority CP-ABE scheme that achieves both forward security and backward security.

3) We provide security analysis and performance analysis of the proposed scheme, and the experimental results show the efficiency of our scheme.

1.3 Organization

The remaining of this paper is organized as follows. We first give some preliminaries in Section 2. Then, we give the definition of the system model and framework in Section 3. In Section 4, we propose a revocable data sharing scheme based on CP-ABE for vehicular fogs. Section 5 gives the security and the performance analysis of our scheme. Finally, the conclusion is given in Section 6.

2 Preliminaries

In this section, we briefly describe some fundamental backgrounds used in this paper, including bilinear maps, access structure and linear secret sharing schemes (LSSS).

2.1 Bilinear maps

Definition 1 (bilinear maps) Let G_1,$G_2$and $G_T$be three cyclic groups of prime order$p$. A bilinear map is a map $e: G_1 \times G_2 \rightarrow G_T$with the following properties:

1) Bilinearity: for all$g_1 \in G_1$, $g_2 \in G_2$and$a$, $b \in Z_p$, $e\left(g_1^a, g_2^b\right) = e(g_1, g_2)^{ab}$.
2) Non-degeneracy: there exists$g_1 \in G_1$,$g_2 \in G_2$ such that$e(g_1, g_2) = 1$.
3) Computability: there is an efficient algorithm to compute $e(g_1, g_2)$for any $g_1 \in G_1$and$g_2 \in G_2$.

2.2 Access structure

Definition 2 (access structure [24]) Let $\{P_1, P_2, \cdots, P_n\}$be a set of parties. A collection$A \subseteq 2^{\{P_1, P_2, \cdots, P_n\}}$ is monotone if$\forall B$, C: if$B \in A$and$B \subseteq C$then$C \in A$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $A$of non-empty subsets of$\{P_1, P_2, \cdots, P_n\}$, i.e.,$A \subseteq 2^{\{P_1, P_2, \cdots, P_n\}} \setminus \{\phi\}$. The sets in A are called the authorized sets, and the sets not in$A$are called the unauthorized sets.

2.3 Linear secret sharing schemes

Definition 3 (linear secret-sharing schemes (LSSS) [24]) We recall the description of LSSS as follows [24]. LetΠbe a secret sharing scheme over a set of parties$\wp$with realizing an access structure$\mathbb{A}$. We say that$\Pi$is a linear secret sharing scheme over$\mathbb{Z}_p$if:

1) the piece for each party forms a vector over $\mathbb{Z}_p$.
2) during the generation of the pieces, the dealer chooses independent random variables, denotedr_2 , $\cdots$, r_n, each one distributed uniformly over$\mathbb{Z}_p$. Each coordinate of the piece of every party is a linear combination of r_2 , $\cdots$, r_n and the secret$s$. That is, let$M$denotes a matrix with$l$rows and$n$columns. For the vector $\overrightarrow{v}^T = (s, r_2, \cdots, r_n)$ and any authorized set, there exist constants$\{w_i \in \mathbb{Z}_p\}_{i \in I}$such that, if$\{\lambda_i\}$are valid shares of any secret$s$according toΠ, then$\sum_{i \in I} w_i \lambda_i = s$, where $\lambda_i = (M\overrightarrow{v})_i$ and$I \subset \{1, 2, \cdots, l\}$.

3 System model and framework

3.1 System model

A simple three level hierarchy is adopted in our system as Fig. 1. In this framework, each vehicle is attached to one of fog devices. Fog devices can be interconnected and each of them is linked to the cloud.

In the system, fog devices at the edge are responsible for storing and transmitting the data encrypted by data owners (vehicles), which are connected to the fog devices through a short-range communications such as Wi-Fi, ZigBee, Bluetooth, etc., and issue control commands to actuators. Those data and computation which needs more computing power are sent to the cloud from the fog devices through high-speed wire or wireless communication. As it is expensive and time-consuming to send all of data from vehicles to the cloud directly

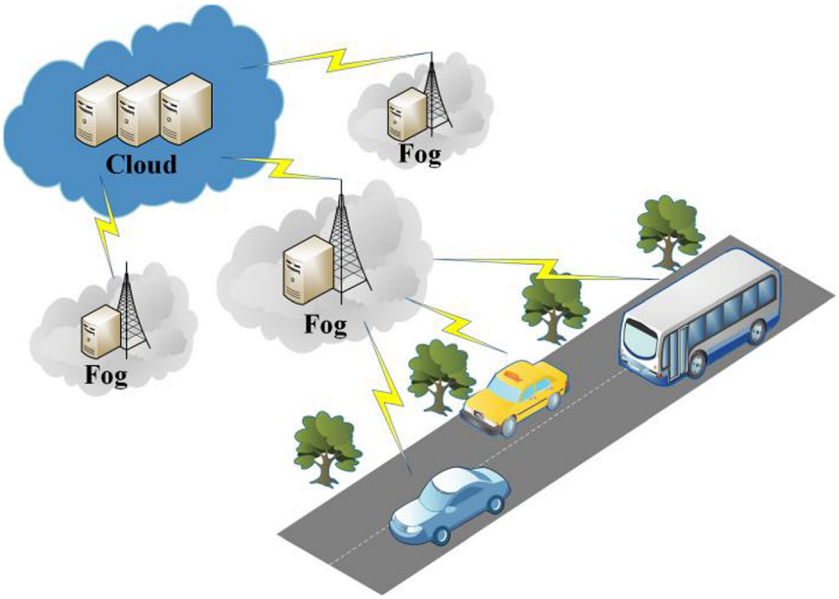


Fig. 1 The vehicular network system model in Fog and Cloud environment

through the high latency network, fog layer is positioned nearby vehicles and autonomously processes data in real time nearby the network edge. Because fog devices have more memory or storage ability for computing, it is immediately possible to process a significant amount of data from data owners.

In addition, to improve the efficiency of data sharing, the method of cache [25] is adopted in our system. Fog devices can store a certain amount of data encrypted by the cloud so that any users managed by this fog device can easily access the data from the buffer without asking the cloud. Once a user requests to access a file, the fog device will first check whether the file exists in the buffer. If the file exists in the buffer, then the user can decrypt it with his secret key. Otherwise, he has to send an access request to the cloud, after that the cloud will send the file to the fog. Especially, once the buffer is full, all the data in the buffer has to be cleared so that new data can be stored.

As for the framework of the cloud, it consists of five types of entities: the cloud server (server), a global certificate authority (CA), the attribute authorities (AAs), the data owners (owners) and the data consumers (users), as shown in Fig. 2.

The CA is a fully trusted global certificate authority which is responsible for issuing a global unique identity*UID*for each user and a unique identity*AID*for each AA in the system. However, the CA is not involved in any attribute management and any generation of secret keys which are associated with attributes.

Each AA is an independent attribute authority that is responsible for issuing, revoking and updating users' attributes within its administration domain. In our scheme, each AA is responsible for generating a public attribute key for each attribute it manages and a private key which consists of transformation key and secret key for each user. The transformation key is stored at the server and secret key is kept by the user.

The data owners define access control policies over attributes from multiple attribute authorities and then encrypt the data under the policies. After that, the owners send the encrypted data to the cloud server.

The server stores the owners' data and provides data access service to users. Moreover, it can help the users decrypt the ciphertext. Only the users whose attributes satisfy the access control policy can the server decrypt the ciphertext with their transformation keys. After that, it sends the partially decrypted data to the corresponding users.

The users can request their secret keys from the relevant authorities. After downloading any encrypted data file shared on the cloud, the users first ask the server to decrypt it with their transformation keys. Upon receiving the partially decrypted data from the server, the users further decrypt it with their secret keys.

However, there are some security issues that have to be considered in our system [26]. According to the fog devices, there exists a potential attack which is man-in-the-middle attack. In this attack, gateways serving as fog devices maybe compromised or replaced by fake ones. Traditional anomaly detection methods can hardly expose man-in-the-middle attack without noticeable features of this attack collected from the fog [27]. In some scenarios, it is difficult to protect communication between fog devices and terminal devices using encryption method. Encryption and decryption methods consume large amount of battery on mobile device. Therefore, in our system we assume that fog devices are trusted and they will not be attacked by malicious users.

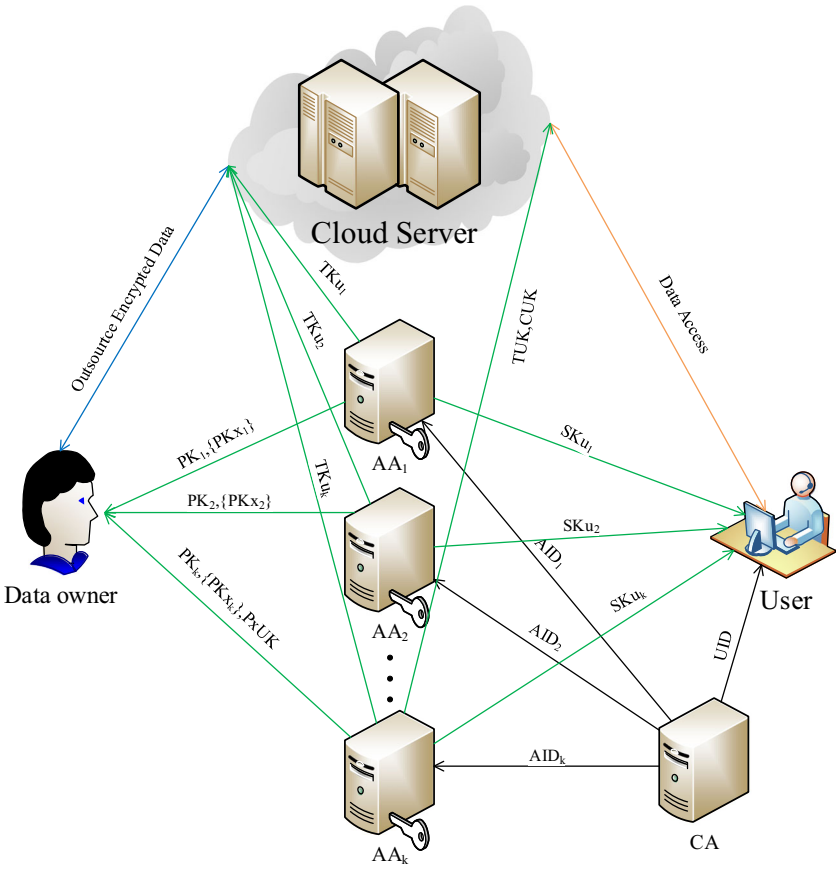


Fig. 2 System model of multi-authority access control in cloud storage

On the other hand, in our multi-authority cloud storage systems, we assume that the CA is trusted in the system, but we still need to prevent it from decrypting any ciphertext. Each authority is also trusted, but it can be corrupted by the adversary. The server is curious but honest. It is curious about the content of the encrypted data or the received message, but will execute correctly the task assigned by each authority. The users are dishonest and may collude to obtain unauthorized access to data.

3.2 Framework

Definition 4 (multi-authority access control scheme) **CASetup** $(1^{\lambda})\rightarrow\{GP, UID, AID\}$**:** the CA Setup algorithm is run by the CA. It takes no input other than the implicit security parameterλ. It outputs the global parameterGP, the authority identity$AID$and the user identityUID.

AASetup $(AID)\rightarrow\{PK_k, SK_k, \{PK_{x_k}\}\}$**:** the attribute authority setup algorithm is run by each AA. It takes the authority identityAIDas input. It outputs public attribute keys$\{PK_{x_k}\}$ for all attributes issued by each AA and a pair of authority public keys$PK_k$and authority secret keySK_k.

Encrypt $(GP, \{PK_k\}_{k\in I_A}, \{PK_{x_k}\}_{x_k\in S_{A_k}}^{k\in I_A}, M, \mathbb{A})\rightarrow\{CT\}$**:** the encryption algorithm is run by the data owner. It takes as inputs the global parameterGP, a set of public keys$\{PK_k\}_{k\in I_A}$ from the involved authority setI_A, a set of public attribute keys $\{PK_{x_k}\}_{x_k\in S_{A_k}}^{k\in I_A}$, a message$M$and an access structure$\mathbb{A}$. It outputs the ciphertextCT.

KeyGen $(GP, S_U, UID, \{PK_{x_k}\}, SK_k)\rightarrow\{TK_{j,k}, SK_j\}$**:** the key generation algorithm is run by each AA. It takes as inputs the global parameterGP, a set of attributes of the userS_U, the user identityUID, the public attribute key$\{PK_{x_k}\}$ and the authority secret keySK_k. It outputs the transformation keys$TK_{j,k}$and the user secret keysSK_j.

Transform $(TK_{j,k}, CT)\rightarrow CT'$**:** the transformation algorithm is run by the cloud server. It takes as inputs the transformation key$TK_{j,k}$and the ciphertextCT. It outputs the partially decrypted ciphertextCT'.

Decrypt $(CT', SK_j)\rightarrow M$**:** the decryption algorithm is run by the users. It takes as inputs the partially decrypted ciphertext$CT'$and the user secret keySK_j. It outputs the message$M$.

RekeyUpdate $(UID, L_{j,k}, D_i, v_{\tilde{x}_k})\rightarrow\{PxUK, TUK, CUK\}$**:** the rekey update algorithm is run by the involved authorities. It takes as inputs$UID$of each non-revoked user, $L_{j,k}$of the transformation key$TK_{j,k}$, $D_i$of the ciphertext$CT$and the current attribute version key$\tilde{v}_{x_k}$. It outputs the public attribute update key$PxUK$, the transformation update keys$TUK$and the ciphertext update keyCUK.

PKxUpdate ($PK_{\tilde{x}_k}$, PxUK)→$PK^*_{x_k}$**:** the PKx update algorithm is run by the data owner. It takes as input the current public attribute key$PK_{\tilde{x}_k}$ and the public attribute update key$PxUK$.It outputs a new public attribute key$PK^*_{x_k}$.

TKUpdate ($TK_{j,k}$, UID, TUK)→$TK^*_{j,k}$**:** the TK update algorithm is run by the cloud server. It takes as inputs the current transformation keys$TK_{j,k}$and the transformation update keysTUK. It outputs a new transformation key$TK^*_{j,k}$ for each non-revoked user who has the attribute$\tilde{x\sim}_k$.

ReEnc (CT, CUK)→CT^***:** the re-encryption algorithm is run by the cloud server. It takes as inputs the current ciphertext$CT$and the ciphertext update keyCUK. It outputs a new ciphertextCT'.

4 Our construction

In this section, we first give an overview of the challenges and techniques of designing multi-authority access control schemes for cloud storage systems. Then, we propose the detailed construction of multi-authority CP-ABE method and use it to design a data sharing scheme based on CP-ABE for vehicular fogs with user and attribute revocation.

4.1 Overview

One challenge issue in the design of multi-authority CP-ABE scheme is how to tie the secret keys from different authorities together and prevent collusion attack.

In our method, a certificate authority (CA) is introduced to assign a global user identity$UID$to each user and a global authority identity$AID$to each authority. Since UID is unique in the system, secret keys with the sameUIDcan be tied together for decryption and it can prevent the collusion attack. Also, since each authority is associated with anAID, all the attributes are distinguishable even though some attributes present the same meaning. Thus, the collusion attack can be resisted by usingAIDandUID.

The other challenge issue is the problem of revocation. According to user revocation, in our scheme, we do not need to update other unrevoked users' secret keys and re-encrypt the ciphertext. The only operation we need is to delete the revoked user's transformation key. Once the$TK$is eliminated, the cloud server is no longer able to perform the transformation algorithm for the revoked user. Thus, the revoked user cannot decrypt the data he wants.

For attribute revocation, the method of attribute version number is introduced in our scheme. We first choose a version number for every attribute in the system. When an attribute revocation occurs, only those involved components in secret keys and ciphertext need to be updated. The involved authority will generate a new version key for the revoked attribute and generate the update keys which contain a public attribute update key, a transformation update key and a ciphertext update key. With these update keys, the involved components in the ciphertext can be updated to the current version, and each non-revoked user can also decrypt the new ciphertext.

Especially, to reduce the computation overhead of users, both the key update and ciphertext re-encryption operations are delegated to the cloud, thus the decryption time for users can be reduced much.

4.2 Our construction

The construction of our access control scheme consists of four phases: System Initialization, Key Generation, Encryption and Decryption.

Phase 1: System Initialization

There are two steps in the system initialization phase: CA Setup and AA Setup.

1) **CA Setup** (1^λ)→{GP, UID, AID}**:**

The CA runs the CA Setup algorithm, which takes a security parameter as input. Let$\mathbb{G}_1$, $\mathbb{G}_2$ and$\mathbb{G}_T$ be the multiplicative groups with the same prime order$p$, and$e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be the bilinear map. Let$g_1$be the generator of$\mathbb{G}_1$ and$g_2$be the generator of$\mathbb{G}_2$. Let$H : \{0,1\}^* \to \mathbb{G}_2$ be a hash function such that the security will be modeled in the random oracle. The CA then chooses a random number$a \in \mathbb{Z}_p$and sets the global parameter $GP = \{g_2^a, g_1, g_2, H\}$.

Every AA and user should register itself to the CA during the system initialization. The CA then assigns a unique global authority identity$AID$to each legitimate AA and a unique global user identity$UID$to every legitimate user.

2) **AA Setup** (AID)→$\{PK_k, SK_k, \{PK_{x_k}\}\}$ **:**

Each authority runs the AA Setup algorithm. LetS_{A_k} denote the set of all attributes managed byAA_k.AA_kfirst chooses two random exponents$\alpha_k, \beta_k \in \mathbb{Z}_p$. For each attribute$x_k \in S_{A_k}$, AA_kchooses an attribute version key as$VK_{x_k} = v_{x_k}$ and then generates the public attribute keys as$\{PK_{x_k}\} = g_2^{v_{x_k}} H(x_k)$.Then it publishes $PK_k = \{e(g_1, g_2)^{\alpha_k}\}$ as its public key and keeps$SK_k = \{\alpha_k, \beta_k\}$as its secret key.

Phase 2: Encryption

Encrypt $\left(GP, \{PK_k\}_{k\in I_A}, \{PK_{x_k}\}_{x_k \in S_{A_k}}^{k\in I_A}, M, \mathbb{A}\right) \to \{CT\}$:

The encryption algorithm takes as inputs the global parameter GP, the public keys $\{PK_k\}_{k\in I_A}$, the public attribute keys $\{PK_{x_k}\}_{x_k\in S_{A_k}}^{k\in I_A}$, the message M and an access structure (A, ρ) over all the selected attributes from the involved authorities. Let A be a $l \times n$ matrix, where l denotes the total number of all the attributes. The function ρ maps rows of the matrix A to attributes.

The encryption algorithm first chooses a random encryption exponent $s \in \mathbb{Z}_p$ and a random vector $\vec{v} = (s, y_2, \cdots, y_n) \in \mathbb{Z}_p^n$ with s as its first entry, where $y_2, \cdots, y_n$ are used to share the encryption exponent s. For $i = 1, \cdots, l$, it computes $\lambda_i = \vec{v} \cdot A_i$, where A_i is the vector corresponding to the i-th row of A. Then it randomly chooses $\gamma_1, \gamma_2, \cdots, \gamma_l \in \mathbb{Z}_p$ and computes the ciphertext as.

$$CT = \left\{ C = M \cdot e(g_1, g_2)^{\sum_{k\in I_A} \alpha_k s}, C' = g_1{}^{s}, \left(C_i = g_2{}^{a\lambda_i} \cdot \left(g_2^{v_{\rho(i)}} \cdot H(\rho(i)) \right)^{-\gamma_i}, D_i = g_1^{\gamma_i} \right)_{i\in\{1,\cdots,l\}} \right\} \tag{1}$$

Phase 3: Key Generation

KeyGen $(GP, S_u, UID, \{PK_{x_k}\}, SK_k) \rightarrow \{TK_{j,k}, SK_j\}$:

Let S_A and S_U denote the set of authorities and the set of users in the system respectively. Each authority first assigns a set of attributes $S_{j,k} (j \in S_U, k \in S_A)$ to each legal user, then chooses a random number $z_j \in \mathbb{Z}_p$ for each user and let $SK_j = \{z_j\}$ as the user's secret key. Each AA then runs the KeyGen algorithm to generate the transformation key as

$$TK_{j,k} = \left\{ K_{j,k} = g_2^{\frac{\alpha_k}{z_j}} g_2^{\frac{a\beta_k}{z_j}}, L_{j,k} = g_1^{\frac{\beta_k}{z_j}}, \{K_{j,\rho(k)}\}_{\rho(k)\in S_{j,k}} = \left(g_2^{v_{\rho(k)}} H(\rho(k)) \right)^{\frac{\beta_k}{z_j}} \right\} \tag{2}$$

$TK_{j,k}$ is used for data decryption and it is stored in the cloud server.

Phase 4: Decryption

Each legal user in the system can query the encrypted data from the cloud server. But only those users whose attributes satisfy the access structure can decrypt the data. The decryption consists of two steps.

1) **Transform** $(TK_{j,k}, CT) \rightarrow CT'$**:**

When a user wants to download a file in the system, the cloud server will first check his transformation key. If the corresponding attributes does not satisfy the access structure, the cloud server outputs $\perp$. Otherwise it chooses a set of constants $\omega_i \in \mathbb{Z}_p$ such that, if λ_i are valid shares of the secret s according to $\mathbb{A}$, then $\sum_{i\in I} \omega_i \lambda_i = s$, where $I = \{1, \cdots, l\}$.

Then the cloud server computes:

$$\prod_{k\in I_A} \frac{e(C', K_{j,k})}{\prod_{i\in I} \left(e(C_i^{\omega_i}, L_{j,k}) \cdot e(D_i^{\omega_i}, K_{j,\rho(i)}) \right)} = e(g_1, g_2)^{\sum_{k\in I_A} \frac{\alpha_k s}{z_j}} \tag{3}$$

Thus the cloud server gets the partially decrypted data $CT' = e(g_1, g_2)^{\sum_{k\in I_A} \frac{\alpha_k s}{z_j}}$ and then sends it to the user.

2) **Decrypt** $(CT', SK_j) \rightarrow M$**:**

Upon receiving the data from the cloud server which is partially decrypted, the user runs the decryption algorithm to decrypt the ciphertext by using its secret key SK_j. It computes M as

$$M = \frac{C}{CT'^{z_j}} \tag{4}$$

4.3 Revocation scheme

1) **User Revocation**

User revocation is executed whenever to restrict a user from accessing the outsourced data files again. In our scheme, when user revocation happens, we do not

need to update other unrevoked users' secret keys and re-encrypt the ciphertext. The only operation we need is to delete the revoked user's *TK*. Once the *TK* is eliminated, the cloud server is no longer able to perform the transformation algorithm for the revoked user. Thus, the revoked user cannot decrypt the data he wants.

2) **Attribute Revocation**

There are two phases in attribute revocation: *Key Update* and *Ciphertext Update*.

Phase1: Key Update

The key update includes three steps: *Rekey Update*, *PKx Update* and *TK Update*.

(1) **RekeyUpdate** $(UID, L_{j,k}, D_i, v_{\tilde{x}_k}) \rightarrow \{PxUK, TUK, CUK\}$**:**

Here *UID* denotes all other users except the revoked user with UID'. The involved authority AA_k first generates a new attribute version key $v'_{\tilde{x}_k}$. It then computes the public attribute update key as $\mathrm{PxUK} = \frac{\tilde{v}'_{x_k}}{\tilde{v}_{x_k}}$, the transformation update keys as $TUK = g_2^{\frac{\beta_k\left(v'_{\tilde{x}_k} - v_{\tilde{x}_k}\right)}{z_j}}$ for each non-revoked user who has the attribute $\tilde{x}_k$ and the ciphertext update key as $CUK = g_2^{-\gamma_i\left(v'_{\tilde{x}_k} - v_{\tilde{x}_k}\right)}$. Then it sends *PxUK* to the data owner to update the public attribute key PK_{x_k}, and sends *TUK* and *CUK* to the cloud server to update $TK_{j,k}$ and *CT*.

(2) **PKxUpdate** $(PK_{\tilde{x}_k}, \mathrm{PxUK}) \rightarrow PK^*_{x_k}$ **:**

Upon receiving the public attribute update key*PxUK*, the data owner updates the public attribute key as

$$PK^*_{x_k} = (PK_{x_k})^{\mathrm{PxUK}} \tag{5}$$

(3) **TKUpdate** $(TK_{j,k}, UID, TUK) \rightarrow TK^*_{j,k}$ **:**

Upon receiving the transformation update keys*TUK*, the cloud server runs the transformation key update algorithm to update the corresponding transformation keys as $K^*_{\rho(i)=\tilde{x}_k} = K_{\rho(i)=\tilde{x}_k} \cdot TUK$ for each non-revoked user who has the attribute $\tilde{x}_k$.

Thus the transformation keys *TK* can be updated as

$$TK^* = \left\{ \begin{array}{l} K^*_j = g_2^{\frac{a}{z_j}} g_2^{\frac{\alpha_k \beta_k}{z_j}}, L = g_1^{\frac{\beta_k}{z_j}} \\ K^*_{\rho(i)\neq\tilde{x}_k} = \left(g_2^{v_{x_k}} H(\rho(i))\right)^{\frac{\beta_k}{z_j}} \\ K^*_{\rho(i)=\tilde{x}_k} = \left(g_2^{v'_{\tilde{x}_k}} H(\rho(i))\right)^{\frac{\beta_k}{z_j}} \end{array} \right\} \tag{6}$$

Phase 2: Ciphertext Update

$\mathrm{ReEnc}(CT, CUK) \rightarrow CT^*$:

Upon receiving the ciphertext update keys*CUK*, the cloud server runs the ciphertext update algorithm to update the corresponding ciphertext as $C^*_i = C_i \cdot CUK$.

Then the new ciphertext CT^* is published as

$$CT^* = \left\{ \begin{array}{l} C = M \cdot e(g_1, g_2)^{\sum \alpha_k s}, C' = g_1^s \\ \left(C_i = g_2^{a\lambda_i} \cdot \left(g_2^{v_{x_k}} \cdot H(\rho(i))\right)^{-\gamma_i}, D_i = g_1^{\gamma_i}\right)_{\rho(i)\neq\tilde{x}_k, i\in\{1,\cdots,l\}} \\ \left(C_i = g_2^{a\lambda_i} \cdot \left(g_2^{v'_{x_k}} \cdot H(\rho(i))\right)^{-\gamma_i}, D_i = g_1^{\gamma_i}\right)_{\rho(i)=\tilde{x}_k, i\in\{1,\cdots,l\}} \end{array} \right\} \tag{7}$$

5 Analysis of our scheme

In this section, we present the security analysis and performance analysis of our multi-authority CP-ABE access control scheme.

5.1 Security analysis

The security analysis of our multi-authority CP-ABE access control scheme contains the following available properties.

Table 1 Comparison of flexibility

Schemes	Access Structure	Authority	Against collusion attack	Revocation
Chase [28]	Only 'AND'	Multiple	Yes	No
Hur [23]	Access tree	Single	Yes	Attribute and user
Lewko [29]	LSSS	Multiple	Yes	No
Ruj [30]	Access tree	Multiple	Yes	User
Our scheme	LSSS	Multiple	Yes	Attribute and user

1) Collusion tolerant

Our multi-authority CP-ABE access control scheme is secure against the collusion attack for any number of users. Suppose that the number of authorities that are involved in the ciphertext is *n*and the number of colluding authorities is *m*. If $m = n$, intuitively, these authorities can obtain all the secret keys which can be used to decrypt the ciphertext. If $m \leq n-1$, their exists at least one authority from which the secret key cannot be obtained, thus the ciphertext cannot be decrypted. Therefore, our scheme achieves collusion tolerance of up to$(n-1)$authorities.

2) Data confidentially

The outsourced data can be confidential against a user whose attributes cannot satisfy the access policy. Since the attributes cannot satisfy the access structure in the ciphertext, the user cannot recover the intermediate value $\left(\prod_{k\in I_A} e(g_1,g_2)^{\frac{\alpha_k}{z_j}}\right)^s$ during the decryption process. On the other hand, when a user is revoked, the cloud server will delete his transformation key. Without the transformation key, he cannot recover the intermediate value either.

In addition, since the cloud server is trustless, the outsourced data should also be confidential against the cloud. In our scheme, though the cloud obtains the users' transformation keys, it can only obtain the partially decrypted ciphertext $\left(\prod_{k\in I_A} e(g_1,g_2)^{\frac{\alpha_k}{z_j}}\right)^s$. Without the appointed user's secret key, the cloud cannot further decrypt the ciphertext. Therefore, data confidentiality against the curious-but-honest cloud service provider is also guaranteed.

Table 2 Comparison of storage overhead

Entity	Lewko's scheme	Our scheme
Data owner	$2\sum_{k\in I_A} n_k \lvert p\rvert$	$\left(N_A + 3 + \sum_{k\in I_A} n_k\right) \lvert p\rvert$
Authority	$2n_k\lvert p\rvert$	$n_k\lvert p\rvert$
Cloud	$(3l+1)\lvert p\rvert$	$\left(3N_A + 2l + 1 + \sum_{k\in I_A} n_{k,UID}\right) \lvert p\rvert$
User	$\sum_{k\in I_A} n_{k,UID}\cdot\lvert p\rvert$	Almost can be ignored

3) Forward and backward security

Our multi-authority CP-ABE access control scheme guarantees forward security and backward security of the outsourced data against any revoked and newly joining users respectively. Forward security means that the revoked user (whose attribute is revoked) cannot decrypt the new ciphertext that require the revoked attributes to decrypt. While backward security means that the newly joined user can also decrypt the previously published ciphertext that are encrypted with previous public keys if it has sufficient attributes.

In our scheme, when a user is revoked, his transformation keys will be deleted by the cloud server. Thus, the revoked user cannot decrypt the ciphertext with his transformation keys. Therefore, the forward security of the outsourced data is guaranteed.

On the other hand, when a new user joins to share the outsourced data, the ciphertext will be re-encrypted by the cloud so that he can also decrypt the cipthertext. Therefore, the backward security of the outsourced data can also be guaranteed.

5.2 Performance analysis

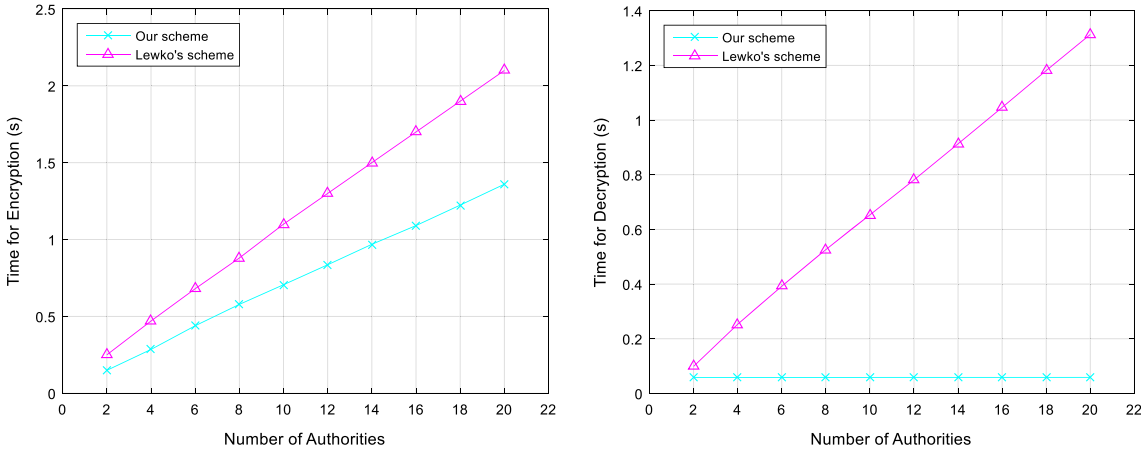
The performance analysis of our multi-authority CP-ABE access control scheme contains thefollowing available properties.

1) Flexibility analysis

We compare our scheme with previous multi-authority CP-ABE schemes in Table 1 with regard to the access structure type, the type of authority, the security against collusion attack, and the support of revocation.

Table 3 Comparison of communication cost for attribute revocation

Communication cost between	Ruj's scheme	Our scheme
Key Update	None	$n_{non,x}\cdot\lvert p\rvert$
Ciphertext Update	$(n_{c,x}\cdot n_{non,x}+1)\lvert p\rvert$	$\lvert p\rvert$



(a) Encryption (b) Decryption

Fig. 3 Comparison of computing cost with different number of authorities

From Table 1, we can conclude that our scheme is more flexible than other schemes. It is easy to find that the flexibility of Lewko's scheme [29] is similar with our scheme. Therefore we will further compare our scheme with it in terms of storage overhead.

2) Efficiency analysis

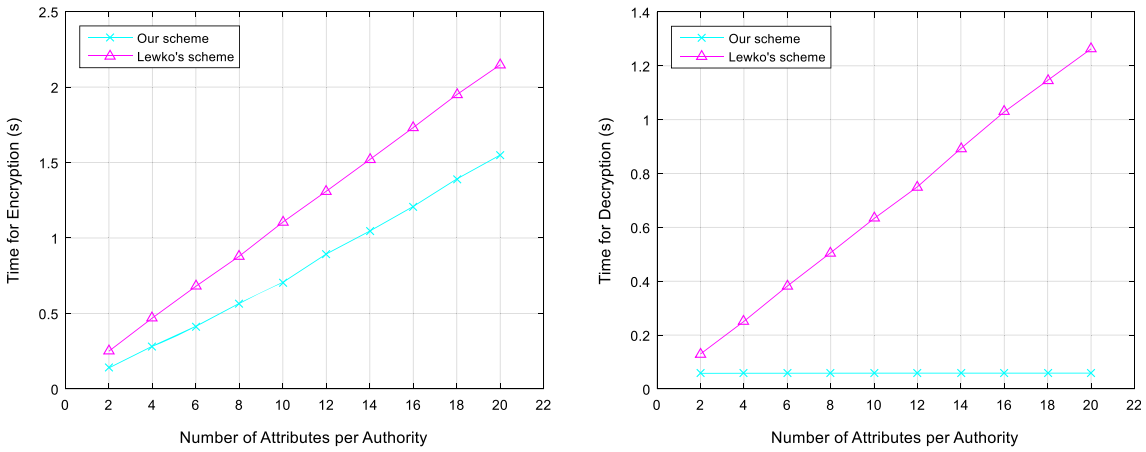
We present the efficiency analysis of our scheme in terms of the storage overhead, communication cost and computation cost.

A) Storage overhead

To compare our scheme with Lewko's scheme, here we modified our scheme from the asymmetric groups into symmetric groups. Let $|p|$ denote the element size in $\mathbb{G}$, $\mathbb{G}_T$ and $\mathbb{Z}_p$. Let N_A denote the total number of authorities in the system and I_A denote the set of all the authorities in the system and each authority manages $n_k (k \in I_A)$ attributes. For a user with UID, let $n_{k,UID}$ denote the number of attributes obtained from the authority with $AID_k (k \in I_A)$. Let l be the total number of attributes used for encryption.

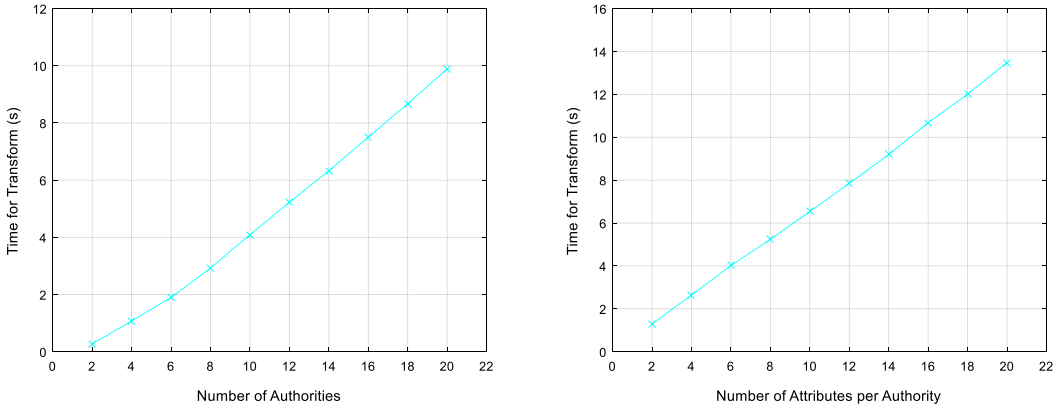
In our scheme, the storage overhead on the authority is just the version keys of each attributes. While in Lewko's scheme the authority's secret keys contribute to the storage overhead on it. In our scheme, the storage overhead of each data owner comes from all the public parameters. Since the storage overhead on the cloud is the size of ciphertext and the transformation keys of each user, the user's storage overhead is just the users' secret keys which can almost be ignored.

From Table 2, we can conclude that the storage overhead on the authority and users in our scheme is much less than the one in Lewko's scheme, especially for users. Note that if more authorities involved in the system, our scheme incurs much less storage overhead than Lewko's scheme.



(a) Encryption (b) Decryption

Fig. 4 Comparison of computing cost with different number of attributes per authority



(a) Different number of authoties (b) Different number of attributes per authority

Fig. 5 Computing cost of transformation

B) Communication cost

Let $n_{c,x}$ denote the number of ciphertext contains x and $n_{non,x}$ denote the number of non-revoked users hold$x$.We compare the communication cost of attribute revocation of our scheme with Ruj's scheme [30], as shown in Table 3. It is obvious that the communication cost of attribute revocation in Ruj's scheme is linear to the number of ciphertext which contains the revoked attributes. Therefore, if the number of ciphertext in cloud storage system is very large, the communication cost for attribute revocation in Ruj's scheme will be very heavy.

C) Computation cost

We implemented our scheme in Charm [31], a framework developed to facilitate the rapid prototyping of cryptographic schemes and protocols. It is based on the Python language which allows the programmer to write code similar to the theoretical implementations. However, the routines that implement the dominant group operations use the PBC library [32] (written natively in C) and the time overhead imposed by the use of Python is usually less than 1%. Charm also provides routines for applying and using LSSS schemes needed for Attribute-Based systems. For more information on Charm we refer the reader to [31, 33]. All our implementations are executed on an Intel® Pentium® CPU G630@270GHz with 4.00GB RAM running Ubuntu14.04 and Python2.7.

We compared the computing time incurred in encryption and decryption. The experiment conditions can be divided into two cases. In Fig. 3, the number of attributes per authority is set to 10. In Fig. 4, the number of authoritiesis fixed to 10. Fig. 3 (a) describes the comparison of encryption time with different number of authorities. Fig. 3 (b) shows the comparison of decryption time. It is obvious that our scheme requires less time for encryption and decryption than Lewko's scheme, especially for decryption. Since in the decryption phase, major computation overhead is delegated to the cloud, the computation left for users is only a little. Therefor decryption time for users is so little that can almost be ignored. From Fig. 3, we have similar results. Computing cost of transformation is shown in Fig. 5. On the whole, it can be concluded that our scheme's computation efficiency is much better than Lewko's scheme.

6 Conclusion

In this paper, we proposed a revocable data sharing scheme for vehicular fogs. We presented a new multi-authority CP-ABE scheme with efficient decryption to realize data access control in vehicular network system, and designed an efficient user and attribute revocation method for it. The analysis and the simulation results show that our scheme is secure and highly efficient.

Acknowledgements This work has been financially supported by the National Natural Science Foundation of China (No. 61303216, No. 61272457, No. U1401251, and No. 61373172), the National High Technology Research and Development Program of China (863 Program) (No. 2012AA013102), the Open Research Project of the State Key Laboratory of Industrial Control Technology, Zhejiang University, China (No. ICT170312), and National 111 Program of China B16037 and B08038.

References

1. Zhu H, Chang S, Lu L, Zhang W (2016) RUPS: fixing relative distances among urban vehicles with context-aware trajectories. Proceedings of IPDPS 2016:123–131
2. Zhu H, Chang S, Li M, Naik K, Shen S (2011) Exploiting temporal dependency for opportunistic forwarding in urban vehicular networks. Proceedings of INFOCOM 2011:2192–2200
3. Gerla M, Kleinrock L (2011) Vehicular networks and the future of the mobile internet. Comput Netw 55(2):457–469

4. Luan T, Shen X, Bai F (2013, 2013) Integrity-oriented content transmission in highway vehicular ad hoc networks. Proceedings of INFOCOM:2562–2570
5. Fernando N, Loke S, Rahayu W (2013) Mobile cloud computing: a survey. Futur Gener Comput Syst 29(1):84–106
6. Tuli A, Hasteer N, Sharma M, Bansal A (2013) Exploring challenges in mobile cloud computing: An overview. Proceedings of Next Generation Information Technology Summit 2013:496–501
7. Bonomi F (2011) Connected vehicles, the internet of things, and fog computing. VANET 2011:13–15
8. Bonomi F, Milito R, Zhu J, Addepalli S (2012, 2012) Fog computing and its role in the internet of things. Proceedings of MCC:13–16
9. Aazam M, Huh E (2014) Fog computing and smart gateway based communication for cloud of things. Proceedings of FiCloud 2014: 464–470
10. Aazam M, Huh E (2015) E-HAMC: leveraging fog computing for emergency alert service. Proceedings of PerCom 2015:518–523
11. Hong K, Lillethun D, Ramachandran U, Ottenwälder B, Koldehofe B (2013) Mobile fog: a programming model for large-scale applicationson the internet of things. Proceedings of MCC 2013:15–20
12. Ottenwalder B, Koldehofe B, Rothermel K, Ramachandran U (2013) Migcep: Operator migration for mobility driven distributed complexevent processing. Proceedings of DEBS 2013:183–194
13. Nishio T, Shinkuma R, Takahashi T, Mandayam N (2013) Service oriented heterogeneous resource sharing for optimizing service latency in mobile cloud. Proceedings of MobileCloud 2013:19–26
14. Dong M, Liu X, Qian Z, Liu A, Wang T (2015) QoE-ensured price competition model for emerging mobile networks. IEEE Wirel Commun 22(4):50–57
15. Wei K, Dong M, Ota K et al (2015) CAMF. Context-aware message forwarding in mobile social networks. IEEE Transactions on Parallel and Distributed Systems 26(8):2178–2187
16. Luo S, Dong M, Ota K et al (2015) A security assessment mechanism for software-defined networking-based mobile networks. Sensors 15(12):31843–31858
17. Ota K, Dong M, Chang S et al (2015) MMCD: cooperative downloading for highway VANETs. IEEE Trans Emerging Topics Comput 3(1):34–43
18. Ostrovsky R, Sahai A, Waters B (2007, 2007) Attribute-based encryption with non-monotonic access structures. Proceedings of CCS:195–203
19. Rafaeli S, Hutchison D (2003) A survey of key management for secure group communication. ACM Comput Surv 35(3):309–329
20. Boyen X, Waters B (2007, 2007) Full-domain subgroup hiding and constant-size group signatures. Proceedings of PKC: 1–15
21. Liang X, Lu R, Lin X, Shen X (2010) Ciphertext policy attribute-based encryption with efficient revocation. IEEE symposium on security and privacy. Pp 321-334
22. Xie X, Ma H, Li J, Chen X (2013) New ciphertext-policy attribute-based access control with efficient revocation. Proceedings of ICT 2013:373–382
23. Hur J, Noh D (2010) Attribute-based access control with efficient revocation in data outsourcing systems. IEEE Transactions on Paralleland Distributed Systems 22(7):1214–1221
24. Beimel A (1996) Secure schemes for secret sharing and key distribution. Technion-Israel Institute of Technology, Faculty of Computer Science. pp 22–28
25. Dong M, Zheng L, Ota K, Guo S, Guo M, Li L (2009) A trade-off approach to optimal resource allocation algorithm with cache technology in ubiquitous computing environment. Proceedings of CSE 2009. Pp 9-15
26. Chang S, Qi Y, Zhu H et al (2012) Footprint: detecting Sybil attacks in urban vehicular networks. IEEE Transactions on Parallel & Distributed Systems 23(6):1103–1114
27. Stojmenovic I, Sheng W (2014, 2014) The fog computing paradigm: scenariosand security issues. Proceedings of FedCSIS:1–8
28. Chase M (2007) Multi-authority attribute based encryption. Proceedings of theory of cryptography. Pp 515-534
29. Lewko A, Waters B (2011, 2011) Decentralizing attribute-based encryption. Proceedings of advances in cryptology–EUROCRYPT:568–588
30. Ruj S, Nayak A, Stojmenovic I (2011) DACC: distributed access control in clouds. Proceedings of TrustCom 2011:91–98
31. Akinyele J, Garman C, Miers I et al (2013) Charm: a framework for rapidly prototyping cryptosystems. J Cryptogr Eng 3(2):111–128
32. Lynn B. PBC Library: The pairing-based cryptography library. http://crypto.stanford.edu/pbc. Accessed 26 Jul 2016
33. Charm: A tool for rapid cryptographic prototyping. http://www.charm-crypto.com. Accessed 26 Jul 2016

Kai Fan received his BS, MS and PhD degrees from Xidian University, P. R. China, in 2002, 2005 and 2007, respectively, in Telecommunication Engineering, Cryptography and Telecommunication and Information System. He is working as an associate professor in State Key Laboratory of Integrated Service Networks at Xidian University. He has published over 40 papers in journals and conferences. He received 3 Chinese patents. He has managed 5 national research projects. His research interests include cloud computing security, IoT security and information security. The mailing address is 2 South Taibai Road, Xidian University, Xi'an 710071, China. The email address is kfan@mail.xidian.edu.cn.