LABAC: A Location-aware Attribute-based Access Control Scheme for Cloud Storage

Yingjie Xue^1, Jianan Hong^1, Wei Li^1, Kaiping Xue^{1,2}*, Peilin Hong^1
1.The Department of EEIS, University of Science and Technology of China, Hefei, Anhui 230027 China
2.State Key Laboratory of Information Security (Institute of Information Engineering),
Chinese Academy of Sciences, Beijing 100093, China.
*kpxue@ustc.edu.cn

***Abstract*—Data access control is a challenging issue in cloud storage. Ciphertext-Policy Attribute-based Encryption (CP-ABE) is a potential cryptographic technique to address the above issue, which is able to enforce data access control based on users' permanent characteristics. However, in some scenarios, access policies are associated with users' temporary conditions (such as access time and location) as well as their permanent ones. CP-ABE cannot deal with such situations commendably.**

In this paper, we focus on the scenario where users' access privilege is determined by their attributes, together with their locations. To cope with this data access control requirement, we propose a location-aware attribute-based access control mechanism (LABAC) for cloud. In LABAC, we uniquely integrate CP-ABE with location trapdoors to make up access policies. In this way, data owners can flexibly combine both users' attributes and locations to implement a fine-grained control of their data. A competitive advantage of LABAC is that it requires no any additional revocation mechanisms to revoke location-aware access privilege when user location changes. Security and performance analysis are presented which show the security and efficiency of LABAC for practical implementations.

***Index Terms*—Cloud Storage, Data Access Control, CP-ABE, Location.**

I. Introduction

Cloud storage, known as Storage as a Service (STaaS) [1], enables users to store their data at remote disks and access them at anytime from any place, and meanwhile significantly reduce their cost of capital expenditure on hardware maintenances. However, this novel data storage paradigm also brings challenging issues on data access control, which must be well resolved to protect data against unauthorized access. Traditional access control architectures usually assume the data owner and data servers are in the same trust domain, while this assumption no longer holds when data are outsourced to a remote cloud server, which resides outside the trust domain of the data owner. As a consequence, traditional access control solutions are not applicable, leaving the enforcement of data access control to be an open problem.

Several works and researches are focusing on finding solutions to the data access control in cloud storage model [2–5]. Among those works, schemes based on Ciphertext-Policy Attribute-Based Encryption (CP-ABE) primitive [2–4] have attracted wide attention for its capacity to grant data owners fine-grained control of their data. Intuitively, CP-ABE is often suggested as a realization to implement role-based access control, where the user's attributes correspond to the long-term roles. Whereas, in some scenarios, a user's access privilege is not only determined by his/her inherent roles but also depends on his/her environmental conditions such as location (referred to as *spatial dimension*). For example, the widespread location-based service (LBS) [6] requires users to show up at specialized location to access sensitive data. The requirement of combing a user's location information and role into access policies in traditional models has been satisfied in previous works, known as GEO-RBAC [7], LRBAC [8], GSTRBAC [9]. However, when it comes to the scenario where data are stored in cloud, little work has strived to solve that problem, except the literatures [10, 11] and [12]. Although in [10], the authors gave a solution named LoTAC for cloud storage model, their proposal lacks fine-grained control of data consumers. Shao et al. [11] designed a fine-grained location-based service framework in cloud, called FINE, but their scheme cannot combine spatial dimension and attribute dimension into access policies flexibly. The work in [11] was constructed on Key-policy Attribute-based Encryption (KP-ABE) prototype, in which data owners cannot directly designate access policies to secure their data.

A trivial solution to combine a user's role and location into access policies is to regard the spatial dimension as an exceptional domain of attributes sets, thus the location information can be seamlessly embedded into CP-ABE access policies as a normal attribute. However, the main difference between a user's location information and his/her attributes is that the attributes are classified by his/her identity which will hold for a long period while the location information is a temporary condition, which is frequently changing over time. If location information is handled as an attribute of the user, his/her attribute set will change constantly with his/her frequent movement. To ensure data confidentiality, the key distributor should always monitor the changes of users attribute set thus to revoke the old-version attributes and assign new attributes to them in time, which is obviously impractical in real scenarios. Even worse, existing proxy re-encryption schemes for CP-ABE to deal with attribute revocation and update are quite inefficient [13, 14], which could paralyze the access control system due to users' frequently changing locations.

In this paper, we address the location-aware role-based access control in cloud storage environment and pro-

978-1-5090-1328-9/16/$31.00 ©2016 IEEE

pose a location-aware attribute-based access control scheme (LABAC) to cope with the problem. We start with a location-aware framework and then give our cryptographic algorithms. Our approach is also based on CP-ABE primitive, to inherit the property of fine granularity for a role-based access control, but the location information is handled in a different manner. In LABAC, sensitive data are encrypted by access policies before outsourcing to the cloud. The location information is embedded as trapdoors inside access policies and attributes are handled in the same way as what CP-ABE does. Location servers are deployed to release trapdoors for users. To decrypt a ciphertext, a user (data consumer) should possess proper attribute set to satisfy the attributes inside the access policy as well as get a token from the corresponding location server to get rid of the trapdoor. The trapdoor is independent of users' attribute sets, which as a result users do not possess private keys associated to their temporary locations. Therefore, the trapdoor approach relieves the burden of revoking and reassigning users when their locations change. Furthermore, LABAC allows the location trapdoors to be set arbitrarily in the access policy along with attributes, and one ciphertext can be associated with multiple trapdoors in terms of different locations. In this way, the location information can be incorporated with users' attribute sets in a flexible manner, and data owners can determine users possessing different attribute sets to access the same datum from different locations, as shown in Figure 1. The main contributions of this work can be summarized as follows.

1) We address the location-aware attribute-based access control problem in cloud and give a general framework. In this framework, location servers are deployed to help handle location-related access control by cryptography, and the attribute-related access control is conducted by fundamental CP-ABE schemes. To the best of our knowledge, it is the first time to incorporate a user's location and attributes in access policies simultaneously.
2) A location-aware access control scheme is proposed, without losing the fine granularity of CP-ABE prototype. Specially, the implementation of location trapdoor in fundamental CP-ABE relieves the burdensome revocation when a user's location changes.
3) Security and performance analysis are provided, which indicates that our proposed scheme is secure, flexible and efficient.

The rest of this paper is organized as follows. In Section II, technical preliminaries are presented. Following the definitions of system model and security assumptions in Section III, we detailed introduce our proposed location-aware attribute-based access control (LABAC) scheme in Section IV. We analyze LABAC in terms of security and performance in Section V. Finally, the conclusion is given in Section VI.

II. Preliminaries and Definitions

A. Access Policy Structures

An access policy structure $\mathcal{T}$ consists of several nodes of a policy tree, and some location-related trapdoors TD. We give an example of access policy structure in Fig 1. A leaf node represents a certain attribute (In Fig. 1, $A_0, \cdots, A_3$ are the relevant attributes), and each non-leaf node represents a threshold gate ("AND", "OR", or other threshold gates). Each non-leaf node x has two logic value n_x and k_x, where n_x is the number of its child node, and k_x is the threshold. Especially, $k_x = 1$ if x is an OR gate, or $k_x = n_x$ if x is an AND gate. We refer readers to [4] for further details about the access policy structure $\mathcal{T}$.

In the structure $\mathcal{T}$, TD_y^k is related to location Loc_k and it is appended to a node y. From the perspective of algorithm, TD can be appended to arbitrary nodes of the policy tree. For instance, in Fig. 1, $TD_{y_2}^2$ is appended to a leaf node in order to restrict the attribute A_1 along with location Loc_2, while $TD_{y_1}^1$ is set upon a non-leaf node to restrict a sub-policy "$A_2 \wedge A_3$" along with location Loc_1.

A user's private key SK is associated with a set of attributes. In Fig. 1, U_1's attributes are A_0, A_2 and A_3, while U_2's attributes are A_0 and A_1.

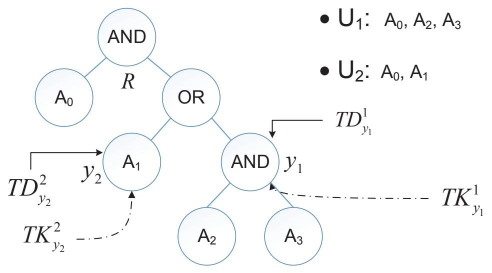


Fig. 1. An Example of Access Policy Structure

B. Trapdoors and Tokens

A location trapdoor (TD) is embedded in a policy tree, such that the corresponding user's access permission is restricted by the status of TD. In this paper, we define two statuses for the location trapdoor: *exposed* or *unexposed*.

Unexposed. A trapdoor (TD) is *unexposed* if the intended users cannot access the corresponding secret through the trapdoor with their private keys.

Exposed. A trapdoor is *exposed* if the intended users can get the corresponding secret through this trapdoor.

Tokens (TK) are generated by location servers to help users transform a trapdoor from *unexposed* status to *exposed* status.

III. System Model and Security Assumptions

A. System Model

Our system consists of six entities: ***cloud server***, an ***attribute authority***, multiple ***location servers***, each with some ***sensors***, many ***data owners*** and ***data consumers***. Fig. 2 shows the organization of these entities.

- ***The data owner (Owner)*** defines access policies and encrypt his/her data under the policies before uploading them to the cloud server.

- ***The attribute authority (AA)*** is responsible to set up the system parameter and distribute private keys to the users according to their attribute sets.
- ***The location servers (LS)*** are located in some particular areas where location information is required in access privilege assignments. It can execute some computation operations for users such as decrypting a trapdoor. The location server can affirm a user's location with the help of ***sensors***.
- ***Sensors*** are deployed in the areas around the location to help location servers authenticate a user's location. They can be RFID sensors, base stations or something else, based on the positioning technique the system uses. We assume the authentication results from sensors can be securely transmitted to location servers.
- ***The data consumer (User)*** can download any ciphertext of his/her interest from the cloud server and tries to decrypt the ciphertext. He/she is equipped with a private key according to his/her attribute set.
- ***The cloud server (CS)*** stores owners' uploaded data and provides access services for data consumers.

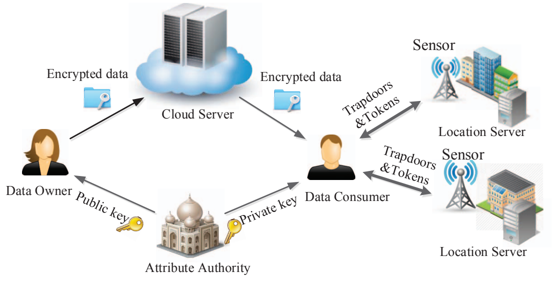


Fig. 2. System Model of LABAC Scheme

B. Security Assumptions

The cloud server is assumed honest-but-curious, which correctly executes the tasks assigned to it for profits but also tries to find out as much information of data contents as possible. On the contrary, ***AA*** is assumed to be fully trusted which will not collude with any entity to harvest data content even if it is highly beneficial. ***LS*** is also assumed fully trusted, which is managed by local administrators who aim to protect data security. We can trust the location servers that they will not collude with users to gain unauthorized rights. That security assumption also fits in ***Sensors***.

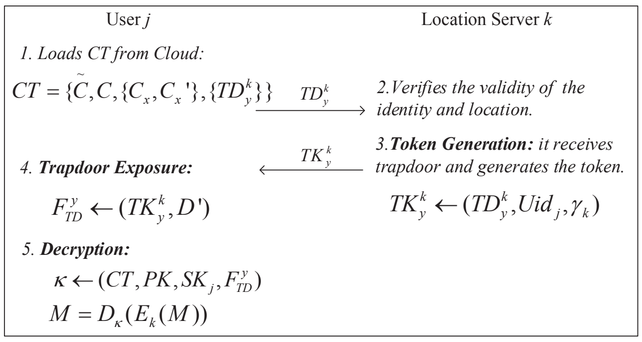
User is assumed dishonest. He/she wants to decrypt data even he/she is not authorized with any potential approaches, such as colluding with other users. We assume ***Owner*** to be honest and will not do harm to their data confidentiality.

IV. Location-aware Attribute-based Access Control Scheme

In this section, we first give an overview of our proposal, then we give detailed descriptions of LABAC scheme.

A. Overview

Our main motivation to design LABAC is to inherit fine granularity from CP-ABE and avoid the burdensome overhead of revocation caused by users' frequently changing locations. We give our solution by creatively setting trapdoors inside the access policy, as shown in Fig. 1. The secret for a successful decryption is distributed not only inside attributes (leaf nodes in access policy) but also inside trapdoors which are related to locations. As an example in Fig. 1, the trapdoor $TD_{y_2}^2$ is set on a node y_2. For U_2, to decrypt the ciphertext according to his/her attribute set, he/she has to interact with location server 2 at location Loc_2 to get a token $TK_{y_2}^2$, with which he/she can transform the trapdoor from *unexposed* to *exposed* status, thus he/she can obtain the intended secret through the trapdoor for a successful decryption. If he/she cannot get the relevant token, he/she cannot decrypt the ciphertext correctly. Fig. 3 shows the main procedure to decrypt a ciphertext. A notable property of token and trapdoor is that they are unique for different ciphertexts. A token of ciphertext A gives no advantage for a user to decrypt ciphertext B ($B \neq A$). Therefore, he/she has no access privilege of other ciphertexts when he/she leaves, without additional revocation mechanisms to revoke him/her.



User *j*

Location Server *k*

1. Loads CT from Cloud:

$CT = \{\tilde{C}, C, \{C_x, C_x'\}, \{TD_y^k\}\} \xrightarrow{TD_y^k}$

2.Verifies the validity of the identity and location.

*3.**Token Generation:** it receives trapdoor and generates the token.*

$\xleftarrow{TK_y^k}$

*4. **Trapdoor Exposure:***

$TK_y^k \leftarrow (TD_y^k, Uid_j, \gamma_k)$

$F_{TD}^y \leftarrow (TK_y^k, D')$

*5. **Decryption:***

$\kappa \leftarrow (CT, PK, SK_j, F_{TD}^y)$

$M = D_\kappa(E_k(M))$

Fig. 3. Main Procedure of Decrypting a Ciphertext

B. Details of LABAC Scheme

*1) **System Initialization**:* The attribute authority (AA) generates $I = [p, \mathbb{G}_1, \mathbb{G}_2, g, e, H_0, H_1, H_2]$, where $e: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is a non-degenerate bilinear map, $\mathbb{G}_1$ and $\mathbb{G}_2$ are cyclic multiplicative groups of order p, g is a generator of $\mathbb{G}_1$, $H_0, H_1: \{0,1\}^* \rightarrow \mathbb{G}_1^*$, $H_2: \mathbb{G}_2 \rightarrow \mathbb{Z}_p^*$. AA also randomly chooses $\alpha, \beta \in \mathbb{Z}_p$, and the public key PK is set as:

$$PK = (I, h = g^\beta, e(g,g)^\alpha)$$

The master key MK of AA is:

$$MK = (\beta, g^\alpha)$$

Besides, assume there are L locations, then every location server k ($k \in 1, 2, \ldots, L$) randomly chooses a location secret key γ_k as its secret key. The public key PK_k of each location server k is:

$$PK_k = \{\mathbb{F}_{Loc_k}, l_k = g^{\gamma_k}\}$$

$\mathbb{F}_{Loc_k}$ is the format description of that location. The master secret key of location Loc_k is:

$$MK_k = \gamma_k$$

In our system, each user U_j is assigned a unique ID Uid_j.

*2) **Encryption**:* The data owner first encrypts his/her data M with a symmetric key $\kappa \in \mathbb{G}_2$. Then he/she encrypts κ under the access policy he/she has defined and finally uploads the whole encrypted data. Owner generates the ciphertext CT by the following algorithm.

The algorithm chooses a polynomial q_x for each node x in the tree $\mathcal{T}$. These polynomials are chosen in a top-down manner, starting from the root node R. For each node x in the tree, set the degree d_x of the polynomial q_x to be one less than the threshold value k_x of that node, that is $d_x = k_x - 1$. And for any node x, it has two values q_x^0, q_x^1. q_x^0 is a parameter inherited from its parent node (or dealt with root node, if x is a root node), and q_x^1 is the one shared with its child node (or dealt with the relevant attribute, if x is a leaf node).

Starting with the root node R, the algorithm chooses a random $s \in \mathbb{Z}_p$ and sets $q_R^0 = s$.

For each node x with assigned value q_x^0, if it is related to location Loc_k, $s_x^k \in \mathbb{Z}_p$ is randomly chosen for the trapdoor. With random $\upsilon_k \in \mathbb{Z}_p$, the trapdoor is set as:

$$TD_x^k = (A_x = g^{\upsilon_k}, B_x = s_x^k + H_2(e(H_1(\mathbb{F}_{Loc_k}), l_k)^{\upsilon_k}))$$

Then the other value q_x^1 is computed as:

$$\begin{cases} q_x^1 = q_x^0 - s_x^k & x \text{ is related to location } Loc_k \\ q_x^1 = q_x^0 & \text{otherwise} \end{cases} \tag{1}$$

For any non-leaf node x, the polynomial is randomly chosen with 2 restrictions: 1) $q_x(0) = q_x^1$; 2) $d_x = k_x - 1$. For any of its child node z, set $q_z^0 = q_x(index(z))$.

Let $\mathcal{Y}$ be the set of leaf nodes in $\mathcal{T}$. The ciphertext is then constructed as:

$$\begin{aligned} CT = \{&\widetilde{C} = \kappa e(g,g)^{\alpha s}, C = h^s, \\ &\forall x \in \mathcal{Y},\ C_x = g^{q_x^1}, C'_x = H_1(att(x))^{q_x^1}; \\ &\forall TD_y^k \in \mathcal{T}, k \in \{1,2,\ldots,L\}, TD_y^k = (A_y, B_y)\} \end{aligned}$$

We assume the access policy $\mathcal{T}$ is attached to CT implicitly. The format of data on the cloud server is shown in Fig. 4.

Description	$E_\kappa(M)$	$CT = \{\widetilde{C}, C, \{C_x, C'_x\}, \{TD_y^k\}\}$

Fig. 4. Format of Owner's Data On the Cloud Server

*3) **Key Generation**:* For user U_j with attribute set S_j, AA randomly chooses $u_j \in \mathbb{Z}_p$. Then, for every attribute $Att_i \in S_j$, it assigns a random value $r_i \in \mathbb{Z}_p$. The private key AA generates for user U_j is:

$$\begin{aligned} &SK_j = \{(D = g^{(\alpha+u_j)/\beta} H_0(Uid_j)^{u_j/\beta}, D' = g^{u_j} \\ &\forall Att_i \in S_j : D_i = (gH_0(Uid_j))^{u_j} H_1(Att_i)^{r_i}, D'_i = g^{r_i})\} \end{aligned}$$

*4) **Token Generation**:* In this phase, a location server outputs a token for the user to expose the trapdoor. The location server first confirms the user's identity and his/her location, with the help of sensors around the location. After confirmation, the location server k receives the trapdoor TD_y^k from user U_j and computes:

$$TD_y'^k = B_y - H_2(e(H_1(\mathbb{F}_{Loc_k})^{\gamma_k}, A_y))$$

If the trapdoor is decrypted correctly, then $TD'_y = s_y^k$. The location server generates the relevant token and sends it to the user. The token TK_y^k is computed as:

$$TK_y^k = (gH_0(Uid_j))^{s_y^k}$$

*5) **Trapdoor Exposure**:* On receiving token TK_y^k from location server k, the user exposes the trapdoor TD_y^k by computing F_{TD}^y as:

$$\begin{aligned} F_{TD}^y =& e(TK_y^k, D') \\ =& e((gH_0(Uid_j))^{s_y^k}, g^{u_j}) \\ =& e(gH_0(Uid_j), g)^{u_j s_y^k} \end{aligned}$$

*6) **Decryption**:* The user U_j decrypts the ciphertext with his/her private key SK_j. We first define a recursive algorithm $Decrypt(CT, SK_j, x)$ that takes as input a ciphertext CT, a private key SK_j, and a node x from $\mathcal{T}$.

If the node x is a leaf node, then we let $Att_i = att(x)$ and define as follows.

If $Att_i \in S_j$, then

$$\begin{aligned} F_x =& DecryptNode(CT, SK_j, x) \\ =& \frac{e(D_i, C_x)}{e(D'_i, C'_x)} \\ =& \frac{e((gH_0(Uid_j))^{u_j} H_1(Att_i)^{r_i}, g^{q_x^1})}{e(g^{r_i}, H_1(Att_i)^{q_x^1})} \\ =& e(gH_0(Uid_j), g)^{u_j q_x^1} \end{aligned}$$

If $Att_i \notin S_j$, then we define $DecryptNode(CT, SK, x) = \perp$.

Faced with $\forall z \in \mathcal{T}$ (including leaf nodes and non-leaf nodes) which is linked with a trapdoor $TD_z^k \in \mathcal{T}$ and it is *exposed*, set F'_z as:

$$\begin{aligned} F'_z &= F_z \cdot F_{TD}^z \\ &= e(gH_0(Uid_j), g)^{u_j(q_z^1 + s_z^k)} \\ &= e(gH_0(Uid_j), g)^{u_j q_z^0} \end{aligned}$$

If the trapdoor is *unexposed*, the decryption is blocked, and the algorithm returns $\perp$. For $\forall z \in \mathcal{T}$ (including leaf nodes and non-leaf nodes) that is not set with any trapdoors:

$$\begin{aligned} F'_z &= F_z \\ &= e(gH_0(Uid_j), g)^{u_j q_z^1} \\ &= e(gH_0(Uid_j), g)^{u_j q_z^0} \end{aligned}$$

Until now, all $F'_z = e(gH_0(Uid_j), g)^{u_j q_z^0}$. We now consider the recursive case when x is a non-leaf node. For all node z that are children of x, let S_x be an arbitrary k_x-sized set of child nodes z such that $F'_z \neq \perp$. If no such set exists then the node is not satisfied and the function returns $\perp$.

Then for each non-leaf node x:

$$\begin{aligned} F_x =& \prod_{z \in S_x} F_z'^{\Delta_{i,S'_x}} \\ =& \prod_{z \in S_x} e(gH_0(Uid_j), g)^{u_j q_z^0 \prod_{y \in S_x, y \neq z} \frac{index(y)}{index(y) - index(z)}} \\ =& e(gH_0(Uid_j), g)^{u_j q_x^1} \end{aligned}$$

For root node R, denote $F_R = (e(gH_0(Uid_j), g)^{u_j q_R^1})$. Set F'_R as:

$$\begin{cases} F'_R = F_R \cdot F_{TD}^R & TD_R^k \text{ is set upon } R \\ F'_R = F_R & \text{otherwise} \end{cases}$$

If $\mathcal{T}$ is satisfied by S_j and no *unexposed* trapdoors block the decryption, $F'_R = e(gH_0(Uid_j), g)^{u_j s}$. Then the algorithm computes:

$$\kappa' = \frac{\tilde{C}}{e(C, D)/F'_R} = \kappa$$

With the symmetric key κ, the user can obtain the plaintext M.

V. Security and Performance Analysis

A. Security Analysis

Security properties of LABAC can be summarized as follows.

1) Data Confidentiality: We can classify adversaries against LABAC into two categories: 1) adversaries with unsatisfied attribute set; 2) the ones that are not located in the access area. If LABAC is secure against the above two categories, then it can resist any individual attack, including the one that neither has satisfied attribute set, nor in the access location. The following paragraphs discuss these features.

LABAC makes necessary modifications towards CP-ABE, so that location information can be embedded into access policies. These modifications have not change the structure of CP-ABE algorithm, so that it holds the same property of data confidentiality against the first category of adversaries, as Waters' scheme [4].

The set and exposure of trapdoors implements *Identity-based Encryption (IBE)* scheme [15]. Thus, the security of token can be proved secure in random oracle model. Moreover, without a correct token, the reconstructed F'_R will be a random element of $\mathbb{G}_2$, which makes the adversary unable to compromise LABAC, who belongs to the second category of adversaries.

Towards the one with neither satisfied attribute set nor access location, we can either assign him/her additional attribute-related private keys or location-related tokens, so that he/she becomes either kind of adversaries. Note that the assigned information has not reduce his/her advantage. From the above analysis, we can conclude that LABAC is confidential against any individual adversary.

2) Security Against Collusion Attack: Users with different attribute sets may collude to combine a new private key that is associated with a larger attribute set to decrypt unauthorized ciphertexts. Similar to other collusion-resistant schemes, LABAC is collusion resistant because there is an unknown random value u_j for every different U_j in private keys (for example, for U_j, $D_i = (gH_0(Uid_j))^{u_j} \cdot H_1(Att_i)^{r_i}$). Therefore, private keys from U_i of random value u_i cannot combine with private keys from U_j of u_j to make a new private key that has larger privileges.

What is worth noting is that there may be another kind of collusion of users, that is, U_j may let U_i $(i \neq j)$ to get his/her tokens, so that U_j can access unauthorized data, which can only be accessed by U_j if himself/herself is in U_i's current location. We resolve the potential threat by the authentication of users at certain location and making the tokens related to the user who is right-now-authenticated. Thus, U_i who is showing up at some particular position Loc_k will get token as $(gH_0(Uid_i))^{s_y^k}$, which cannot be used by U_j who has $F_y = e(gH_0(Uid_j), g)^{u_j q_y^1}$. In this way, such collusion attack can be well resisted.

3) Non-revocation Security: Unlike private keys which grant users access privileges associated with their attribute sets for a long time, one token only give users capabilities to decrypt an exclusive ciphertext. That is to say, even if the user can decrypt ciphertext A at certain location Loc_i with token $TK_{y_1}^i$ for trapdoor $TD_{y_1}^i$ of A, when she/she leaves from location Loc_i, the token $TK_{y_1}^i$ can not be used to expose any other trapdoor $TD_y^i \in \mathcal{T}$ of ciphertext $B (B \neq A)$. Thus, his/her access privilege related to location Loc_i is revoked automatically when he/she leaves.

4) Security Against Location Server Compromise: In LABAC, the location server k is only assigned the ability to decrypt trapdoors associated with location Loc_k and it has no additional privilege to break the security of CP-ABE schemes. If some location server i is compromised, it only influences the security of ciphertexts whose access policies are related to that location, while other locations and attribute-related access control are not affected.

B. Performance Analysis

1) Functional Properties: From the perspective of functionality, in Table I, we make comparisons among our proposed framework and Waters' CP-ABE [4], LoTAC [10] and FINE [12], which address the similar problem in cloud storage. Notably, LABAC allows trapdoors to be set upon any node x in access tree, thus location information and attributes can be combined flexibly. This flexibility can also be achieved by Waters' CP-ABE [4] if we consider location as a normal attribute. However, additional revocation is needed to revoke users' attributes which associates with changing locations.

TABLE I
Functional properties

Schemes	CP-ABE	LoTAC	FINE	LABAC
Fine granularity	$\surd$	$\times$	$\surd$	$\surd$
Location-awareness	$\times$	$\surd$	$\surd$	$\surd$
Flexible combination	$\surd$	$\times$	$\times$	$\surd$
Need of additional revocation	Yes	No	No	No

2) Location-related computation overhead: From this subsection, we numerically evaluates the performance of LABAC versus CP-ABE schemes, where the location information is handled as a normal attribute. For one ciphertext associated with l locations, we evaluate the overhead caused by the enforcement of location information in access policies. The main operations include multiplication T_m and exponentiation T_e, and pairing T_p. $T_m^{\mathbb{G}_0}$, $T_m^{\mathbb{G}_1}$ and $T_m^{\mathbb{G}_2}$ represent the computation cost of multiplication on $\mathbb{G}_0$, $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively.

$T_p^{\mathbb{G}_0}, T_p^{\mathbb{G}_1}, T_e^{\mathbb{G}_0}, T_e^{\mathbb{G}_1}, T_e^{\mathbb{G}_2}$ are similar notations for operations on the three groups. The comparison results versus Waters' scheme [4] are shown in Table II. As we can see, LABAC does not increase much overhead to owners and it even reduces decryption cost for users.

TABLE II
LOCATION-RELATED COMPUTATION OVERHEAD

Entity	[4]	LABAC
$Owner$	$2lT_e^{\mathbb{G}_0}$	$(T_e^{\mathbb{G}_1} + T_e^{\mathbb{G}_2} + T_p^{\mathbb{G}_1})l$
$User$	$2T_p^{\mathbb{G}_0} + T_e^{\mathbb{G}_0} + 2T_m^{\mathbb{G}_0}$	$T_m^{\mathbb{G}_2} + T_p^{\mathbb{G}_1}$

3) Computation overhead with location-changing users: To illustrate the efficiency of LABAC, we evaluate the computation overhead caused by location-changing users versus literature [14], which is a CP-ABE scheme with revocation mechanisms. Assume there are L location servers, and for each location, there is an average of N related users and W ciphertexts related to it. A user is equipped with d ciphertexts to decrypt at one location. Table III shows the revocation overhead of [14] to revoke a user and computation overhead of location servers to generate tokens. As shown in Table III, we reduce the cost for authority and cloud server but add overhead for location servers. But it is a reasonable tradeoff to protect data confidentiality, since LABAC allows access privilege to be revoked immediately and automatically when the user leaves, without other entities involved in revocation.

TABLE III
COMPUTATION OVERHEAD CAUSED BY LOCATION-CHANGING USERS

Entity	[14]	LABAC
AA	$2NT_e^{\mathbb{G}} + 3NT_m^{\mathbb{G}}$	0
CS	$(2T_e^{\mathbb{G}} + T_m^{\mathbb{G}})W$	0
LS	0	$(2T_e^{\mathbb{G}_1} + T_p^{\mathbb{G}_1} + T_m^{\mathbb{G}_1})d/L$

VI. CONCLUSION

This paper proposes a location-aware attribute-based access control scheme for cloud storage, in which the location information is flexibly set as trapdoors inside fundamental access policies of CP-ABE, and trapdoors are released with the help of location servers. The trapdoor approach makes that the change of users' locations will not cause revocation of users' attributes. Our analysis shows that the above approach is effective and our proposed LABAC brings little overhead to data consumers, attribute authorities and the cloud.

From the perspective of security, LABAC shows another advantage that compromise of one location server will only influence the data associated with this location, while other data remain confidential. In our future work, we will focus on the restoration mechanisms in case a certain location server is compromised, including re-allocation of updated location server and secure re-encryption technique for relevant data.

ACKNOWLEDGMENT

This work is supported by the National Natural Science Foundation of China under Grant No. 61379129, Youth Innovation Promotion Association CAS, and the Fundamental Research Funds for the Central Universities.

REFERENCES

[1] J. Wu, L. Ping, X. Ge, Y. Wang, and J. Fu, "Cloud storage as the infrastructure of cloud computing," in *Proceedings of the 2010 IEEE International Conference on Intelligent Computing and Cognitive Informatics*. IEEE, 2010, pp. 380–383.

[2] S. J. De and S. Ruj, "Decentralized Access Control on Data in the Cloud with Fast Encryption and Outsourced Decryption," in *Proceedings of the 2015 IEEE Global Communications Conference*. IEEE, 2015, pp. 1–6.

[3] J. Shao, R. Lu, and X. Lin, "Fine-grained data sharing in cloud computing for mobile devices," in *Proceedings of the 2015 IEEE International Conference on Computer Communications*. IEEE, 2015, pp. 2677–2685.

[4] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proceedings of the 2007 IEEE Symposium on Security and Privacy*. IEEE, 2007, pp. 321–334.

[5] E. Goh, H. Shacham, N. Modadugu, and D. Boneh, "SiRiUS: Securing Remote Untrusted Storage." in *Proceedings of the 2003 Network and Distributed System Security Symposium*, vol. 3, 2003, pp. 131–145.

[6] M. Li, H. Zhu, Z. Gao, S. Chen, L. Yu, S. Hu, and K. Ren, "All your location are belong to us: Breaking mobile social networks for automated user location tracking," in *Proceedings of the 15th ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2014, pp. 43–52.

[7] E. Bertino, B. Catania, M. L. Damiani, and P. Perlasca, "GEO-RBAC: a spatially aware RBAC," in *Proceedings of the 10th ACM symposium on Access control models and technologies*. ACM, 2005, pp. 29–37.

[8] I. Ray, M. Kumar, and L. Yu, "LRBAC: a location-aware role-based access control model," in *Information Systems Security*. Springer, 2006, pp. 147–161.

[9] R. Abdunabi, M. Al-Lail, I. Ray, and R. B. France, "Specification, validation, and enforcement of a generalized spatio-temporal role-based access control model," *Systems Journal, IEEE*, vol. 7, no. 3, pp. 501–515, 2013.

[10] E. Androulaki, C. Soriente, L. Malisa, and S. Capkun, "Enforcing Location and Time-Based Access Control on Cloud-Stored Data," in *Proceedings of the 34th IEEE International Conference on Distributed Computing Systems*. IEEE, 2014, pp. 637–648.

[11] Y. Zhu, D. Ma, D. Huang, and C. Hu, "Enabling secure location-based services in mobile cloud computing," in *Proceedings of the 2nd ACM SIGCOMM workshop on Mobile cloud computing*. ACM, 2013, pp. 27–32.

[12] J. Shao, R. Lu, and X. Lin, "FINE: A fine-grained privacy-preserving location-based service framework for mobile devices," in *Proceedings of the 33rd IEEE International Conference on Computer Communications*. IEEE, 2014, pp. 244–252.

[13] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security*. ACM, 2010, pp. 261–270.

[14] K. Yang, X. Jia, and K. Ren, "Attribute-based fine-grained access control with efficient revocation in cloud storage systems," in *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security*. ACM, 2013, pp. 523–528.

[15] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Proceedings of the 21st Annual International Cryptology Conference*. Springer, 2001, pp. 213–229.