International Journal of Grid and Distributed Computing
Vol.6, No.6 (2013), pp.1-12
http://dx.doi.org/10.14257/ijgdc.2013.6.6.01

Context-Aware Access Control Model for Cloud Computing

Zhenji Zhou[1], Lifa Wu[2] and Zheng Hong[3]

Institute of Command Information System, PLA University of science and technology Nanjing, Jiangsu, China

[1]zhou_zhenji@163.com, [2]wulifa@vip.163.com, [3]hongzhengjs@139.com

Abstract

In view of malicious insider attacks on cloud computing environments, a new Context-Aware Access Control Model for cloud computing (CAACM) was presented. According to the characteristic of cloud computing, we take spatial state, temporal state and platform trust level as context. The model establishes mechanisms of authorization from cloud management role to objects, which enables dynamic activation of role permission by associating cloud management role with context. It also achieves fine-grained access control on cloud objects by supervising the permission of management role in full life cycle. Moreover, it introduces the concept of exclusive managerial role, which extends access control from static protection on resources to dynamic authorization on managerial roles. Further, it describes the approach of role permission activation systematically. CAACM formally proves to be safe and it lays the groundwork for the deployment of CAACM in cloud computing systems.

Keywords: *Cloud Computing, Access Control, Malicious Insider, Context Awareness*

1. Introduction

As the extension of multiple computing modes such as parallel computing, distributed computing and utility computing, cloud computing provides convenient and low-cost computing services. However, it also brings new security challenges [1]. The users of computing services should concern not only the cost of data storage but also the security and privacy of managed data. Malicious insider, one of seven threats of cloud computing has become the most serious problem and caused wide public concern in recent years [2].

In existing platforms of cloud computing, traditional distributed access control models are still used [3]. However, there is a notable difference between the mode of cloud computing and traditional data storage. In cloud computing, both the backup and migration of managed data are performed by managers, and the user can access to computing services only using the provided interfaces. Because of this mode, malicious insiders can steal confidential data easily through privileged operations [4].

Access control, the selective restriction of access to data or other resource, is always a major problem in computer security. Currently, the access control models fall into three types: MAC (Mandatory Access Control), DAC (Discretionary Access Control) and RBAC (Role-Based Access Control). In comparison, RBAC has emerged as a key solution for the threat of malicious insiders, since it simplifies authority management by separating users with permissions logically [5].

For large systems with complex relations between a large number of roles, Sandhu *et al.*, proposed a model named ARBAC97 [6], which extended RBAC model with

ISSN: 2005-4262 IJGDC

manage role and the corresponding licensing strategy. ARBAC97 improves the manageability of systems, but it doesn't constrain the permissions of management roles. In API level of cloud computing, Sirisha *et al.*, proposed a two-phase access control mechanism based on RBAC model [7]. However, it is incompatible with the openness of cloud computing since only accesses from white-list users are allowed. Li *et al.*, proposed model S-RBAC [8] to perform access control on SaaS mode, but they didn't consider the temporal constraints on permission license. Based on behavior definition [9], Lin *et al.*, proposed access control model CCACSM [10] for cloud computing. CCACSM combines BLP model with Biba model to achieve high confidentiality and integrity. Unfortunately, it doesn't take Malicious Insider into account as well.

Currently, most of existing access control models protect resources from system perspective. Once a subject gains access to an object, it will be able to use the permission in its full-life cycle. In this case, the models are not able to dynamically adjust distributed permissions according to environment information (*e.g.*, time, location and platform). It's obvious that such management style might very well lead to losses as it offends the principle of least privilege. In recent years, more and more researchers have realized the importance of context in authorization. Many context-sensitive access control model have been proposed, such as GTRBAC[11], X-GRBAC [12], GRBAC [13], GEO-RBAC [14] and SC-RBAC [15]. For different application scenarios, these models enhance the description power of authority policies by extending roles with time and location, which improves the security and flexibility of cloud computing significantly. However, the managers of cloud computing should manage system resources through specific platform (software, hardware and so on). If the platform is tampered maliciously, the security of cloud computing will still not be guaranteed even the temporal and spatial factors of managers are constrained [16].

To perform access control on malicious insiders of cloud computing, we introduce the notion of context by synthesizing multiple aspects of security information, such as temporal state, spatial state and trust level of platform. On this basis, we proposed CAACM, a novel Context-Aware Access Control Model for cloud computing. The advantages of CAACM are the following: 1) it dynamically constrains the permissions of manager roles in full-life cycle by associating roles with context, and achieves fine-grained access control from managers to objects. 2) It follows the principle of duty segregation by extending static protection on resources to dynamic authority protection on manager roles, and thus guarantees the security of sensitive system operations. 3) It follows the principle of least privilege by using a novel method of role permission activation, and thus improves the credibility of system. Moreover, we formally prove CAACM to be secure, which lays the groundwork for its deployment in cloud computing systems.

2. CAACM

2.1. Fundamental Conceptions

To illustrate the principle of CAACM, we first present several fundamental Conceptions as follows.

Trust Level of Platform. It indicates how credible the platform seem to the users of cloud computing. Generally, the trust levels of cloud computing platform can be grouped into three classes: public, secret and top-secret [17]. The permission of a manager role is different when the role accesses resources on different platforms. For

example, the manager can only access public resources on public platforms, while it can configure the whole system on top-secret platforms.

Spatial State. The location set of all entities in cloud computing. Location can be of two types [18]: hierarchic location (topological description, such as room number) and coordinate location (description in Descartes' style, such as GPS). We select the former as it is more suitable to describe spatial relation. Actually, the permission of a role is different when the role accesses resources at different locations. For instance, the manager of a branch company has system privileges in the branch, while it is only a general user in controlling corporation. More details about location detection of entities can be found in prior work [15].

Temporal State. The set of temporal constraints. The temporal information of management role can be divided with different granularity according to security requirements. For example, the time can be divided into business hours and commuting hours. The manager can access system resources during business hours. However, the manager's permission will be canceled if it login during commuting hours. Moreover, business hours can be further divided into general business hours and confidential business hours. The manager can access public resources during general business hours, while it should operate sensitive information during confidential business hours.

Definition 1. (Context). The context c is defined as a 3-tuple (f, g, t) , where:

- f denotes the trust level of platform.
- g denotes the spatial state.
- t denotes the temporal state.

For the convenience of explanation, we further define $(f, g, t) \in F \times G \times T$, where F is the set of trust levels of platforms, G is the set of spatial states and T is the set of temporal states.

Definition 2. (Manager role of cloud computing). The manager role of cloud computing *car* is defined as a 2-tuple (ar, c), where:

- ar denotes the manager role.
- c denotes the context of ar.

Analogously, we define $car \in CAR = (AR \times C)$, where AR is the set of manager roles, C is the set of Contexts. In order to follow the principle of duty segregation, we further give the definition of exclusive manager role by introducing the conception of exclusive context.

Definition 3. (Mutually exclusive context). Two contexts are mutually exclusive context (abbreviated as *ELC*), if there is no manager belonging to these contexts.

Definition 4. (Mutually exclusive manager role). Two manager roles are mutually exclusive during authorization or operation, if the members of one manager role is disjoint with the members of another manager role. The mutually exclusive manager roles during authorization and operation are denoted as SER_a and SER_r separately. If the roles are mutually exclusive during authorization and operation simultaneously, they can be denoted as *SER*.

2.2. Formal Definition

Based on the model of ARBA97, CAACM associates manager role of cloud computing with context and achieves fine-grained access control from manager to objects. The structure of proposed model CAACM is illustrated in Figure 1. The shading indicates the additional extension of CAACM. The formal definition of CAACM is presented as follows:

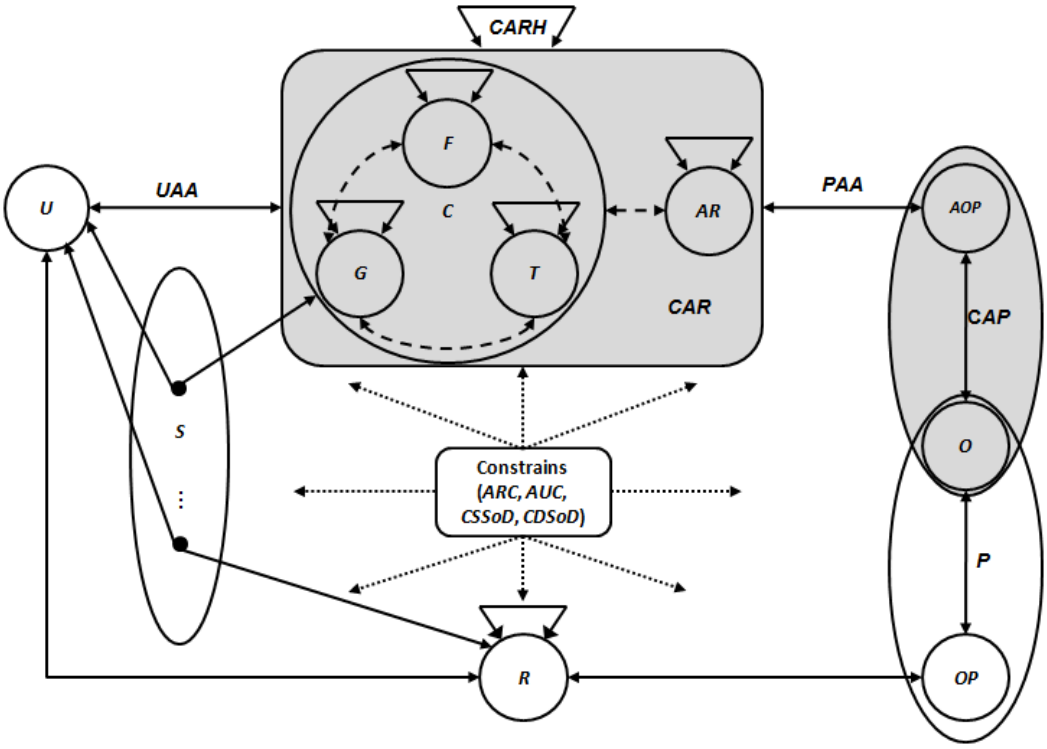

Figure 1. The structure of CAACM

Definition 5. (CAACM). CAACM is defined as a multi-tuple (*U*, *R*, *S*, *OP*, *O*, *CAR*, *AOP*, *UAA*, *PAA*, *CARH*, *ARC*, *AUC*, *CSSoD*, *CDSoD*), where:

- User set *U*, role set *R*, session set *S*, operation set *OP*, object set *O* and permission set *P* are consist with the definition of ARBA97.
- *CAR*, *AOP* and *CAP* denotes the set of manage roles, manage operations and manage permissions, where $CAR \cap R = \varnothing$, $CAP = 2^{AOP \times O}$ and $CAP \cap P = \varnothing$.
- $UAA \subseteq U \times CAR$ denotes the role assignment of users.
- $PAA \subseteq CAP \times CAR$ *denotes* the permission assignment of manage roles.
- $CARH \subseteq CAR \times CAR$ is a partially ordered set over *CAR*, which indicates the hierarchy of manager roles. Given trust levels of platform f_i and f_j, f_i is the sub-level of f_j (denoted by $f_i \leq f_j$) if $\{p_i\} \subseteq \{p_j\}$, where p_i and p_j are the permissions of a role derived from f_i and f_j separately with the same spatial state and temporal state. Analogously, $g_i \leq g_j$ and $t_i \leq t_j$ indicate similar meanings. Given $car_i = (r_i, (p_i, g_i,$

t_i))and $car_j(r_j, (f_j, g_j, t_j))$, $(car_i , car_j) \in CARH$ if and only if $r_i \leq r_j$, $f_i \leq f_j$, $g_i \leq g_j$ and $t_i \leq t_j$. For simplicity, we denote $(car_i , car_j) \in CARH$ as $car_i \leq car_j$.

- $ARC: AR \rightarrow \mathrm{N}$, cardinality constraints on manager roles, maps a manager role to a number. It describes the maximum number of managers that can be assigned to manager roles.
- $AUC: C \rightarrow \mathrm{N}$, cardinality constraints on context, maps an instance of context to a number. It describes the maximum number of managers user the instance of context.
- $CSSoD \subseteq (2^{SERa} \times N)$. *CSSoD* is the set of 2-tuple (*rs*, *n*), where *rs* represents a set of mutually exclusive manager role and *n* represents a number not less than 2. It describe the maximum number of managers that can be assigned to the roles in the set, which can be formally defined as $\forall(rs, n) \in CSSoD, \forall t \subseteq rs: |t| \geq n \Rightarrow \cap\ r \in t\ AuthorizedUser(r) = \varnothing$. *CSSoD* avoid permission conflicts at the same time by perform restrictions.
- $CDSoD \subseteq (2^{SERr} \times N)$. *CDSoD* is the set of 2-tuple (*rs*, *n*), where *rs* represents a set of mutually exclusive manager role and *n* represents a number not less than 2. It describe the maximum number of managers that can activate manager roles, which can be formally defined as $\forall(rs, n) \in CDSoD, \forall s \in S, \forall t \subseteq SessionRoles(s) \cap rs: |t| \geq n \Rightarrow \cap\ r \in t\ AuthorizedUser(r) = \varnothing$. *CSSoD* is used to avoid conflicts during the permission activation of session managers.

Compared to ARBA97, the advantages of CAACM are the following: 1) it achieves fine-grained access control from managers to objects by associating manager roles with context. 2) It guarantees multi-level security by including hierarchy of manager roles. 3) It enhances the security of cloud computing system by introducing cardinality constraints on manager roles and context. 4) It achieves management security during authorization and operation by supporting static duty segregation and dynamic duty segregation.

2.3. Access Control Policies

The security of CAACM depends on the completeness of security policies. Gavrila *et al.*, [19] have proposed twenty security policies in RBAC. Unfortunately, these policies are still insufficient since they have not taken context and role activation into account. According to the security requirements of cloud computing, CAACM should further support following six policies:

P1. The number of authorized manage users for manage roles should not exceed the cardinality of that role. Formally:

$$\forall car \in CAR, |AuthorizedUser(car)| \leq ARC(car)$$

P2. The number of authorized managers under an instance of context should not exceed the cardinality of that instance. Formally:

$$\forall u \in U, s \in S, \forall car \in CAR, \forall c \in C \square u \in SessionUser(s) \wedge (car, c) \in SessionRoles(s) \wedge u \in AuthorizedUser(car) \wedge Contains(c, GetContext(u)) \leq AUC(c)$$

P3. Two manager roles should not be assigned to the same manager is they are mutually exclusive during authorization. Formally:

$$\forall u \in U, ser_i, ser_j \in SER \square ser_i \neq ser_j \wedge u \in AuthorizedUser(ser_i) \wedge u \in AuthorizedUser(ser_j) \Rightarrow (ser_i, ser_j) \notin SER_a$$

P4. Two manager roles are mutually exclusive during authorization, if and only if the contexts of their authorization are mutually exclusive. Two manager roles are mutually exclusive during operation, if and only if the contexts of their operation are mutually exclusive. Formally:

$$\forall car_i, car_j \in CAR, \forall c_1, c_2 \in C \square ((car_i, c_1), (car_j, c_2)) \in SER_a \Rightarrow (car_i, car_j) \in SER_a \vee (c_1, c_2) \in ELC$$

$$\forall car_i, car_j \in CAR, \forall c_1, c_2 \in C \square ((car_i, c_1), (car_j, c_2)) \in SER_r \Rightarrow (car_i, car_j) \in SER_r \vee (c_1, c_2) \in ELC$$

P5. Two manager roles are not mutually exclusive if they are active in the same session. Formally:

$$\forall ser_i, ser_j \in SER, \forall s \in S \square ser_i \neq ser_j \wedge (\exists c_1, c_2 \in C \square ser_i \in EffectiveSessionRoles(s, c_1) \wedge ser_j \in EffectiveSessionRoles(s, c_2)) \Rightarrow (ser_i\ ser_j) \notin SER_r$$

P6. The context of access request should fall within the prescribed limits. Formally:

$$\forall s \in S, \forall c \in C \square (\exists car \in CAR, \exists c \in EffectiveSessionRoles(s, c))$$

The first two policies constrain the number of managers to reduce the risk of information leakage. Based on P3, P4 and P5, it is apparent that mutually exclusive manager role is the base of management duty segregation. P6 achieves dynamic permission management by constraining the context of manager roles.

2.4. Access Control Mechanism

CAACM is a dynamic access control model depending on time, location and platform environment. It assigns permission licenses according to above mentioned policies. The access control of CAACM on managers has three phases: *permission assignment*, *role activation* and *dynamic authorization*.

2.4.1. Permission assignment: There are two aspects to permission assignment: assigning manager roles to users and assigning permissions to manager roles.

Definition 6. (Manager role assignment function). Manager role assignment function is defined as *AssignedCar*: $U \times CR \rightarrow 2^{CAR}$, a mapping from users to manager roles. *CR* is the set of prerequisites that the role assignment should satisfy. For example, if the prerequisite *cr* is true for a given user *u*, then $AssignedCar(u) = \{car \in CAR | cr \in CR \wedge (u, car) \in UAA\}$

Definition 7. (Permission assignment function). Permission assignment function is defined as *AssignedCap*: $CAR \times CP \rightarrow 2^{CAP}$, a mapping from users to permissions. *CP* is the set of prerequisites that the permission assignment should satisfy, e.g., $AssignedCap(car) = \{cap \in CAP\ | cp \in CP \wedge (cap, car) \in PAA\}$ if the prerequisite *cr* is true for a given manager role *car*.

2.4.2. Role activation: In CAACM, the manager role integrated with context is essentially dynamic. Users do not have to select the role for activation directly. The roles will be configured as active or inactive automatically according to context. Firstly we present the definition of *user mapping function* and *manager role mapping function*.

Definition 8. (User mapping function). User mapping function is defined as *SessionUser*: $S \rightarrow U$, a mapping from a session to a user. In the life cycle of a session s_i, the user *SessionUser*(s_i) is invariable.

Definition 9. (Manager role mapping function). User mapping function is defined as *SessionRoles*: $S \rightarrow 2^{CAR}$, a mapping from a session to manager roles. $SessionRoles(s_i) \subseteq \{(car, c) \in CAR | (SessionUser(s_i), (car, c)) \in UAA\}$, $SessionRoles(s_i)$ indicates the set of activable manager roles in session s_i.

CAACM evaluates the relationship between mutually exclusive context and current context to determine the effective session role. The definition of valid session role are described as follows:

Definition 10. (Effective session role). Effective session role is defined as *EffectiveSessionRoles*: $S \times C \rightarrow 2^{CAR}$, where $EffectiveSessionRoles(s, c_i) = \{(r, c_j) \in CAR | (r, c_j) \in SessionRoles(s) \wedge c_i \leq c_j) = True\}$

CAACM select the validate session roles according to the context of manager. An role r is an effective session role, if r is an session role in context c_i and c_i is contained in the context of a manager.

2.4.3. Dynamic Authorization: Effective session role is the basis of making decisions on access requests. An access request can be defined as a 4-tuple (s, c, aop, o) which denotes manager of session s in context c try to perform operation aop on an object o. The following is the definition of authorization function.

Definition 11. (Authorization function). An access request aar (s, c, aop, o) $\in S \times C \times AOP \times O$, is authorized in context c if :

$$(aop, o) \in \bigcup_{car \in EffectiveSessionRoles(s, c)} AssignedCap(car) .$$

For a received access request, CAACM get the current context of manager c and permit the access if an session manager is in context c and (aop, o) is included in validate set of role authorization.

2.5. Security Proof

The operations of manager on user data and the state transition in CAACM can be proved to be security, if all system states in the model are ensured to be safe. In other words, the correctness of CAACM is proved. The following definitions are essential to describe the relation of state transition.

Definition 12. (Current access set). Curernt access set $AAR \subseteq S \times C \times AOP \times O$ are the set of operations which are performed by current session manager in current context.

Definition 13. (System State). System state of CAACM v can be defined as (*AAR*, *U*, *S*, *O*, *CAR*, *AOP*, *UAA*, *PAA*, *CARH*, *ARC*, *AUC*, *CSSoD*, *CDSoD*). For simplicity, system state can be abbreviated as (*AAR*, *OTHERS*). V is defined as the set of all system states.

Definition 14. (Policy decision set). Policy decision set D= {“yes”, “no”, “error”, “?”} includes all the possible decisions of CAACM on access requests. “yes” means the

request is authorized, while "no" means the request is rejected. "error" denotes that some unknown error is occured. "?" means CAACM is not able to handle the request.

Definition 15. (Relation of state transition). Given a rule set $\omega=\{\rho 1, \cdots, \rho_s\}$. For any request $r_k \in RE$, $d_m \in D$, set of system state V, subsequent set of system state V', the relation of state transition $W(\omega) \subset RE \times D \times V \times V'$ is defined as:

(1) $(r_k, d_m, V, V') \in W(\omega)$, if and only if $d_m \neq$ "?" and $d_m \neq$ "error".

(2) There is an unique $i(1 \leq i \leq s)$ such that $(d_m, V') = \rho_i(r_k, V)$.

Definition 16. (CAACM system). CAACM system is defined as a 4-tuple $\Sigma(RE, D, W, z_0) \subset X \times Y \times Z$, where RE is the set of manager requests, D is the policy decision set, W is the relation of state transition and z_0 is the initial state. X is defined as the set of all possible request sequence. Y is defined

Definition 17. (Security state). System state $v \in V$ is a security state if v meet the six policies of access control.

According to above definitions, input x produces the subsequent state of policy decision y and state z from the initial state z_0. The system $\Box(RE, D, W, z_0)$ includes all the execution sequence from state V_0. State sequence $\{z_1, z_2, \ldots, z_i, \ldots\}$ is a secure state sequence if z_i is secure state for all i. Each element (x, y, z) in system $\Box(RE, D, W, z_0)$ is considered as a system profile. A system profile (x, y, z) is secure if z is a secure state sequence. On this basis, the definition of secure system is presented as follows.

Definition 18. (Secure system). System $\Sigma(RE, D, W, z_0)$ is a secure system if initial state z_0 is secure and each profile $(x, y, z) \in \Sigma(RE, D, W, z_0)$ is secure profile.

LEMMA 1: $\Sigma(RE, D, W, z_0)$ is a secure system, if initial state z_0 is secure and the following conditions are true for any relation of state transition $(R_i, D_j, (AAR, OTHER), (AAR', OTHER'))$:

(1) Any subsequent operation that transforms the system state is in line with the six access control policies. That is, if $(s, c, aop, o) \in AAR' - AAR$, it will be consistent with these policies.

(2) Any request operation that is inconsistent with the six policies is not included in the set of subsequent operations.

Proof: Given an arbitrary $(x, y, z) \in \Sigma(RE, D, W, z_0)$, $z_t = (AAR_t, OTHER_t)$ for every t.

Assume $(x_1, y_1, z_1, z_0) \in W$. First of all, we should prove that z_1 is secure if z_0 is secure. Then, we will conclude that CAACM system is secure by induction.

Apparently, $AAR_1 = (AAR_1 - AAR_0) \cup (AAR_1 \cap AAR_0)$ and $(AAR_1 - AAR_0) \cap (AAR_1 \cap AAR_0) = \varnothing$.

Assume $(s, c, aop, o) \in AAR_1$, then $(s, c, aop, o) \in (AAR_1 - AAR_0)$ or $(s, c, aop, o) \in (AAR_1 \cap AAR_0)$.

Further assume $(s, c, aop, o) \in (AAR_1 - AAR_0)$, then it is consistent with the six access control policies according to condition (1).

Given $AAR^* = \{(s, c, op, o) | (s, c, op, o)$ is not consistent with the six policies, it is obvious that $(AAR^* \cap AAR_1) = \varnothing$ according to condition (2).

However, $AAR^* \cap (AAR_1 \cap AAR_0) = (AAR^* \cap AAR_1) \cap AAR_0 = \varnothing$, when AAR^* belongs to $AAR_1 \cap AAR_0$.

Thus, $(s, c, aop, o) \notin AAR^*$ if $(s, c, aop, o) \in (AAR_1 - AAR_0)$. Based on the above two cases, it is proved that z_1 is a security state since (s, c, aop, o) should be consistent with the six policies.

Moreover, z_t is proved to be secure by induction on N_t. we can conclude that system profile (x, y, z) is a secure profile. Because of the arbitrariness of (x, y, z), $\Sigma(RE, D, W, z_0)$ must be a secure system. Here this lemma is proved.

THEOREM 1: The CAACM system is secure.

Proof: The initial state of CAACM system is secure, and all subsequent operations are consistent with the six access control policies. According to LEMMA 1, we can conclude that the CAACM system is secure.

3. Application of CAACM

CAACM is suitable for outsource-oriented systems, especially for the systems based on cloud computing. In a company of cloud computing, there are: 1) system managers who manage the authorization of access control, 2) business managers who manage all system resources, 3) normal users who access public resources and their private resources. To ensure the user privacy and data security, the company has strict regulations. A system manager and a business manager should operate on secret and top-secret platforms respectively. Furthermore, they should operate during office hours and in their own office rooms.

In order to derive the spatial and temporal information of staffs, the company issues every staff with RFID tag and installs RFID reader at the entrance of every office. All manage platforms in the company has internal TPM and a integrity measurement system based on TCG architecture [20]. The CAACM system will record spatial and temporal information of a manager when he (or she) enter or leave a room by RFID verification. Moreover, the system will measure the trust level of manage platform and construct the manager context when the manager login the platform.

The above-mentioned application shows that $R=\{SM, BM, NU\}$ and $AR=\{SM, BM\}$, where SM is the set of system managers, BM is the set of business managers, NU is the set of normal users. The system context $C=\{SC, BC, NC\}$, where SC is the context of all system managers, BC is the context of all business managers, NC is the context of all normal users. The constructed manager roles of cloud computing are the following: (SM, SC), (SM, NC), (BM, BC), (BM, NC), (NU, NC).

The authorization processes of system managers and business managers are similar. Assume business manager A create session s_1 in context c_1. If c_1 is included in BC, the manager role (BM, BC) is effective in session s_1. If A create session s_1 in other context, (BM, BC) will be the only effective manage role of A and have the same permission as normal users.

Considering the disclosure risk, CAACM carries out constraints on the numbers of manager roles and managers: $AR \in ARC$ and $C \in AUC$. To defend against collusion attacks from managers, CAACM also demands that a staff should not act as a system manager and a business manager simultaneously. In other words, it carries out constraints of static duty segregation: $((SM, SC), (BM, BC)) \in CSSoD$. Moreover, CAACM demands that a business manager should not login in the context of a normal

user to protect trade secrets of the company. In other words, it carries out constraints of dynamic duty segregation: $((BM, BC), (BM, NC)) \in CDSoD$. In summary, these constraints ensure that managers can only access specific system resources at a specific time and location.

4. Model Comparison

In order to reflect the advantages of CAACM, we compare it with existing models of access control in the following aspects: 1) whether context-awareness (including time, location and platform trust level) is supported, 2) whether cloud computing is supported and 3) whether the solution of malicious insider problem is supported. The results are shown in Table 1, where "√" means supported, "×" means unsupported and "○" means partially supported.

Table 1. Comparison between CAACM and Existing Models

	Context			Cloud computing	malicious insider problem
	Time	Location	Trust level		
ARBAC97[6]	×	×	×	×	×
CAB-RBAC [7]	×	×	×	○	×
S-RBAC [8]	×	×	×	○	×
GTRBAC [11]	×	×	×	×	×
X-GTRBAC[12]	×	√	×	×	×
SCA-RBAC[13]	√	×	×	×	×
GEO-RBAC [14]	√	√	×	×	×
SG-RBAC[15]	√	√	×	×	×
ABAC[9]	√	√	×	×	×
CCACSM[10]	√	√	×	○	×
CAACM	√	√	√	√	√

Table 1 shows that ARBAC97, CAB-RBAC and S-RBAC are not suitable for cloud computing since they cannot restrict the spatial state, temporal state and platform trust level of managers. Although GTRBAC and X-GTRBAC are more flexible by analyzing temporal constraints, they still do not support constraints on location and trust level. SCA-RBAC, GEO-RBAC, SG-RBAC and ABAC involves the conception of role location, but they do not take platform trust level into account and ignore the dependence of access control on role location. Moreover, CCACSM in [10] performs access control only on normal users and ignores the risks from malicious insiders. Generally, these models are carried out without consideration for the characteristics of cloud computing and their application is restricted. By comparison, CAACM not only involves constraints on context including time, location and platform trust level, but also achieves fine-grained access control on managers. Thus, it will be widely applied in cloud computing.

5. Conclusion and Future Work

Permission management is a key security problem of cloud computing. In view of the characteristics and demands of cloud computing on manager access control, a Context-Aware Access Control Model based on ARBAC97 for cloud computing (CAACM) was proposed. The model not only inherits the advantages of ARBAC97, but also protect the privacy and data security of users by adding context and manager roles. Moreover, we formally prove the security of CAACM to ensure that malicious insiders are not able to compromise the data security of users.

As an access control model, CAACM is worthy of further study. One of greatest challenges is how to ensure the integrity of CAACM platform. The trust level of platform will lose its credibility if the integrity is broken. How to monitor the integrity of platform in full-life cycle is a hot issue for our further research. Credibility measurement is another challenge for CAACM system. It is usually a time-consuming task with high redundancy since cloud computing system contains a great number of VMs. How to improve the efficiency of credibility measurement is also an important research topic.

Acknowledgements

This work is supported by Natural Science Foundation of Jiangsu China under Grant No. BK2011115, and also supported by Laboratory of Military Network Technology of PLA University of Science and Technology.

References

[1] Y. Chen, V. Paxson and H. K. Randy, par University of California at Berkeley, (**2010)**.
[2] Top Threats to Cloud Computing, http://www.cloudsecurityalliance.org/topthreats/csathreats.v1.0.pdf.
[3] T. Mather, S. Kumaraswamy and S. Latif, "Cloud Security and Privacy", USA: O Reilly Media, (**2009)**.
[4] Cloud Security Alliance, "Security Guidance for Critical Areas of Focus in Cloud Computing V2.1", The Cloud Security Appliance, (**2009)** December.
[5] D. G. Feng, M. Zhang, Y. Zhang, *et al.*, "Study on cloud computing security", Journal of Software, vol. 22, no. 1, (**2011)**, pp. 71-83.
[6] R. S. Sandhu, V. Bhamidipati and Q. Munawer, "The ARBAC97 model for role-based administration of ro1es", ACM Trans. Information and System Security, vol. 2, no. 1, (**1999)**, pp. 105-135.
[7] A. Sirisha and G. Geethakumari, "API access control in cloud using the role based access control model", Trendz in Information Sciences and Computing-TISC2010, (**2010)**.
[8] D. Li, C. Liu, Q. Wei, *et al.*, "RBAC-Based access control for SaaS systems", 2010 2nd International Conference on Information Engineering and Computer Science (ICIECS), (**2010)**.
[9] F. H. Li, W. Wang, J. f. Ma, *et al.*, "Action-based access control model and administration of actions", Acta Electronica Sinica, vol. 36, no. 10, (**2008)**, pp. 1881-1890.
[10] G. -y. Lin, S. He, H. Huang, *et al.*, "Access control security model based on behavior in cloud computing environment", Journal on Communications, vol. 33, no. 3, (**2012)**, pp. 59-66.
[11] J. Joshi, E. Bertino, U. Latif, *et al.*, "A generalized temporal rolebased access control model", IEEE Trans on Knowledge and Data Engineering, vol. 17, no. 1, (**2005)**, pp. 4- 23.
[12] R. Bhatti, A. Ghafoor, E. Bertino, *et al.*, "X-GTRBAC: an XMLbased policy specification framework and architecture for enterprise-wide access control", ACM Trans on Information and System Security, vol. 8, no. 2, (**2005)**, pp. 187-227.
[13] M. J. Covington, W. Long, S. Srinivasan, *et al.*, "Securing context-aware applications using environment roles [A]. Proc of 6th ACM Symposium on Access Control Models and Technologies[C]. New York: ACM, (**2001)**, pp. 10-20.
[14] M. L. Damiani, E. Bertino, B. Catania, *et al.*, "GEO-RBAC: a spatially aware RBAC", ACM Trans on Information and System Security, vol. 10, no. 1, (**2007)**, pp. 1-42.
[15] Z. Hong, H. Ye-ping and S. Z. Guo, "A formal model for access control with supporting spatial context", Science in China Series F: Information Sciences, vol. 50, no. 3, (**2007)**, pp. 419-439.

[16] N. Santos, K. P. Gummadi and R. Rodrigues, "Towards trusted cloud computing", Sahu S, (ed.), USENIX Association Proc. of the Workshop on Hot Topics in Cloud Computing 2009, San Diego, (**2009)**.
[17] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum and D. Boneh, "Terra: A virtual machine-based platform for trusted computing", Scott ML, Peterson LL, (eds.), Proc. of the 19th ACM Symp. on Operating Systems Principles (SOSP 2003). New York: ACM Press, (**2003)**, pp. 193−206.
[18] C. Jiang and P. Steenkiste, "A hybrid location model with a computable location identifier for ubiquitous computing", Proceedings of the 4th International Conference on Ubiquitous Computing. London: Springer-Verlag, (**2002)**, pp. 246-263
[19] S. I. Gavrila and J. F. Barkley, "Formal specification for role based access control user/role and role/role relationship management", Proceedings of the 3rd ACM Workshop on Role-Based Access Control, New York: ACM Press, (**1998)**, pp. 81-90.
[20] R. Sailer, X. Zhang, T. Jaeger and L. van Doorn, "Design and implementation of a TCG-based integrity measurement architecture", Proc. of the 13th USENIX Security Symp. Berkeley: USENIX, **(2004)**, pp. 223−238.

Authors

Zhenji Zhou

He received his MS degree in computer science from the PLA Uni. of Sci. & Tech in 2010. Now he is a PhD candidate at the Institute of Command Automation, PLA University of Science and Technology (PLAUST). His current research interests include different aspects of Information Security and Cloud Security.

Lifa Wu

He received his PhD degree in computer science from the Nanjing University. Now he is full professor of Institute of Command Automation, PLAUST. His current research interests include different aspects of Network Security and Protocol Reverse Engineering.

Zheng Hong

He received his PhD degree in computer science from PLAUST in 2007. Now he is an associate professor of the Institute of Command Automation, PLAUST. His current research interests include different aspects of Malware Detection and Network Security.

