**H-ABAC: A Hierarchical System Based Access Control Model for Self-administering Management in Information System**

Gang LIU, Hui-min SONG, Run-nan ZHANG, Can WANG, Quan WANG* and Lu FANG  
No. 2 South Taibai Road XIDIAN University, Xi'an, China, 710071  
*Corresponding author

**Keywords:** Access control, Hierarchical system, Attribute based access control(ABAC), Role based access control(RBAC), System security.

**Abstract.** Currently, resource sharing and system security are critical issues. This paper proposes a hierarchical system based access control model termed H-ABAC. It improves the attribute based access control(ABAC) model in reducing the management stress and addressing the problem of insufficient policy repository space. The following describes the H-ABAC model in terms of its self-administering mode, attribute definitions, policy formulation and authorization architecture, which demonstrate the advantages of H-ABAC. The supervisor mode not only reduces system administrator stress but also prevents all problems of a session in role-based access control (RBAC). Different definitions of the attributes in H-ABAC can be employed for different fields with different access control granularities, which enhances the flexibility of the model compared with traditional access control models. A scenario that illustrates how this new model is applied to the real world is provided.

**Introduction**

With the development of the Internet in the last several decades, resource sharing has attracted significant attention. The emergence of shared resources has caused numerous concerns with security. HongHai Shen and Prasun Dewan suggest that access control is an indispensable part of any resource sharing systems[1]. Because access control only permits legally authorized subjects to access protected resources[2], illegal subjects are prevented from accessing protected resources and legal users are prevented from unauthorized access to protected resources. Many researchers have investigated access control, but the permission management in access control model is still a problem which have not been addressed.

Nowadays, one of the mainstream access control models is attribute-based access control (ABAC) model. ABAC is a flexible approach that implements access control policies limited by the computational language and richness of available attributes only, which renders ideal for many distributed or rapidly changing environments[3]. The attributes of ABAC refer to subject attributes, resource attributes, operation attributes and environment attributes. ABAC can be applied to different fields by defining different attribute types and the context can be implemented with simple semantics. However, apart from the advantages above, ABAC not only has the problems of insufficient policy repository space and the heavy management stress, but also cannot process abnormal situations not described by policy rules.

Based on the analysis of above-mentioned model, this paper proposes a new hierarchical system structure based ABAC model termed H-ABAC. Firstly, this hierarchical system in H-ABAC model achieves self-administering with the management of the administrator to reduce the heavy management stress in ABAC and addresses some problems which are not described by policies in abnormal circumstance. This hierarchical system represented inclusion relation is different from the classic hierarchical control system proposed several years ago. Generally, the classic hierarchical control system has been applied on artificial intelligence, manufacturing[4], web service[5] and so
on while H-ABAC focuses on the management of permission and the assignment of subjects and resources. Secondly, H-ABAC inherits all the advantages of ABAC, such as implementing the context with simple semantics and making it a valuable candidate for securing collaboration between organizations, especially over an open network[6]. Thus, H-ABAC is flexible in the management of permission and the assignment of subjects and resources.

This paper is organized as follows: section 2 interprets the hierarchical system and the administrator privileges in different hierarchies; section 3 details the attributes definition and the access control policies to manage users and administrators; section 4 describes a scenario for illustrating the application method; section 5 summarizes the whole thesis and provides the future development tendency.

Structure of the Hierarchical System

Hierarchy Definition

The system consists of a lot of hierarchies based on the real-world. In order to illustrate more clearly, this paper defines the system structure as three hierarchies in Fig. 1.

Fig. 1 shows that the application system has been divided into a lot of domains and each domain has also been divided into a lot of areas. Each subject and resource can be assigned to the different domains or areas at the same time.

Area is an abstraction that owns a collection of subjects and resources. In one area, parts of the subject or resource attributes may have the same value, such as subjects of the same age, subjects of the same goal or resources of the same type. Area has a many-to-many relation with subjects and resources. Each area only has one area administrator to manage the subjects and resources assigned. The area administrator permission will be detailed in the next section.

Domain is an abstraction that owns a collection of similar areas, subjects and resources in order to manage areas conveniently. Each domain only has one domain administrator to manage the area assigned. Moreover, the subjects not assigned to any areas will be managed by the domain administrator directly which only use the subject permission authorized by the system in advance. The domain administrator permission will be detailed in the following also.

System does not refer to just a computer system. That is to say, system also can be an embedded systems or an application system such as an enterprise management system and so on. Moreover, a system includes at least one domain.

Administrator Permission

Administrators are composed of area administrators, domain administrators and system administrator. Different administrators manage different fields but can be assigned to the same
subject. In H-ABAC, the system administrator role is senior to domain administrator and thereby inherits all permissions from intra-system domain administrator. The system administrator role can have permissions besides those it inherited. So the relations among the domain administrator role and the intra-domain area administrators are. This inheritance relation is similar to the role hierarchies proposed in [7]. The next section introduces the three types of administrator permission.

Area Administrator (AAR). The area administrator is the leader of an area, which not only assigns the resources to the subjects in the area but also corresponds with the domain administrator. Thus, the area administrator is allowed to perform the following tasks: remove the area subjects; modify specific attribute values of the area resources; assign the area permission to the area subjects; make, modify and delete the area policy rules; apply for other subjects and resources from the domain administrator.

A system must limit parts of the area administrator permission to guarantee that the area administrator has no effect on subjects and resources outside of the area. Thus, the area administrator is not allowed to perform the following tasks: add the subjects into the area without applying to the domain administrator; create the area and the area administrator.

After an area is assigned subjects and resources by the domain administrator, the area administrator determines whether the resource permission of the area needs to be modified. If a resource is authorized for the entire area, the resource permission remains the same. If a resource is authorized to one or more area subjects, the area administrator makes policy rules to grant the permission to the area subjects. In this manner, H-ABAC enables the permission to be dynamically expanded or restricted but not having redundancy. When revoking a resource permission from an area subject, the area administrator deletes the policy rule directly.

When adding a subject to an area, the area administrator gains approval from the domain administrator. After the domain administrator adds the area id to a subject, the area administrator assigns a permission to the subject. The area administrator must assure that each area resource is accessible. The area administrator should focus on attribute redundancy because repeated authorization is not allowed.

Domain Administrator (DAR). The domain administrator manages at least one area while the area administrator only manages the area to which it belongs. The domain administrator also manages the subjects and resources that are not assigned to any areas in the domain. Thus, the domain administrator has the permission of performing all the operations for all subjects and resources in the domain. The domain administrator is allowed to perform the following tasks: create areas; assign the domain subjects and resources into the area; make the policy rules only used in the domain or area; suspend areas; activate areas; delete areas; apply for other subjects and resources from the system administrators.

When creating an area, the domain administrator assigns the subjects and resources that are relevant to the area with adding the area id. After creating the area, the domain administrator must grant the area administrator role to only one subject. When the area has encountered problems, the domain administrator adds a context to enable the area to enter the suspension state. When the problems have been solved, the domain administrator deletes the context to enable the area to enter the running state. During the system running time, the area administrator obtains subjects and resources from the domain administrator. The domain administrator must assure that each domain resource is accessible.

System Administrator (SAR). In H-ABAC, the system administrator has the permission of performing all operations for all subjects and resources in the entire system. The system administrator is allowed to perform the following tasks: create domains; delete domains; provide a recycling mechanism; add the subject permission; delete the subject permission.
When creating a domain, the system administrator assigns subjects and resources that are relevant to the domain. After creating the domain, the system administrator must grant the domain administrator role to only one subject. When deleting a domain, the system administrator revokes all permissions, subjects and resources in the domain. In addition, the system provides a recycling mechanism to delete unnecessary resources that are confirmed by the system administrator. The system administrator assures that each system resource is accessible.

In this way, the domain administrator enables domains to self-administer, which is similar to the area. Thus, the domain reduces the management stress from the system administrator further and even adds some constraints among areas through contexts.

**H-ABAC Model**

This section defines four types of attributes, which are based on the previously described hierarchical system; introduces policies, including access control policies and administrator access control rules; proposes the architecture of H-ABAC; and illustrates H-ABAC with an application scenario.

**Attribute Definitions**

The basic attributes in H-ABAC model consist of subject attributes, resource attributes, operation attributes and environment attributes, which are represented by SA, RA, OA and EA, respectively.

**Subject Attribute (SA).** A subject is an entity (e.g., a user, application, or process) that takes action on a resource. A subject attribute in H-ABAC is defined as \((s, role, ... )\). Each subject (abbreviated as \(S\)) has a unique subject id in the system. The role includes a set of role ids that are assigned to a subject to judge whether the subject is administrator. To recognize the role to which the area belongs, the role id consists of the belong id and the role id with the introduction of the abstraction symbol ‘&’. The symbol ‘&’ is only used to connect the id without any mathematical calculations. The belong id consists of the domain id and the area id with the introduction of the abstraction symbol ‘&’. This method guarantees the permission that is granted to the role will only be employed when the subject is assigned to the area to which the role belongs. The actual application determines whether the subject, domain, area or role are described as a set of positive integers and their scope.

**Operation Attribute (OA).** An operation is an action (e.g., write and read) in which a subject employs a resource. The operation attribute values are represented by a set of unique operation types, which are termed \(O\), in the system. A system can define different operation types in different actual application.

**Resource Attribute (RA).** A resource is an entity (e.g., a web service, data structure, or system component) that is acted on by a subject. Each resource (abbreviated as \(R\)) has a unique resource id in the system. The resource attribute in the H-ABAC model is defined as \((R, ... )\). The resources are assigned to different autonomous domains and areas through the different policy rules.

**Environment Attribute (EA).** The definitions of the environment attributes in H-ABAC and ABAC are identical. The environment attributes are very special attributes. For the environment attributes, attribute authentication is based on conditions in the environment when a request is made in ABAC[8]. In H-ABAC, the environment attributes are used to suspend and activate areas. For example, the domain administrator adds an access control rule, in which the subject attribute value is 7&3 and the environment attribute value is in the suspension state. When activating an area, the domain administrator only deletes this rule directly.
H-ABAC Policies

In the previous section, attributes have been determined. This section introduces the access control policy in H-ABAC.

Basic Policy Model. The H-ABAC policy model is defined based on [9]:

1) \( S_A \ (1 \leq k \leq K) \), \( R_A \ (1 \leq m \leq M) \), and \( O_A \ (n = 1) \) are the pre-defined attributes for subjects, resources, and operations, respectively;

2) \( \text{ATTR}(s), \text{ATTR}(r), \) and \( \text{ATTR}(o) \) represent attribute assignment relations for subjects, resources and operations respectively which covers all attributes values:

- \( \text{ATTR}(s) \subseteq S_A1 \times S_A2 \times ... \times S_AK \);
- \( \text{ATTR}(r) \subseteq R_A1 \times R_A2 \times ... \times R_AM \);
- \( \text{ATTR}(o) \subseteq O_A1 \).

For example, the system formalizes represent a subject whose \( S \) is ‘123’, \( \text{domain id} \) is ‘7’, \( \text{area id} \) is ‘2’, and \( \text{role id} \) is ‘area administrator’ as follows: \( \text{ATTR}(s) = (S_A1(s = 123), S_A2(\text{domain} = 7), S_A3(\text{area} = 7&2), S_A4(\text{role} = 7&2&\text{area administrator})) \).

Basic Policy Model. In the most general form, a policy rule decides on whether a subject can take an operation on a resource which is a Boolean function of \( s, r \) and \( o \) attributes:

\[
\text{Rule: } \text{can}_\text{access} \ (s, r, o) \leftarrow f \ (\text{ATTR}(s), \text{ATTR}(r), \text{ATTR}(o)).
\]

Given all the attribute assignments of \( s, r \) and \( o \), if the function’s evaluation is true, then the access to the resource is granted; otherwise the access is denied.

A simple example will illustrate the essence of the rule. Above all, the system defines: \( \text{ATTR}(s) \subseteq S_A1(id) \times S_A2(age) \); \( \text{ATTR}(r) \subseteq R_A1(id) \); \( \text{ATTR}(o) \subseteq O_A1(type) \). Thus, the value of the attributes has been defined as follows: \( S_A1(id) \subseteq (1..200) \); \( S_A2(age) \subseteq (16..100) \); \( R_A1(id) \subseteq (7, 8) \); \( O_A1(type) = \{\text{observe, write, modify} \} \). Meanwhile, if the system wants the id of the subject between 1 and 100 to observe the resource which the id value is 7, it just makes the access control policy rules as follows:

\[
\text{Rule: } \text{can}_\text{access} \ (s, r, o) \leftarrow (id(s) \in (1..100)) \land (\text{type}(o)=\text{observe}) \land (id(r)=7).
\]

In this way, system achieves large-scale authorization to subject in the system; authorizes based only one or several attributes of subject; modifies the authorization at any time. Thus, this access control policy rule not only ensures the information and user privacy security, but also improves the flexibility and dynamic authorization management.

Administrator Access Control Rules. This section proposed a special rule for administrators, which is termed the administrator access control rule. The following rules are only employed by the administrator to modify the attribute values of subjects and resources.

When removing a subject from an area, the area administrator deletes the \( \text{area id} \) that is relevant to the subject. When creating an area, the domain administrator adds the \( \text{area id} \) to the relevant subjects and resources. When deleting an area, the domain administrator deletes the \( \text{area id} \) from the relevant subjects and resources. When creating a domain, the system administrator adds the \( \text{domain id} \) to the relevant subjects and resources. When deleting a domain, the system administrator deletes the \( \text{domain id} \) from the relevant subjects and resources.

All administrator access control rules, including the formalized definitions that are used by subject \( s \) to take an operation on subject \( s1 \) or resource \( r1 \) are shown in Table 1.
<table>
<thead>
<tr>
<th>Administrator</th>
<th>Permission</th>
<th>Formulation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Area Administrator (AAR)</td>
<td>Delete the area id from the $s_1$</td>
<td>$can_access\ (s, s_1, o) \leftarrow (\text{Role}(s) = \text{AAR}) \land (\text{Area}(s_1) = \text{area id}) \land (\text{O}(o) = \text{Delete})$</td>
</tr>
<tr>
<td>Domain Administrator (DAR)</td>
<td>Add the area id to the $s_1$</td>
<td>$can_access\ (s, s_1, o) \leftarrow (\text{Role}(s) = \text{DAR}) \land (\text{Area}(s_1) = \text{area id}) \land (\text{O}(o) = \text{Add})$</td>
</tr>
<tr>
<td>Domain Administrator (DAR)</td>
<td>Add the area id to the $r_1$</td>
<td>$can_access\ (s, r_1, o) \leftarrow (\text{Role}(s) = \text{DAR}) \land (\text{Area}(r_1) = \text{area id}) \land (\text{O}(o) = \text{Add})$</td>
</tr>
<tr>
<td>Domain Administrator (DAR)</td>
<td>Delete the area id from the $r_1$</td>
<td>$can_access\ (s, r_1, o) \leftarrow (\text{Role}(s) = \text{DAR}) \land (\text{Area}(r_1) = \text{area id}) \land (\text{O}(o) = \text{Delete})$</td>
</tr>
<tr>
<td>System Administrator (SAR)</td>
<td>Add the domain id to the $s_1$</td>
<td>$can_access\ (s, s_1, o) \leftarrow (\text{Role}(s) = \text{SAR}) \land (\text{Domain}(s_1) = \text{domain id}) \land (\text{O}(o) = \text{Add})$</td>
</tr>
<tr>
<td>System Administrator (SAR)</td>
<td>Add the domain id to the $r_1$</td>
<td>$can_access\ (s, r_1, o) \leftarrow (\text{Role}(s) = \text{SAR}) \land (\text{Domain}(r_1) = \text{domain id}) \land (\text{O}(o) = \text{Add})$</td>
</tr>
<tr>
<td>System Administrator (SAR)</td>
<td>Delete the domain id from the $s_1$</td>
<td>$can_access\ (s, s_1, o) \leftarrow (\text{Role}(s) = \text{SAR}) \land (\text{Domain}(s_1) = \text{domain id}) \land (\text{O}(o) = \text{Delete})$</td>
</tr>
<tr>
<td>System Administrator (SAR)</td>
<td>Delete the domain id from the $r_1$</td>
<td>$can_access\ (s, r_1, o) \leftarrow (\text{Role}(s) = \text{SAR}) \land (\text{Domain}(r_1) = \text{domain id}) \land (\text{O}(o) = \text{Delete})$</td>
</tr>
</tbody>
</table>

In Table 1, the role represents all roles that are assigned to the subject. Thus, when the system determines whether the subject had the administrator role, the value of the area id in the role id must be consistent with the current operation environment id. The area administrator only modifies parts of the resource permission data, including the area id or the area id & role id. The system administrator applies all rules of the domain administrator, whereas the domain administrator applies all rules of the area administrator. There are also some operations in policy rules cannot be described by the formalized definition: the area administrator makes the area policy rules; the domain administrator creates the area id; and the system administrator creates the domain id.

**Architecture of H-ABAC**

A typical authorization architecture of H-ABAC model is illustrated in Fig. 2:
This architecture integrates XACML in real implementations. XACML (eXtensible Access Control Markup Language) is an OASIS standard that describes both a policy language implemented in XML and an access control decision request/response language implemented in XML[10]. The XACML profile specifies four main actors in H-ABAC model to handle access decisions according to [11]: Policy administration point (PAP): The system entity creates a policy or policy set; Policy information point (PIP): The system entity acts as a source of attribute values. That is to say, PIP is the point where the necessary attributes for the policy evaluation are retrieved from several external or internal actors[12]; Policy enforcement point (PEP): The system entity performs access control by making decision requests and enforcing authorization decisions; Policy decision point (PDP): The system entity evaluates applicable policy and renders an authorization decision.

When getting an access request, PDP collects all the necessary values of SA, RA and OA from PIP, such as the area id. Then, PAP finds the policy rules that are relevant to the access. Next, PDP combines the attribute values with policies. Finally, PDP makes an access control decision and return it to PEP. If the access has been permitted, the subject takes this action on the resource. If the access has been denied, PEP returns the information to the subject. If there are no policy rules relevant to judge whether the access is permitted, PDP returns the inapplicable to PEP and PEP returns the information to the subject.

An illustrative Scenario

This section uses a simple Chinese bulletin board system (BBS) forum as an example to illustrate the operation mode of H-ABAC model. First of all, the simple structure of the BBS is described in Fig. 3.

In this BBS, users have been divided two types, visitors and registered users. The users registered have different permission according to their credits. When a registered user has created a new resource which means a post in the BBS, the administrator can take actions on him or her. That is to say, the user has been assigned to the area managed by the administrator. User assignment is based on the policy rules made by the system administrator while resource assignment is based on its attribute value. BBS administrator makes different policy rules based on the user role and credit as follows:

- Visitors can only browse the resource in the information section.
- Registered users whose credit is greater than 0 and less than 1000 can not only browse and reply the post verified by the administrator but also create a new post in the sharing board.
- Registered users whose credit is equal to 1000 or greater than 1000 can not only browse and reply the post approved by the administrator but also create a new post in the sharing and entertainment board.
- The user can modify and delete the post which he or she creates.
- The registered users whose role is the webmaster can create, delete, modify and reply the post even changing its verification state within their own management limitation. Besides, the webmaster also has the authority to add or reduce credits of the users who create or reply the post within their management limitation.
- The registered users whose role is the super webmaster can create, delete, modify and reply the post even changing its verification state within their own management limitation. Besides, the webmaster also has the authority to add or reduce credits of the users who create or reply the post within their management limitation.

---

**BBS Forum**

<table>
<thead>
<tr>
<th>Sharing Board(Domain 1):</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Section(Area 1): Restore information and news relevant.</td>
</tr>
<tr>
<td>Resource Section(Area 2): Share and download resources.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Entertainment Board(Domain 2):</th>
</tr>
</thead>
<tbody>
<tr>
<td>Irrigation Section(Area 1): Chat with friends whatever you want.</td>
</tr>
<tr>
<td>Exhibition Section(Area 2): Show yourselves.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Logistics Board(Domain 3):</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meeting Section(Area 1): Have a meeting among administrators.</td>
</tr>
<tr>
<td>Recycle Section(Area 2): Place the unapproved or deleted posts.</td>
</tr>
</tbody>
</table>

---

BBS is a complete system and the structure of it presents an inclusion relation. Thus, using H-ABAC model to manage the users and posts is flexible and feasible. The forum administrator is the system administrator, super webmasters are the domain administrators and webmasters are the area administrators. Administrators use the administrator access control rules to achieve self-administering, such as the addition or reduction of the user credits. Domain and Area ids are also labeled in Fig. 3.

According to the scenario above, system defines SA, RA, OA and makes the policy rules. Above all, the attribute of subject, resource and operation has been described in Fig. 4.

---

**Figure 3. Structure of the BBS Forum.**

**Figure 4. Attribute Definition in the BBS Forum.**
Especially, the value of Type(o) has been defined as (browse, create, modify, delete, reply, change), the value of CS(r) has been defined as (approve, unapproved, others), the value of belong(r) has been defined as (Domain id & Area id) and other attribute values have been defined based on the practical application, such as the value of ID(s) based on his or her register time. If the subject is not an administrator, the value of ML(s) will be an empty set; If the role of the subject includes the area administrator and the domain administrator, the value of ML(s) will be the union of all the ML(s) values of the roles. The modification time refers to the last modification time. After defining the attribute, the policy rules are made in the following:

Rule1: can_access(s, r, o) ← (Role(s) = visitor) ∧ (Belong(r) = 1&1) ∧ (CS(r) = approve) ∧ (Type(o) = browse).

Rule2: can_access(s, r, o) ← (Role(s) = registered user) ∧ (Credit(s) ≥ 0) ∧ (Credit(s) < 1000) ∧ (Belong(r) = 1&1 ∧ 1&2) ∧ ((Type(o) = create) ∨ ((CS(r) = approve) ∧ (Type(o) = browse ∧ reply))).

Rule3: can_access(s, r, o) ← (Role(s) = registered user) ∧ (Credit(s) ≥ 1000) ∧ (Belong(r) = 1&1 ∧ 1&2 ∧ 2&1 ∧ 2&2) ∧ ((Type(o) = create) ∨ ((CS(r) = approve) ∧ (Type(o) = browse ∧ reply))).

Rule4: can_access(s, r, o) ← (ID(s) = Creator(r)) ∧ (Type(o) = delete ∧ modify) ∧ (CS(r) = approve).

Rule5: can_access(s, r, o) ← (Role(s) = webmaster) ∧ (Belong(r) = ML(s)) ∧ (Type(o) = create ∧ delete ∧ modify ∧ change ∧ reply).

Rule6: can_access(s, r, o) ← (Role(s) = super webmaster) ∧ (Belong(r) < ML(s)) ∧ (Type(o) = create ∧ delete ∧ modify ∧ change ∧ reply).

By making a series of policy rules and adding the administrator permission, H-ABAC can achieve the self-administrating. When something unexpected has happened, both webmaster and super webmaster have authority on dealing with it using the administrator permission. In this way, H-ABAC reduces the stress from system administrator with distributing it into several parts to domain and area administrators.

When exceptions occur, such as one user taking a large number of replying actions in several seconds, ABAC cannot make rules to help the system address the problem unexpected. All the abnormal circumstances cannot be expected in advance. However, administrator can use his or her administrator permission to reduce the user credits in H-ABAC. When the credit of the user is negative, the user cannot do any actions in the system. Similarly, if the resource is in the abnormal state, administrator deletes it directly. Thus, administrators address the abnormal circumstances through evaluating them and taking actions on their credit or current state to protect the system security.
Conclusions

This paper proposes an improved ABAC model based on hierarchical system termed H-ABAC which achieves self-administering. It maintains the safety of the system and achieves the resource sharing through ABAC policy rules. H-ABAC supports for the authorization to the whole area and the interaction between the system and the subject. For example, the subject can accept or refuse to be granted the administrator role. H-ABAC also has different access control granularity with different definition of attributes. If integrity level has been defined as the subject attribute and the access control rules has been defined based on level, the coarse-grain access control has been obtained; if the definition of attributes is like above, in the form of id, the fine-grain access control has been obtained.

Moreover, a system with many subjects and resources needs to make a larger number of policy rules to manage which may lead to the problem of insufficient policy repository space in ABAC. Thus, the authorization and assignment management are also difficult. H-ABAC assigns the subjects and resources into different domains and areas managed by administrators with the self-administering management. Different domains and area can make different policy rules which are stored into different space. By this way, H-ABAC reduces the management stress and addresses the problem of the insufficient policy repository space. Administrators even process some abnormal situations not described by policy rules with administrator permission. To sum up, H-ABAC has all the merits existed in ABAC and solves some of its shortages.

Through comparative analysis, this paper indicates that H-ABAC model is feasible. As for the overall system, domain can be divided into several hierarchies to achieve multi-hierarchy which may be the one of future research directions. The environment attribute has not been detailed which is only used to suspend and activate areas. Similarly, the constrain rules between domains or areas have not been decided. In addition, both the assignment rules of the subject and resource attributes and the conflict resolution may also become the further research directions in H-ABAC.

Acknowledgements

This research was financially supported by the National Natural Science Foundation (NNSF) of China (Grant No. 61572385).

Conflict of interest statement: There is no conflict of interest regarding the publication of this paper.

References


