Journal of Universal Computer Science, vol. 24, no. 4 (2018), 392-416
submitted: 15/12/17, accepted: 15/4/18, appeared: 28/4/18

A New Hybrid Access Control Model for Security Policies in Multimodal Applications Environments

Hasiba Ben Attia
(LINFI Lab, Computer Science Departement, Biskra University, Biskra, Algeria
ben.attia.hasiba@gmail.com)

Laid Kahloul
(LINFI Lab, Computer Science Departement, Biskra University, Biskra, Algeria
kahloul2006@yahoo.fr)

Saber Benharzallah
(Batna 2 University, Batna, LINFI Lab, Biskra University, Biskra, Algeria
sbharz@yahoo.fr)

Abstract: New technologies as cloud computing and internet of things (IoT) has expanded the range of multimodal applications. This expansion, in several computing and heterogeneous environments, makes access control an important issue in multimodal applications. Indeed, a variety of access control models have been developed to address different aspects of security problems. The two most popular basic models are: Role Based Access Control (RBAC) and Attribute Based Access Control (ABAC). The both models RBAC and ABAC have their specific features and they can complement each other. For that, providing a hybrid model which considers both concepts "roles" as well as "attributes" has become an important research topic. This paper proposes a new access control model based principally on roles, attributes, access modes and the type of resources. An empirical method is applied to compare the new proposed model versus three existing models: RBAC, ABAC, and the hybrid model Attribute Enhanced RBAC (AERBAC). The results of the empirical method demonstrate that the new proposed model acquires the advantages of the two models RBAC and ABAC and avoids their limitations. In fact, the new proposed model reduces the complexity of security policies and allows expressing the fine granularity of systems without any explosion in the number of roles or rules in the security policy.

Key Words: Security policies, Access control, Hybrid access control model, RBAC, ABAC, Comparative method, Flexible model, Scalable model, Fine-grained model

Category: H.1, K.6.5, D.4.6

1 Introduction

Multimodal applications has expanded rapidly to touch large number of domains: Internet of things (IoT) in [Markku et al., 2015], education (eg. m-learning) in [Alghabban et al., 2017], health care in [Reinschluessel et al., 2017], military in [Aaltonen and Laarni, 2017]. In this large range of areas' systems, complexity of security administration remains an important challenge. Many models were proposed to deal with this complexity. Currently, there are two key basic models that are the most used models in the design and the implementation of security policies, in large networking systems. These two models are: Role Based Access Control (RBAC) [Ferraiolo et al., 2001] and Attribute Based Access Control (ABAC) [Brossard et al., 2017]. Each of RBAC and ABAC has their benefits and drawbacks; therefore, providing a hybrid model that combines the benefits of these two techniques will achieve flexible, scalable and fine-grained access control [Varadharajan et al., 2015]. Initially, designed as a model to mainstream commerce systems, RBAC and ABAC have found applications in several areas: health care using RBAC in [Moon Sun Shin and Jeong, 2015] and using ABAC in [Mukherjee et al., 2017], work-flow systems using RBAC in [Liu et al., 2015], education using RBAC in [Le et al., 2014], web services and their architecture using RBAC in [Ranchal et al., 2016] and using ABAC in [Zhang and Zhang, 2017], social networks using RBAC in [Pang and Zhang, 2015] and using ABAC in [Hsu and Ray, 2016], wireless networks using RBAC in [Nagarajan and Gopalan, 2016a] and [Chen et al., 2016], cloud computing using RBAC in [Luo et al., 2016] and using ABAC in [Ngo et al., 2016], financial industry using ABAC in [Qiu et al., 2016], mobile environment using ABAC in [Li et al., 2014], etc. Several work were proposed to combine the advantages of the ABAC and RBAC and overcome their limits. To evaluate these hybrids models, we need to compare them with the both models.

In this paper, we aim to propose a new "hybrid access control model" based on the two existing models RBAC and ABAC. The new proposed model inherits the advantages of the two models and aims to overcome their limitations. In order to demonstrate the strengths of the new proposed model, an empirical comparison is realised. This empirical comparison is based on four metrics that are inspired form the limitations of RBAC and ABAC. The empirical comparison is used to evaluate the proposed new model versus three existing models: RBAC, ABAC and AERBAC (Attribute Enhanced RBAC). From the obtained results, it is proved that the new hybrid model provides more flexibility, scalability, fine-grained capacity.

The rest of the paper is organized as follows. Section 2 exposes related work dealing with RBAC, ABAC and hybrid models. Section 3 starts by presenting the requirements and needs for a new access control model, then it presents the principle and the components of the new proposed access control model. Section 4 details the principles of the proposed empirical comparison approach, the proposed metrics, a demonstrative example, and analyses the models RBAC, ABAC, AERBAC and the new proposed model using the defined metrics. Finally, section 5 concludes the paper and identifies future perspectives.

2 Related work

In RBAC model, permissions are associated with roles that users have as a part of an organization. Thus, the user's access to resources is decided based on his role. Therefore, a role can be considered as a collection of users that have the same set of permissions. This approach has two principle advantages, on one hand, users will access only to the resources that they require to achieve their tasks, under the suitable mode. On the other hand, the system administration is made easy. However, in the basic RBAC, the access decision will be complex [Covington and Sastry, 2006] and not adequate [Kuhn et al., 2010] when the contextual attributes are required to granting the access. Moreover, the permissions are referring to individual objects. This kind of referring leads to role-permission explosion problem in situations including large number of objects. To resolve these disadvantages, ABAC [Brossard et al., 2017] was proposed. The ABAC model introduces the concept of attribute, hence an ABAC system is composed of three sets of entities: users, resources and the environment. Each of these three entities have specific attributes. An attribute consists of a pair *(key, value)* and the permissions of users depend on their attributes. Even the ABAC was proposed to facilitate the management of security, the proposed solution by ABAC can be as complicated as that of RBAC in some cases [Rajpoot et al., 2015]. According to [Coyne and Weil, 2013], in ABAC the role names are still associated with users, but they are no more considered as collections of permissions. In most systems, there are private objects dedicated to a particular user and where the access is qualified as "unique access" (for example, the report card of a student) versus "multiple access" in the case of shared objects. To restrict the access to these private objects, the two models resolve the situation differently. In fact, RBAC introduces a private role for each student whereas ABAC introduces a private rule for each student. In this case, the system does not benefit from the advantages of the role of RBAC and the attributes of ABAC. Besides this problem, granting a request of a user in both models (RBAC or ABAC) requires to check the user permissions one by one to make decision to grant or deny the access.

According to [Zhang and Wu, 2016], RBAC and ABAC can not be directly applied to IoT because of their limitations. However, the both models still have some advantages that can be exploited in IoT applications. RBAC deals with the distribution problem of competencies where time and location change, while ABAC deals with the dynamic propagation problems of users. The both models RBAC and ABAC have their specific features and they can complement each other. The idea to merge RBAC and ABAC in one model has become an important research topic, in order to acquire advantages of these two models. However, the proposed solutions for merging the both models are still insufficient. Indeed, NIST organization has announced a challenging project to define a new security model [Kuhn et al., 2010] based on the both existing models.

Many researchers have adopted the idea and several propositions are developed. RABAC (Role-centric Attribute- Based Access Control) [Jin et al., 2012] is the first formal hybrid model which proposes an assignment of roles avoiding role-explosion problem. RABAC is an extension of RBAC with permission filtering policy (PFP) which constrains the available set of permissions based on user and object attributes by using Boolean expression (function). According to [Rajpoot et al., 2015], the RABAC approach does not incorporate environment attributes and so that it is not suitable for systems involving frequently changing attributes. The authors in [Rajpoot et al., 2015] combine RBAC and ABAC in one new model AERBAC (Attributes Enhanced Role-Based Access Control), by using contextual information and exploiting the contents of the resources to provide fine-grained access control mechanism. Several works as spatio-temporal RBAC [Kulkarni and Tripathi, 2008] and context-aware RBAC [Nagarajan and Gopalan, 2016b] focus on the merge of access context in RBAC. However, these models suffer from the role-explosion problem (a big number of roles). To deal with this problem, a new spatio-temporal RBAC [Abdunabi et al., 2014] model was proposed by introducing the concept of spatio-temporal zones to abstract location and time into one single entity. In this last model, using zones prevents the creation of new roles when spatio-temporal constraints associated with them change.

3 A new hybrid access control model

In this section, we present our proposed model which is a hybrid model based on both models RBAC and ABAC. The proposed model integrates the multiple accesses as well as the unique access. Before presenting the new proposed model, we start by listing a set of requirements which must be fulfilled by a suitable access control model.

3.1 Requirements for a suitable access control model

To deal optimally with security policies, an access control model is expected to guarantee the following needs.

- Reduce the complexity of the security policy. This requires the reduction of two metrics: *Written Permissions Number (WPN)* and the *evaluated permissions number (EPN)*. The WPN is the total number of the written permissions, by the administrator, to define what a user or a group can or can not do. The EPN is the number of permissions which will be evaluated, by the system, to decide that a user has not the requested permission. In fact, reducing WPN leads to reduce the EPN which makes the auditing in the model easier.

- Use a suitable format of rules or permissions allowing to express the complex granularity of systems without any explosion.
- Use a suitable format of rules or permissions allowing to express the access to private objects.

To achieve the above requirements, we need to consider the following basic ideas in the new proposed model.

- Use the Role concept; thus, divide the users according to their functions.
- Each role has a set of permissions which are expressed in rules.
- In each rule, we express the object, user, and environment features.
- Divide the set of rules according to the access actions. Because there is one rule for each access actions, in each role, the decision if a user has not the requested permission needs to evaluate just one rule in each active role of this user. Hence, the number EPN equals to number of Active Roles (AR): $EPN = AR$.

A model, which considers the above basic requirements, will be able to overcome all access control models requirements and will provide the advantages of ABAC and RBAC. In the next section, we establish a new model in order to meet these required features.

3.2 Principle of the proposed model

To benefit from RBAC, we define a set of roles and each role has its permissions, but rather than considering permissions as a set of permissions referring to individual objects and to one instance of the access action, we divide the permissions assigned to a role according to its access actions. In the RBAC model, if the designer wants to express the fact that a role "admin" can read papers and mails then he must define a set of permissions assigned to this role as: (admin, read, paper1), ... , (admin, read, papern), (admin, read, mail1), ... , (admin, read, mailm). However, in our proposed model, we propose to define an assignment of permissions as: (admin, read, papers and mails). This last assignment is used to express that all objects which the admin can read are papers and mails; hence we collect the objects into sets, according to the access type (read, write, etc) by the role. In each set, we separate the objects dedicated to a particular user (i.e. unique access) from the objects dedicated to multi users (i.e. multi-access) into two subsets. To benefit from ABAC advantages, identification of permissions takes into consideration the different attributes of objects, subjects (users) and environment. By exploiting the attributes of objects, users, environment, and the concept of ABAC rule, we assume that our model overcomes the following problems which face other models: (i) the role permissions explosion, (ii) roles explosion, and (iii) the exponentially augmentation of the groups when the number of object categories increases. The use of the private objects and shared objects to integrate the multi access and the unique access, in the same model, makes the model more "realistic|flexible" and respects the concept of the role in RBAC as well as the concept of attribute in ABAC (i.e. to restrict the access of each user only to his data, we do not need to define private role or specific rule to each user). The assignment of a role to a set of users allows the designer controlling the separation of duties, list of privileges, confidentiality and to use the principle of role inheritance. Figure1 depicts the principle of the proposed model. The constraints and the mechanisms required by the model will be presented in the next section.

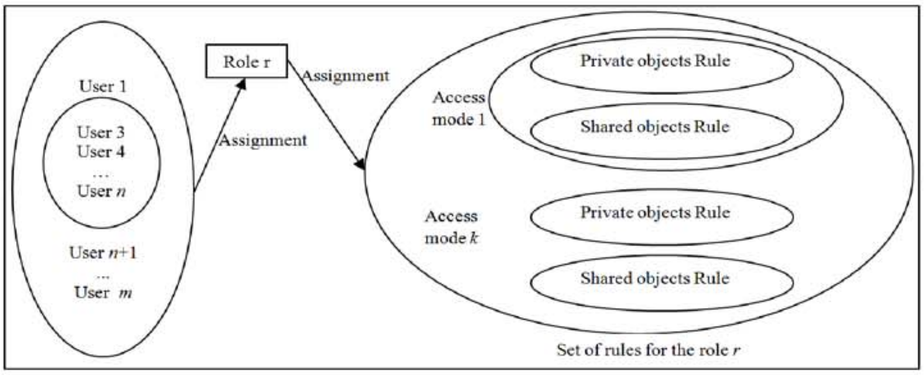

Figure 1: Principle of the proposed model

The Figure 1 illustrates that the permissions of users are assigned to them according to their roles (the same principle as RBAC). Each role has a set of rules defining its permissions (read object, write object, delete object, execute object, etc.). This set of rules is constructed by defining two rules for each access type (read access rules, delete access rules, etc.). The first type of rules concerns the access to shared objects and the second type of rules concerns the access to private objects. This separation between rules makes the model more efficient. Instead of checking the user query using lot of rules, the model checks the user query using only one rule. Indeed, the syntax of rules used in this model allows the designer to divide the permission according to the access type for both object types and takes into consideration the different attributes of objects, subjects (users) and environment.

3.3 The security policy under the proposed model

A policy is a set of rules that define the behaviour of a system. The system that uses this policy is expected to satisfy this set of rules in all its states. In this section, we present the security policy under the proposed model. This policy requires determining sets, functions, rules and constraints. In our proposed model, we have extended basic sets and functions defined in ABAC model.

3.3.1 The Sets

We distinguish five entities in the system: user, object, role, rule, and permission. Each of these entities yields to a specific set in the policy. Thus, the policy defines the following sets.

- S: denotes the set of subjects (users) that can manipulate or access the resources or objects in this system.
- O: denotes the set of objects. Both subjects and objects have their unique identifier u_{id} and r_{id} respectively.
- At: denotes users, resources and environment attributes. The attribute can have a single value (*atomic value*) or multi-atomic value. The set At is composed of several subsets denoted Au_i, Ar_j and Ae. Au_i is the set of $user_i$ attributes, Ar_j is the set of $resource_j$ attributes, and Ae is the set of environment attributes (such as time and location). In the set Ar_j, we use the attribute *Refer_ To* to define the owner of the $resource_j$. The attribute *Refer_ To* contains the value u_{id} if the $resource_j$ belongs to the user (u_{id}) otherwise it contains "null".
- R: denotes the set of roles. The users interact with the system according to their roles.
- Ac: denotes the set of access action (i.e. read, write, view, control, etc.).
- P: denotes the set of permissions.
- RL: denotes the set of rules which assign permissions to each role.

3.3.2 The Functions

Three functions are introduced as follows. (i) $Vu(u,\ a)$: returns the value of the attribute a of the user u, otherwise it returns *null* if the user has not this attribute; (ii) $Vr(rs,\ a)$: returns the value of attribute a for a resource rs, otherwise it returns *null* if the resource has not this attribute; (iii) $Dr(u)$: returns the set of rules dedicated to a user u according to his active role.

3.3.3 The rules

We assume that the set of rules "RL" consists of all role rules subsets RL_r, where $r \in R$ (R is the set of roles). For each role r, the set RL_r is composed of subsets $RL_{r_{acc}}$, such that $acc \in A$ (A is the set of access actions). Lets consider R_m and R_{un} the two sets defined as follows.

- R_m: represents the set of rules which bind to the users the access acc to the multiple access objects.
- R_{un}: represents the set of rules which bind to the users the access acc to the unique access objects.

Each set $RL_{r_{acc}}$ includes one rule from the set R_m and one rule from the set R_{un}.

We use the tuple (acc, rs), containing a resource rs and an access mode acc, to express that a user has the appropriate permission to perform the action acc on the resource rs. The set UP_u of tuples (acc, rs) denotes all the permissions assigned to the user u.

Finally, a rule is a tuple (t, r, acc, cst), such that:

- t: is the type of the rule (unique or multiple). The value of t can be R_{un} or R_m.
- $r \in R$. r is a role.
- acc: access mode, $acc \in A$;
- cst: is a constraint. The constraint cst is a logical formula built upon the two functions $Vu(u, att)$ and $Vr(rs, att)$, such that:
  - $Vu(u, att)$ gives the attribute value of att for the user u.
  - $Vr(rs, att)$ gives the attribute value of att for the resource rs.

  The constraint cst can be written according to the following grammar.

  $cst := true$
  $cst ::= cst\ and \mid or\ Vu(u, att) = Vr(rs, att)$
  $cst ::= cst\ and \mid or\ Vu(u, att1) = Vr(rs, att2)$
  $cst ::= cst\ and \mid or\ Vu(u, att) = const$
  $cst ::= cst\ and \mid or\ Vr(rs, att) = const$
  $cst ::= cst\ and \mid or\ Vu(u, att) \supseteq Vr(rs, att)$

  such that u is a user, rs is a resource, att, $att1$ and $att2$ are attributes, and $const$ is a constant value of an attribute. Besides the above elements, a constraint may include another statements like the time or location (environment attributes). Moreover, if we have tow rules that have the same constraint *cst* and role r with two different access actions acc_1 and acc_1 then we write them in one rule as: $(t, r, acc_1 or acc_2, cst)$.

3.3.4 Constraints in the proposed model

We use the same constraints as defined in RBAC, which are: (i) Static and dynamic separation of duties (SoD), (ii) Role hierarchy, (iii) the cardinality of roles, (iv) role authorization and (v) role execution. After assigning a role to a user, we check the following elements.

- **The role authorization**: is the role authorized for this user?
- **The static separation of duties**: this role is not already assigned to some users who have static conflicts with the current user? this role is not in static conflicts with the already assigned roles to the current user?
- **The cardinality of role**: is the number of users assigned to this role less than the cardinality of the role?

After the verification of these constraints, we add the name of the role to the multi atomic values of the user's "Role-attribute".

- **Dynamic separation of duties**: we use multi atomic value attribute "Active" to express the activated roles by the user. To insert the name of a role into the "Active attribute", we should verify that the role and the user are not in dynamic conflicts. This concerns two cases: (i) this role cannot be activated by the current user because it is already activated by another user who has conflict with the current user, or (ii) this role cannot be activated by the current user because the current user activated already another role which is in conflict with the current role.
- **Role hierarchy**: to give the user a permission, we use the rules of the active role (role execution). If the role r activated by a user contains another role r' (r is a senior role of r') then this user will own the permissions assigned to r' too.

3.3.5 The mechanism

In this section, we present the mechanism of the access decision. It is to decide if a user u requesting access to a resource rs, through the access mode *acc*, is authorized or not to access rs. The mechanism is implemented by two algorithms. The first algorithm evaluates the access query of a user to a specific resource.

The second algorithm evaluates the access query of a user to a set of resources sharing the same attributes (for example, three distinguished resources: "text document", "video", and "audio" which concerns all the same resource). In the second algorithm (which is not presented in the paper), the request is denied if the user does not activate the role (role execution). The algorithm decides if the permission is granted or denied by evaluating the specific rule which is composed of the three following elements.

- The active role (role execution) or its juniors (Role hierarchy).
- The requested access mode.
- The type of the requested objects (unique or multiple).

For space reason, the second algorithm is not presented in this paper.

The Figure 2 summarizes the mechanism of the access decision implemented in the algorithm 1 (i.e., when the user requires access to a specific resource).

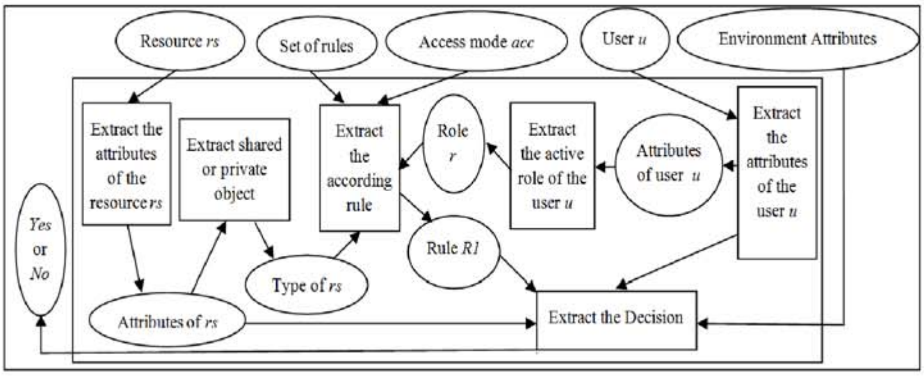

Figure 2: Mechanism of access decision when the user requires access to a specific resource

In the algorithm of multi-access, the request is denied if the user does not activated the role (role execution). The algorithm decides if the permission is granted or denied, by evaluating the tow rules (unique and multiple) which are composed of the following two elements.

- The active role (role execution) or its juniors (Role hierarchy).
- The requested access mode.

The Figure 3 summarizes the mechanism of the access decision when the user requires access to resources sharing the same attributes.

Algorithm 1: algo:1

Input: Access query ($Rq < u_{id}, acc, r_{id} >$) consisting of user identifier u_{id}, access mode *acc* and resource identifier r_{id} ;

Output: Access; // Access = grant if the user has the permission else Access = deny;

```
List_Active_Role ← ∅ // List of the user's currents activated roles.
Access ← deny // The access is denied until we find that the user has the permission.
User_attributes ← get_attributes(u_id) // Gets all user's attributes.
Active_Roles ← getvalue(User_attributes, Active) // From the user attributes, we get the value of the attribute Active, which contains the user's currents activated roles.
if Active_Roles = null then
    return (Request denied: you not have active role)
else
    Resource_attributes ← get_attributes(r_id); //Return the attributes set of the object r_id
    Type ← getvalue(Resource_attributes, Refer_To); //Returns the value of the attribute Refer_To
    if Type = null then
        Type ← shared;
    else
        Type ← unique;
    List_Active_Role.add(Active_Roles);
    List_junior_roles ← get_junior_roles(Active_Role); //Returns juniors of all active roles.
    List_Active_Role.add(List_junior_roles);
    Environment_attributes ← get_att_Environment(); //Returns the set of Environment attributes.
    while (List_Active_Role ≠ ∅) ∧ (Access = deny) do
        Rule ← get_Rule(Role,Type,acc); // Returns the rule dedicated to restrict the access (acc) of the role (Role) to (type) objects.
        Access ←
        evaluate( Rule, Environment_attributes, Resource_attributes,
        User_attributes);//Matches the query with the rule and returns the result.
    if Access = deny then
        return (Request denied);
    else
        return (Request granted);
```

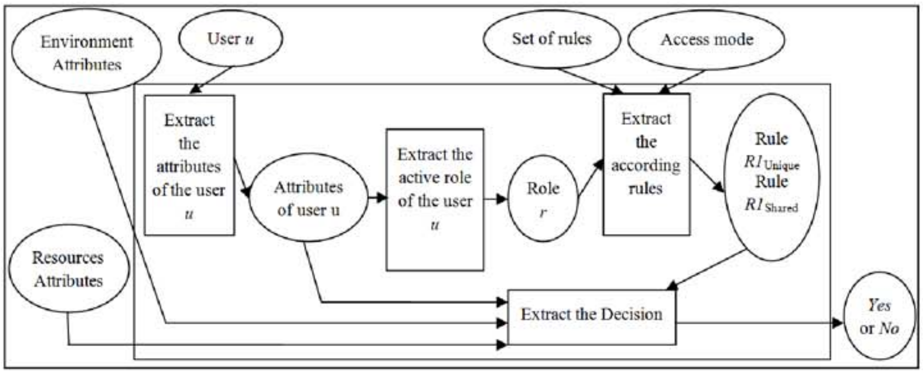

Figure 3: Mechanism of the access decision for multiple resources sharing the same attributes.

4 Evaluation of the new proposed model: empirical comparison with existing models

In order to demonstrate the suitability of the new proposed model, this section provides an empirical comparison approach between this new proposed model and three existing models (RBAC, ABAC and the hybrid model AERBAC).

4.1 Metrics used in the empirical comparison approach

The proposed comparison method is based on the following set of metrics, inspired from [Rajpoot et al., 2015].

- *Written Permissions Number (WPN)*: the total number of the written permissions, by the administrator, to define what a user or group of users can or can not do.
- *Evaluated Permissions Number (EPN)*: it is the number of permissions which will be evaluated, by the system, to decide that the user has not the requested permission.
- *Policy Modification Visualization (PMV)*: it measures how it is hard or easy to visualize the consequences of the policy modification in the access control model.
- *Context-aware Access (CaA)*: it measures if the access control model can handle the dynamic changing of attributes or not.

These four metrics will be evaluated through the policies defined under four models (the new proposed model, RBAC, ABAC and AERBAC), to show how the new proposed model is more suitable than the existing ones. The new proposed model is proved to be more able to handle efficiently security in complicated situation where the number of entities in each set (i.e., users, roles, objects and attributes) increases highly.

4.2 The illustrative example

To make the comparative method easier to understand, we use the following example. In a college, students pass through three levels to get their graduate diploma. In the second level (L_2), students are divided into x specialities ($\{S_i\}_{i \in 1 \dots x}$). In the third level ($L_3$), students are divided into y specialities ($\{S_i\}_{i \in 1 \dots y}$). To manage students' access, the system encloses two kinds of objects: *shared* and *private*. Shared objects are dedicated to a set of users and private objects are dedicated to one user. Two "access actions" are proposed which are *read* and *download*.

The access to shared objects is managed using the following rules: (i) A student in L_1 can access to all courses of his level, (ii) A student in L_2 or L_3 can access only the courses of his speciality, (iii) Only premium users have access to *paid courses*, and (iv) Regular users have access to paid courses only during *promotional periods*. The access to private objects concerns access to marks (i.e., marks are accessible only by the concerned student).

Resources of the system include a set of courses defined in each level L_i, for $i = 1, 2, 3$. These courses are of two kinds, regular courses and paid courses, denoted respectively as RC_{L_i} and PC_{L_i}. In levels L_2 and L_3, courses are divided into specialities. Courses of the speciality S_i for $i = 1 \dots k$ in levels L_2 and L_3 are denoted as $RCL2_{S_i}$, $PCL2_{S_i}$, $RCL3_{S_i}$, $PCL3_{S_i}$, respectively.

Using the previous example, the following sections present a comparative evaluation between the new proposed model and three existing models which are RBAC, ABAC and hybrid model AERBAC. The policy is evaluated under each model to show the advantages of the new proposed model vs the three existing ones.

4.3 RBAC configuration evaluation

The RBAC policy, for the illustrative example, is defined as a set of *roles* and *access permissions*, as follows.

- *Roles*: In a regular users, $1+x+y$ roles are required to express the conditions of levels and specialities. These roles can be denoted as: R_1 for regular students in L_1, $\{R_i\}_{i \in 2 \dots x+1}$ for regular students in L_2, and $\{R_i\}_{i \in x+2 \dots x+1+y}$ for regular students in L_3.

- To express the conditions of premium users, the administrator creates for each regular role a premium role. Hence, the number of roles will be $(1 + x + y) * 2$ roles.

- To express the conditions of promotional periods, the administrator creates for each regular role a promotional role. Hence, the number of roles will be $(1 + x + y) * 3$ roles. A promotional role would be available to users only during promotional periods and it inherits the premium role permissions.

- Access permissions to read regular courses: we need respectively $|RC_{L_1}|$, $|RCL2_{S_1}|$,...$|RCL2_{S_x}|$, $|RCL3_{S_1}|$, ..., $|RCL3_{S_y}|$ permissions for roles R_1, R_2, ... R_x, R_{x+1}, ..., R_{x+1+y}. Each permission has the form (R_i, read, $\{C_j\}_j$), such that R_i is a role and $\{C_j\}_j$ is the set of regular courses accessed by role R_i.

- Access permissions to download regular courses: it is the same as for read permissions. However, a download permission has the form (R_i, download, $\{C_j\}_j$), such that R_i is a role and $\{C_j\}_j$ is the set of regular courses accessed by role R_i.

- Access permissions to read paid courses: we need respectively $|PC_{L_1}|$, $|PCL2_{S_1}|$,...$|PCL2_{S_x}|$, $|PCL3_{S_1}|$,..., $|PCL3_{S_y}|$ permissions for roles R_{x+2+y}, $R_x + 3 + y$, ... $R_2 * x + 1 + y$, $R_{2*x+2+y}$... $R_{2*(x+1+y)}$. Each permission has the form (R_i, read, $\{C_j\}_j$), such that R_i is a role and $\{C_j\}_j$ is the set of paid courses accessed by role R_i.

- Access permissions to download paid courses: The same as for read permission of paid courses. However, a permission has the form (R_i, download, $\{C_j\}_j$), such that R_i is a role and $\{C_j\}_j$ is the set of paid courses accessed by R_i.

- The RBAC does not support access to private objects because this kind of access requires the definition of a new role for each user (i.e., which makes role concept without benefits).

In the following, we analyse each of the four metrics (WPN, EPN, PMV, CaA) separately.

1. *WPN metric*: in RBAC, the configuration that grants permissions to roles is written in the form of direct permissions. Each permission contains an access action and the identifier of an object. We assume that R_i denotes the role identifier, where $1 \leq i \leq N/N$ is the number of roles (in the illustrative example N equals to $3 * (1 + x + y)$). The variable PN_{R_i} denotes the permissions number of the role and it is computed as: $PN_{R_i} = \sum_{acc=1}^{j} NOb_{acc}$, such that j is the number of access actions belonging to the role R_i and

NOb_{acc} is the number of accessible objects by the role R_i through the access action acc.

In the illustrative example, the role R_1 has two access actions (i.e., read ($acc = 1$) and download ($acc = 2$)) and there is RL_1 courses which can be read or downloaded by role R_1. Hence, $j = 2$ and $NOb_1 = RL_1$. The permissions number PN_{R_1} for role R_1 is given as: $PN_{R_1} = \sum_{acc=1}^{2} NOb_{acc} = NOb_1 + NOb_2 = RL_1 + RL_1$. The total number of written permissions WPN is equal to the PN_{R_i} sum; hence, $WPN = \sum_{i=1}^{N} PN_{R_i}$. In fact, we will have: $WPN = 2 * RL1 + x * (2 * RL2) + y * (2 * RL3)$. To simplify the analysis, we suppose that the number of permissions is the same for all roles, hence WPN is computed using the equation 1.

$$WPN = N * PN_{R_i} \tag{1}$$

We study the metric WPN and the number of roles according to five parameters: (i) the number of users, (ii) the number of specialities, (iii) the number of objects in each specialities, (iv) the number of actions in each specialities, (v) and the number of conditions. In total, we have 21 cases. In the first case, all parameters have the value 1. In each case from case 2 to case 21, four parameters are fixed to the value 1 and the fifth parameter is successively affected to the values 10, 100, 1000 and 10000. For example, case 2 affects to the number of users 10, case 3 affects to the number of users 100, case 4 affects to the number of users 1000, case 5 affects to the number of users 10000 (as depicted in Table 1). The same thing is done for the four other parameters. The value of WPN and the number of required roles in each case are plotted in Figure 4.

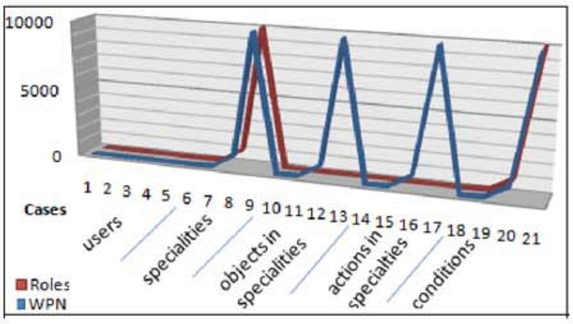

Figure 4: Experimental Results in RBAC

Cases	Users	Specialities	Objects in each Sp	Actions in each Sp	Conditions
1	1	1	1	1	1
2	10	1	1	1	1
3	100	1	1	1	1
4	1000	1	1	1	1
5	10000	1	1	1	1
6	1	10	1	1	1
7	1	100	1	1	1
8	1	1000	1	1	1
9	1	10000	1	1	1
10	1	1	10	1	1
11	1	1	100	1	1
12	1	1	1000	1	1
13	1	1	10000	1	1
14	1	1	1	10	1
15	1	1	1	100	1
16	1	1	1	1000	1
17	1	1	1	10000	1
18	1	1	1	1	10
19	1	1	1	1	100
20	1	1	1	1	1000
21	1	1	1	1	10000

Table 1: The input parameters values for the evaluation of RBAC

From Figure 4, we find that: (i) the number of users has no effect on the WPN (case 2, .., case 5), (ii) There is a WPN explosion on the systems that have large number of objects and complex granularity (Specialities, Actions) (case 7, .., case 21). Hence, RBAC has a lack of expressiveness and does not provide fine-grained access control, (iii) and finally, the roles number increases according to the number of specialities (case 6, ..., case 9).

2. *EPN metric*: To decide that a user has not the requested permission, the RBAC evaluates all the permissions of this user's active roles (i.e., the set AR). So that, the EPN is calculated using equation 2.

$$EPN = \sum_{i=1}^{AR} PNR_i \tag{2}$$

Usually, a user is assigned to a small number of roles. This later means that EPN is not a very big number. In fact, this is correct only if RBAC is used in systems without complex granularity. The previous metric demonstrates that RBAC is not suitable for fine grained systems. The EPN in RBAC indicates that RBAC has not complex auditing.

3. *"Policy modification visualisation" metric*: The policy is written at the role

level; hence, it is easy to visualize the consequences of policy modification. If the administrator adds a permission to a role then all users assigned to this role will have the permission, automatically.

4. *"Context-aware access" metric*: The explosion of WPN in fine grained systems is due to the fact that RBAC does not use the attributes. Models that do not use attributes do not support the context-aware access as the case of RBAC.

4.4 ABAC configuration evaluation

The policy in ABAC is defined as a set of *rules*. According to [Xu and Stoller, 2015], a rule is a tuple (eu, er, O, c) such that eu is a user attribute expression, er is a resource-attribute expression, O is a set of operations and c is a constraint. Therefore, in te case of the illustrative example, ABAC needs to define $2*(x+1+y)$ rules. These rules are required to express the conditions on levels, specialities and access permission, as follows.

- $Rule_1$= ($true$, Role=student ∧ Level=L_1, read or download, type=courses ∧ level=L_1),
- $Rule_2$=($true$, Role=student ∧ Level=L_2 ∧ S = 1, read or download, type =courses ∧ level=L_2 ∧ Speciality=S_1),
  ...
- $Rule_{x+1}$=(Role=student ∧ Level=L_2 ∧ S=x, read or download, type=courses ∧ Level=L_2 ∧ Speciality=S_x),
- $Rule_{x+1+1}$=(Role=student ∧ Level=L_3 ∧ S=1, read or download, type=courses ∧ Level=L_3 ∧ Speciality=S_1),
  ...
- $Rule_{x+1+y}$ =(Role=student ∧ Level=L_3 ∧ S=y, read or download, type=courses ∧ Level=L_3 ∧ Speciality=S_y),
- $Rule_{x+1+y+1}$=($true$, Role=student ∧ Level=L_1 ∧Type= premium∨ today ∈ PromoDates, read or download, type=PaidCourses ∧ level=L_1),
  ...
- $Rule_{2*(x+1+y)}$=(Role=student ∧ Level=L_3 ∧ Sp=y Type= premium ∨ today ∈ PromoDates, read or download, type=PaidCourses ∧ Level=L_3 ∧ Speciality=S_y).

To access to a private object, the "administrator" should write a rule for each user allowing him a unique access to that object. Therefore, if we have

1000 students then the "administrator" needs to rewrite 1000 times the rule ($true$, Title $= u_id$, read or download, Type $=$ Result and ReferTo $= u_id$), such that u_id is the identifier of a user.

In this case, the number of *rules* in ABAC is less than the number of *roles* in RBAC. We are interested to explain this decrease in the number of rules. In the following paragraphs, we study the four metrics (WPN, EPN, PMV, CaA) for ABAC model.

1. *WPN metric*: In ABAC, the configuration that grants permissions to users is written in the form of rules. The number of rules depends on the number of object groups (objects of the same group have the same attributes). As discussed above, there are $2*(x+1+y)$ object sets each of which contains courses of the same type, level and speciality. We denote by NOG the number of object groups, hence the total number of the written permission WPN will be equal to NOG: $WPN = NOG$.

   We will evaluate the WPN according to six parameters: (i) the number of users, (ii) the number of specialities, (iii) the number of objects in each specialities, (iv) the number of actions in each specialities, (v) the number of object conditions, and (vi) the number of environment conditions. We define 25 cases which are similar to the previous section of RBAC. In the first case, all parameters have the value 1. In the other cases, four parameters are fixed to the value 1 and the fifth parameter is successively affected to the values 10, 100, 1000 and 10000. The metric WPN is evaluated in two cases, *with* and *without* private access. The Figure 5 plots the values of WPN in the two cases.

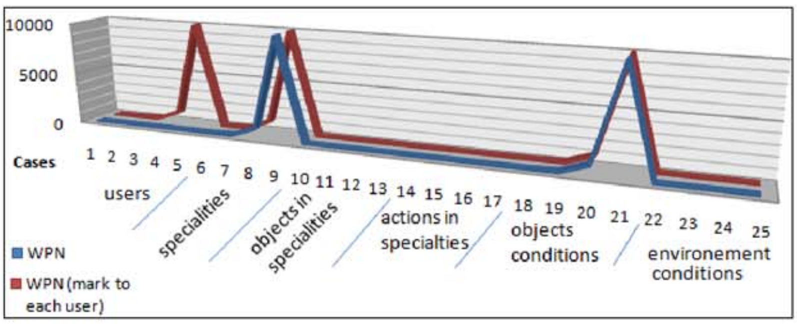

Figure 5: Experimental Results in ABAC

Figure 5 shows that: (i) WPN (in the two cases, with or without private objects) increases when the number of *specialities* or the number of *object* *conditions* increase (Case 9 and case 21). This is justified by the augmentation of NOG required to model the features of objects, (ii) WPN with private objects increases when the *users* number increases (case 5), (iii) finally, the two factors objects number and access actions number have no impact on WPN. Hence, ABAC provides fine-grained access but it still needs some adjustments to support the private objects and to be more suitable for fine_grained systems.

2. *EPN metric*: to decide that the user has not the requested permission, the ABAC will evaluate all the rules with an exhaustive enumeration of attributes, used in each policy rule that we denoted by A_i. Hence, the number EPN is calculated using the equation 3.

$$EPN = \sum_{i=1}^{WPN} A_i \tag{3}$$

Usually, EPN is a very big number which means that ABAC has a complex auditing.

3. *"Policy modification visualization" metric*: in ABAC, it is hard to visualise the consequences of policy modification. If the administrator changes a rule then he will not be able to know all the consequences.

4. *"Context-aware access" metric*: Unlike RBAC, ABAC supports the context-aware access, thanks to the use of the attributes.

4.5 AERBAC configuration evaluation

According to [Rajpoot et al., 2015], the AERBAC policy is defined as a set of roles. Applied to the illustrative example, we require $x + 1 + y$ roles each of which has two access permissions: one with conditions and another without conditions (as described in Table 2). AERBAC does not support the access to private objects (e.g. marks in a courses).

The four metrics are evaluated in the following.

1. *WPN*: The AERBAC creates N roles and each role R_i has a Permission Number PN_{R_i} as in the RBAC case. Hence, the total number of written permissions WPN is equal to the sum of all PN_{R_i}, for $i = 1 \ldots N$. So that, WPN is computed as: $WPN = \sum_{i=1}^{N} PN_{R_i}$.

   Figure 6 plots WPN depending the same set of parameters used in the case of ABAC. Figure 6 shows the following.

Role	Permissions	Conditions
R_1:**Stud**_L_1	(read, (Type(o)=Paid_Course ∧ L(o)=1))	Type(u)=premium ∨ today ∈ PromoDates
	(read, (Type(o)=Course ∧ L(o)=1))	none
....	....	....
R_{1+x+y}:**Stud**_L_3_S_y	(read, (Type(o)=Paid_Course ∧ L(o)=3 ∧ S(o)=y,))	Type(u)=premium ∨ today ∈ PromoDates
	(read, (Type(o)=Course ∧ L(o)=3 ∧ S(o)=y))	none

Table 2: AERBAC configuration

- The number of users, the number of objects and the number of environment conditions have no impact on WPN (as in the case of ABAC).
- The number of access actions has an impact on WPN (as in the case of RBAC).
- The number of specialities and the number of object conditions have an impact on WPN (as in the case of ABAC).

As a conclusion, AERBAC provides fine grained access but it still needs some adjustments to support the private objects and be more suitable in fine grained systems.

2. *EPN and "policy modification visualization" metrics*: These two metrics are similar to the case of RBAC,
3. *Context-aware access metric*: it is similar to the case of ABAC.

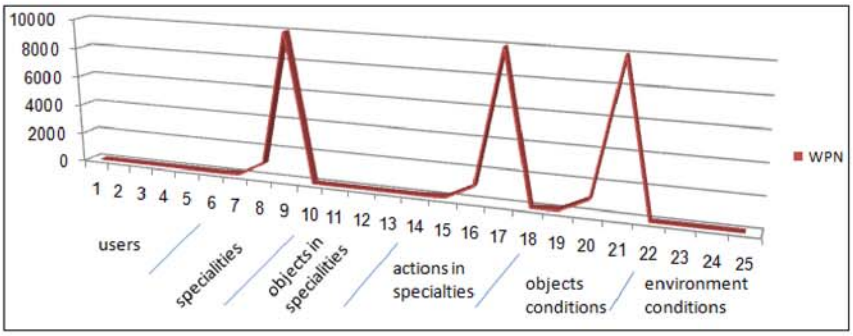

Figure 6: Experimental Results in AERBAC

4.6 Evaluation of the new proposed model

In the new proposed model in this paper, the policy will be defined as a set of roles and a set of access permission rules, as follows.

- *Roles*: Actually, the student in all levels has always the same role which is called "*student*". So that, unlike RBAC case, the new proposed model requires only one role to express the role "*student*".
- *Rules*: According to the format of rules defined in section 3.3, only two rules are required to express all access conditions in the illustrative example. We define *Rule_ 1* to *read* or to *download* the shared objects(free courses and paid courses) and *Rule_ 2*to *read* or *download* the private objects (i.e., marks of a course). Lets denote by PC paid courses, C regular courses and T the type of users (which can be premium or normal) or the type of objects (which can be a mark or a course). Using the previous notations, the model requires only the following two rules to define the policy in the example.
  - *Rule*_1:$\{true, student, [T(o) = C \lor T(o) = PC \land (T(u) = premium \lor today \in PromoDates) \land L(u) = L(o) \land S(u) = S(o)], readordownland\}$
  - *Rule*_2: $\{true, student, T(o) = Note \land Vr(o, Refer\_to) = Vu(u, id)$, *read* or *downland*$\}$

To demonstrate the efficiency of the this new proposed model, we analyse the four metrics in the following paragraphs.

1. *WPN metric*: in the proposed model, the configuration that grants permissions to roles is written in the form of role rules. The number of role rules depends on the number of access actions (in the example, there are 2 access actions). Hence, the WPN is equal to the sum of all PN_{R_i}: $WPN = \sum_{i=1}^{N} PN_{R_i} = TRu$.

   We denote by TRu the total written rules number which represents the WPN metric in the comparative method. PN_{R_i} indicates the rules number of the role and it is computed using equation 4.

$$PN_{R_i} = \sum_{acc=1}^{NAC_{R_i}} NT_{Act_{acc}}. \tag{4}$$

   In equation 4, NAC_{R_i} is the number of access actions belonging to the role R_i. When computing NAC_{R_i}, those access actions which have the same set of accessible objects with the same access conditions are considered as *one action*. For example, the role *student* has two access actions (read and download) which have the same access conditions and the same objects; hence, these two actions are considered as one action when computing PN_{R_i}. $NT_{Act_{acc}}$ indicates the number of "access action types" (this number can be either 1 or 2). In the example, the access action (read/download) has two types (which are: shared and private). Using the illustrative example, we will have the following.

$$PN_{R_1} = \sum_{acc=1}^{1} NT_{Ac_{acc}} = 2. \tag{5}$$

$$WPN = \sum_{i=1}^{1} PN_{R_i} = PN_{R_1} = 2. \tag{6}$$

WPN is computed depending on the same set of parameters used in the evaluation of ABAC and AERBAC. To simplify the analysis, we suppose that the PN_{R_i} of all roles is the same, hence we will have: $WPN = N{*}PN_{R_i}$. We distinguish between two policy cases, case 1 (which is the middle case) and case 2 (which is the worst case).

Case 1:

In this case, we propose that: (i) 50% of access actions have the same objects and conditions sets, (ii) and 50% of access actions have the two access types and 50% of access actions have just shared access. The equation that calculates the PN_{R_i} will be as follows.

$$PN_{R_i} = \sum_{acc=1}^{(NAC_{R_i}/2)+1} NT_{Ac_{acc}} = \sum_{acc=1}^{[(NAC_{R_i}/2)+1]/2} 2 + \sum_{acc=[(NAC_{R_i}/2)+1]/2+1}^{[(NAC_{R_i}/2)+1]} 1. \tag{7}$$

Case 2:

In this case, we propose that: (i) All access actions have not the same objects and conditions sets, (ii) and all access actions have the two access types. So that, $PN_{R_i} = \sum_{acc=1}^{NAC_{R_i}} 2$. The Figure 7 plots the number of roles as well as values of WPN in the two cases, depending on the proposed input parameters. The Figure 7 shows that: (i) the number of users, objects or environment features have no effect on the WPN neither on the roles number, (ii) the WPN number increases, exponentially, in order to model the large number of access actions in the worst case. However, in reality the access actions set is small, (iii) and finally, there is no WPN explosion on the systems that have large number of objects or complex granularity (Specialities). Hence, the

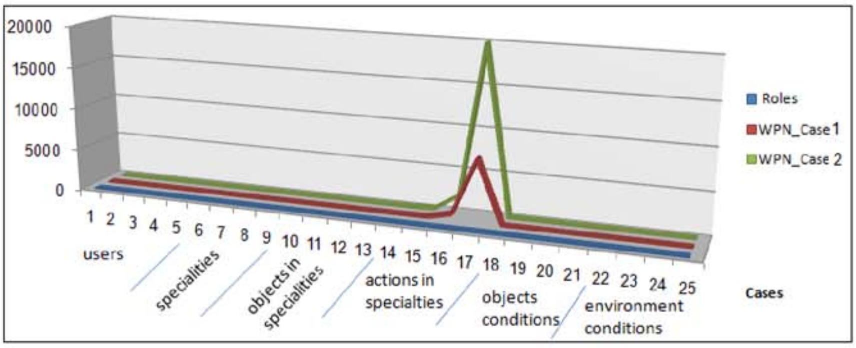

Figure 7: Experimental Results In The Proposed Model

proposed model has a good expressiveness and provides fine-grained access control.

2. *EPN metric* : To decide that the user has not the requested permission, the new proposed model will evaluate at most two rules (one for shared access and one for private access) for each role of this user's active roles (AR). Hence, the number EPN is calculated using: $EPN = \sum_{i=1}^{AR} 2$. Usually, a user is assigned to a small number of roles which means that EPN, so that it has not complex auditing even in high granularity systems.

3. *"Policy modification visualization" metric* and *"Context-aware access" metric*: If the administrator adds a permission to a role then all users assigned to this role will have the permission automatically. The model supports context-aware access.

5 Conclusion

Existing access control models (RBAC, ABAC and AERBAC) suffers from several problems when dealing with complicated security policies in complicated systems. These previous models have several drawbacks such as: explosion in the number of roles and rules, problems with context-awareness, problems with the visualisation of policies update, etc. These drawbacks make these models enable to provide scalability, flexibility, and fine granularity in cloud and IoT environments. To handle these drawbacks, this paper has proposed a new access control model which combines and extends, basically, the two models RBAC and ABAC. In order to demonstrate the advantages of the new proposed model, an empirical study is realised. In this study, the new proposed model is compared versus three existing models based on specific metrics. The results demonstrate that the new proposed model is more suitable than existing ones. A complete validation through simulation and formal verification is planed in future work.

References

[Aaltonen and Laarni, 2017] Aaltonen, I. and Laarni, J. (2017). Field evaluation of a wearable multimodal soldier navigation system. *Applied Ergonomics*, 63:79–90.

[Abdunabi et al., 2014] Abdunabi, R., Sun, W., and Ray, I. (2014). Enforcing spatio-temporal access control in mobile applications. *Computing*, 96(4):313–353.

[Alghabban et al., 2017] Alghabban, W. G., Salama, R. M., and Altalhi, A. H. (2017). Mobile cloud computing: An effective multimodal interface tool for students with dyslexia. *Computers in Human Behavior*, 75:160–166.

[Brossard et al., 2017] Brossard, D., Gebel, G., and Berg, M. (2017). A systematic approach to implementing abac. In *Proceedings of the 2nd ACM Workshop on Attribute-Based Access Control*, pages 53–59. ACM.

[Chen et al., 2016] Chen, B.-C., Yang, C.-T., Yeh, H.-T., and Lin, C.-C. (2016). Mutual authentication protocol for role-based access control using mobile rfid. *Applied Sciences*, 6(8):215.

[Covington and Sastry, 2006] Covington, M. J. and Sastry, M. R. (2006). A contextual attribute-based access control model. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 1996–2006. Springer.

[Coyne and Weil, 2013] Coyne, E. and Weil, T. R. (2013). Abac and rbac: scalable, flexible, and auditable access management. *IT Professional*, 15(3):14–16.

[Ferraiolo et al., 2001] Ferraiolo, D. F., Sandhu, R., Gavrila, S., Kuhn, D. R., and Chandramouli, R. (2001). Proposed nist standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274.

[Hasiba et al., 2017] Hasiba, B. A., Kahloul, L., and Saber, B. (2017). A new hybrid access control model for multi-domain systems. In *4th International Conference on Control, Decision and Information Technologies, CoDIT 2017, Barcelona, Spain, April 5-7, 2017*, pages 766–771.

[Hsu and Ray, 2016] Hsu, A. C. and Ray, I. (2016). Specification and enforcement of location-aware attribute-based access control for online social networks. In *Proceedings of the 2016 ACM International Workshop on Attribute Based Access Control*, pages 25–34. ACM.

[Jin et al., 2012] Jin, X., Sandhu, R., and Krishnan, R. (2012). Rabac: role-centric attribute-based access control. *Computer Network Security*, pages 84–96.

[Kuhn et al., 2010] Kuhn, D. R., Coyne, E. J., and Weil, T. R. (2010). Adding attributes to role-based access control. *Computer*, 43(6):79–81.

[Kulkarni and Tripathi, 2008] Kulkarni, D. and Tripathi, A. (2008). Context-aware role-based access control in pervasive computing systems. In *Proceedings of the 13th ACM symposium on Access control models and technologies*, pages 113–122. ACM.

[Le et al., 2014] Le, X. H., Doll, T., Barbosu, M., Luque, A., and Wang, D. (2014). Evaluation of an enhanced role-based access control model to manage information access in collaborative processes for a statewide clinical education program. *Journal of biomedical informatics*, 50:184–195.

[Li et al., 2014] Li, F., Rahulamathavan, Y., and Rajarajan, M. (2014). Lsd-abac: Lightweight static and dynamic attributes based access control scheme for secure data access in mobile environment. In *Local Computer Networks (LCN), 2014 IEEE 39th Conference on*, pages 354–361. IEEE.

[Liu et al., 2015] Liu, K., Zhou, Z., Chen, Q., and Yang, X. (2015). Towards a rbac workflow model for thesis management. *JSW*, 10(4):480–490.

[Luo et al., 2016] Luo, J., Wang, H., Gong, X., and Li, T. (2016). A novel role-based access control model in cloud environments. *International Journal of Computational Intelligence Systems*, 9(1):1–9.

[Markku et al., 2015] Markku, T., Daniel, S., Klaus-Peter, E., Thomas, O., Dirk, S.-W., and Andrés, L. (2015). Interaction and humans in internet of things. In *INTERACT 2015: Human-Computer Interaction - INTERACT 2015*, pages 633–636. Springer.

[Moon Sun Shin and Jeong, 2015] Moon Sun Shin, Heung Seok Jeon, Y. W. J. B. J. L. and Jeong, S.-P. (2015). Constructing rbac based security model in u-healthcare service platform. *The Scientific World Journal*, pages 1–13.

[Mukherjee et al., 2017] Mukherjee, S., Ray, I., Ray, I., Shirazi, H., Ong, T., and Kahn, M. G. (2017). Attribute based access control for healthcare resources. In *Proceedings of the 2Nd ACM Workshop on Attribute-Based Access Control*, ABAC '17, pages 29–40, New York, NY, USA. ACM.

[Nagarajan and Gopalan, 2016a] Nagarajan, S. and Gopalan, N. (2016a). A dynamic context aware role based access control secure user authentication algorithm for wireless networks. *International Journal of Applied Engineering Research*, 11(6):4141–4143.

[Nagarajan and Gopalan, 2016b] Nagarajan, S. and Gopalan, N. (2016b). A dynamic context aware role based access control secure user authentication algorithm for wireless networks. *International Journal of Applied Engineering Research*, 11(6):4141–4143.

[Ngo et al., 2016] Ngo, C., Demchenko, Y., and de Laat, C. (2016). Multi-tenant attribute-based access control for cloud infrastructure services. *Journal of Information Security and Applications*, 27:65–84.

[Pang and Zhang, 2015] Pang, J. and Zhang, Y. (2015). A new access control scheme for facebook-style social networks. *Computers & Security*, 54:44–59.

[Qiu et al., 2016] Qiu, M., Gai, K., Thuraisingham, B., Tao, L., and Zhao, H. (2016). Proactive user-centric secure data scheme using attribute-based semantic access controls for mobile clouds in financial industry. *Future Generation Computer Systems*.

[Rajpoot et al., 2015] Rajpoot, Q. M., Jensen, C. D., and Krishnan, R. (2015). Attributes enhanced role-based access control model. In *International Conference on Trust and Privacy in Digital Business*, pages 3–17. Springer.

[Ranchal et al., 2016] Ranchal, R., Bhargava, B., Fernando, R., Lei, H., and Jin, Z. (2016). Privacy preserving access control in service-oriented architecture. In *Web Services (ICWS), 2016 IEEE International Conference on*, pages 412–419. IEEE.

[Reinschluessel et al., 2017] Reinschluessel, A. V., Teuber, J., Herrlich, M., Bissel, J., van Eikeren, M., Ganser, J., Koeller, F., Kollasch, F., Mildner, T., Raimondo, L., et al. (2017). Virtual reality for user-centered design and evaluation of touch-free interaction techniques for navigating medical images in the operating room. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 2001–2009. ACM.

[Varadharajan et al., 2015] Varadharajan, V., Amid, A., and Rai, S. (2015). Policy based role centric attribute based access control model policy rc-abac. In *Computing and Network Communications (CoCoNet), 2015 International Conference on*, pages 427–432. IEEE.

[Xu and Stoller, 2015] Xu, Z. and Stoller, S. D. (2015). Mining attribute-based access control policies. *IEEE Transactions on Dependable and Secure Computing*, 12(5):533–545.

[Zhang and Wu, 2016] Zhang, Y. and Wu, X. (2016). Access control in internet of things: A survey. *arXiv preprint arXiv:1610.01065*.

[Zhang and Zhang, 2017] Zhang, Y. and Zhang, B. (2017). A new testing method for xacml 3.0 policy based on abac and data flow. In *Control & Automation (ICCA), 2017 13th IEEE International Conference on*, pages 160–164. IEEE.