Extended RBAC with Context-Based Role-Recommendation

Gang Liu, Shaomin Ji, Runnan Zhang, Guofang Zhang, Lu Fang, and Quan Wang

***Abstract*—The user authorization query (UAQ) problem determines whether there exist an optimum set of roles to be activated to provide a set of permissions requested by a user. It has been deemed as a key issue for efficiently handling user's access request. The cardinality and dynamic separation of duty constraints make the issue more complex in role based access control (RBAC). There are many researches dedicate to analyze the computational complexity of the UAQ problem and try to solve this problem using algorithms. However, all of these researches only consider the dynamic separation of duties, role-cardinality and permission-cardinality constraints, and they do not pay attention to the context constraints. The UAQ problem is raised in the web environment, there is always a lot of context information, and thus this paper intends to use the context to propose a general model to solve the UAQ problem in the web environment.**

***Index Terms*—Role-based access control, user authorization query, context, fine-grained.**

I. Introduction

A recent study [1]shows that the role-based access control (RBAC) has become the most popular access control model. RBAC is simple, which reflects organizational structure, and it is easy to administer and review. In the RBAC users are assigned to roles and roles are assigned to permissions, thereby users are granted permissions through role activation. This simple user authorization model in RBAC is sufficient in well-organized systems (e.g. in a company), since a user is typically assigned to a small number of roles. However, this simple user authorization model cannot address more fine-grained access requests. Today, with the development of internet, the system often consists of hundreds of roles and users in the interconnected and collaborative system, and it is difficult to find the set of roles requested by the user.

There are many researches on the user authorization query (UAQ) problem [2]-[7], including the complexity analysis of the UAQ problem and approaches to solve UAQ problem. The UAQ problem's definition [2] is that in the web environment, the number of roles in the system is large, users do not know which roles they are assigned to, and users only request permission set without activating roles, according to the permission set requested by the user, the system needs to find the role set to activate, and the role set must meet certain conditions. The conditions here not only mean that the role set must contain the permission set requested by the user but also include the dynamic separation of duties, hybrid hierarchy [2], role-cardinality, permission-cardinality [3] and other constraints, which make the problem intractable. These researches use various algorithms to solve this problem, which includes greedy algorithm [2], depth search algorithm [3], and weight algorithm [7] and so on, but neither can solve this problem well.

In the web environment, especially in the interconnected and collaborative systems, access control policies always need to consider context constraints, such as location or time. In contrast to the restrictions mentioned above, context restrictions are the most common constraints in the interconnected system. Here are a lot of researches on the context in the interconnected environment [8]-[10], from which we can see that in the web environment, the context of the application is ubiquitous. As far as we know, no research has been devoted to exploiting the context to solve the UAQ problem.

In this paper, we proposed a context-based role-recommendation (R^3BAC) model based on RBAC model, which uses context condition to filter roles so that it can significantly reduce the range of effective roles and solve the UAQ problem fundamentally, because the precondition of UAQ is assuming there are a large number of roles in the web environment [2]. The R^3BAC model is not only a solution to the UAQ problem, but also to solve the more general problem-role activation problem. In the web environment, there are often hundreds of roles in the system to achieve fine-grained access control, the user or system want to determine an optimum set of roles to be activated to provide a particular set of permissions requested by user is very difficult and time-consuming. In this paper, we define this problem as the role activation problem. The difference between the two problems is that the UAQ problem assumes that the user requests permission set and the system activate the role set according to the permission set requested by user, in the role activation problem, we do not make such assumption because we think the user also needs to be responsible for

Manuscript received February 5, 2018; revised May 10, 2018. This work is supported by the National Natural Science Foundation (NNSF) of China (Grant No. 61572385).

Gang Liu, Shaomin Ji, Runnan Zhang, Guofang Zhang, Lu Fang and Quan Wang are with School of Computer Science and Technology, XIDIAN University, Xi'an, China (e-mail: gliu_xd@163.com, 865332046@qq.com, zhangrunnan_xd@qq.com, zgf356653246@163.com, l_silence104@163.com, qwang_xd@163.com).

determining if the role set to be activated to provide the permission set requested by themselves. In simple terms, the role activation problem includes the UAQ problem. The proposed R^{3}BAC model can filter roles and efficiently locate the role set which contains the permission set requested by the user based on context, the specific model we will describe later.

The paper is organized as follows: Section 2 discusses the context condition briefly. Section 3 develops R^{3}BAC model along with its reference model and functional specifications. Section 4 defines the XACML profile for R^{3}BAC and presents an implementation example. Section 5 concludes the paper.

II. Context Condition

In the web environment, in order to achieve fine-grained model of access control, security administrators often set context restrictions on the roles, and we will use these restrictions to filter roles to solve the role activation problem. In this paper, we will call these context restrictions as context condition uniformly, and the description of the specific context condition is given in this chapter

A. Context

The context has been applied to many areas, such as assisted living [11], hospital information systems [12], tour guides [13], and smart environments [14]. At the same time, using the context to improve the security performance of various applications has become an important work of researchers. Several researchers have developed RBAC models that support context-based access control [15]-[17].

Context is an elusive concept which has many different meanings to different people and communities. There are a lot of context definitions，Anind K. Dey and Gregory D. Abowd [18] refer to context as: “Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.” In a similar definition, Gustaf Neumann and Mark Strembeck [15] define context as: “context in general may consist of almost every available information that describes a specific situation”. Ryan *et al.* [19] define context as the user’s location, environment, identity and time.

The specific context definition is outside the scope of the research. This paper chooses the context defined by Schilit et al. [20] as the context concept in this paper.

B. Context Condition

In order to accurately describe the context condition and apply the context to the R^{3}BAC model, this research introduces the concept of context condition. The context condition is an abstract concept, and it describes the constraint on the role activation. A context condition is defined through the terms context attribute, context function, and context clause:

- A context attribute (catt) represents a certain attribute of the environment whose actual value might change dynamically (e.g. time), or which varies for different instances of the same abstract entity (e.g. location, birthday, or nationality).
- A context function (cfun) is a mechanism to obtain the current value of a specific context attribute (e.g. Date() could be defined to return the current date). The return type of context function can be an integer, a date, a string, and so on.
- A context clause (ccla) is a predicate (a Boolean function) that compares the current value of context attribute with a predefined constant (like Date()==”2016.10.1”) or compares the different context attributes(like Score()>AverageScore()). The range of a context clause is {true, false}.
- A context condition (cond) contains a combination of one or more context clauses (like Birthday (subjectId)==2008.8.8&Nationality(subject)==China).It returns true when all of the context clause hold. Otherwise it returns false.

III. R^{3}BAC Model

In this chapter, we give the description of the R^{3}BAC model, and show how the model filter the roles based on the context conditions. Section 3.1 introduces the R^{3}BAC reference model. Section 3.2 introduces the R^{3}BAC functional model. The R^{3}BAC reference model provides an exact definition of R^{3}BAC basic elements and relations. R^{3}BAC functional model provides an overview of the R^{3}BAC functional specification in four aspects: administrative functions, supporting system functions, review functions and Role Filter Process function.

A. *R^{3}BAC Reference Model*

R^{3}BAC model element sets and their relations are defined in Fig. 1. It includes the basic RBAC model and some new elements and relations.

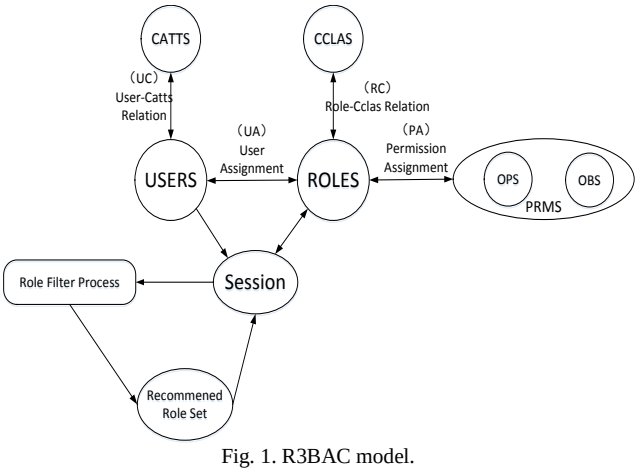


Fig. 1. R3BAC model.

In the R^{3}BAC model, the context attributes(CATTS) are

assigned to the user based on the role sets assigned to the user, and these attributes can be used as monitor for user's context. Fig. 1 illustrates USER-CATT(UC) relation. The arrows indicate a many-to-many relationship (e.g. a user can be assigned to one or more CATTS, and a CATT can be assigned to one or more users). This arrangement provides great flexibility and granularity of CATTs assignment to users. Table I lists context attribute assignments to three persons as example.

TABLE I: ASSIGNMENT OF CONTEXT ATTRIBUTES

	Address	Time	Temperature	Noise	Network Capacity
Alice	×	✓	×	✓	×
Bob	✓	×	×	×	×
Jack	×	×	✓	×	×

This paper uses context clause (CCLAS) to describe the single constraints on the role activation, and the examples are shown in Table II. As shown in Fig. 1, in general, the context clauses and roles are a many-to-many assignment relation, the examples are shown in Table III(e.g. $ccla_1$ and $ccla_4$ are assigned to role_student). The user must meet all the context clauses assigned to the role when he/she activates it.

TABLE II: CONTEXT CONDITIONS

	Address	Time	Temperature	Network Capacity
$ccla_1$	201			
$ccla_2$		09:00≤&&≤12:00		
$ccla_3$			≥36.5℃	
$ccla_4$				≥10G/S

TABLE III: THE ASSIGNMENT OF CONTEXT CONDITIONS

	$ccla_1$	$ccla_2$	$ccla_3$	$ccla_4$
role_student	✓	×	×	✓
role_programmer	×	✓	×	×
role_experimenter	×	×	✓	×

Security administrator establishes role-ccla relations when creating roles, and the user-catt relations is established after assigning user to a roles (For example, assigning user to a role_student role, and according to Table II and Table III, the address attribute and network attribute are also assigned to the user automatically).When the context attributes are assigned to users, then the system can obtain the users' context information, and these information are used to evaluate context condition assigned to the specific role.

The above only describes the R^{3}BAC model elements and their relations, and the specific role filter process is done by the Role Filter Process (RFP) and Recommended Role Set (RSS). The RFP and RSS components are all related to the authorization process. The following figure illustrates these two components' function through comparing the authorization process of the RBAC and R^{3}BAC model, and presents how the R^{3}BAC model filter roles to solve the role activation problem. The authorization process in RBAC is shown in Fig. 2.

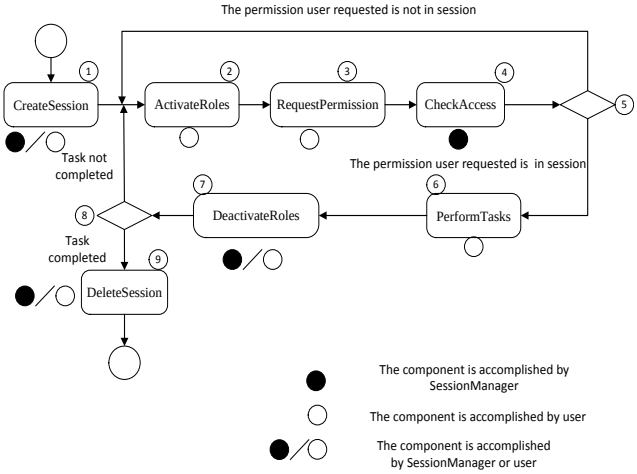


Fig. 2. The authorization process in RBAC.

- *CreateSession*. In NIST RBAC [21], the session is defined as: each session is a mapping of one user to possibly many roles, that is, a user establishes a session during which the user activates some subset of roles that he or she is assigned.
- *ActivateRoles*. *ActivateRoles* operation is a key feature of RBAC which distinguishes it from other access control models. The user selects the appropriate role to activate, and gets the permissions assigned to the role. The *ActivateRoles* operation ensures that only a part of the roles is active, guarantees the least privilege principle, and increases the security of the system. In well-organized systems (e.g. in a company), since a user is typically assigned to a small number of roles, he knows which role should be activated before request permissions.
- *RequestPermission* and *CheckAccess*. Like other access control models, *RequestPermission* and *CheckAccess* are required in the RBAC model. After user activates roles, he/she request permissions to perform task, and the system check whether the permissions is contained in the activated roles.
- *PerformTasks* and *DeactivateRoles*. If the permissions user requested is in the session, the user begins to perform the task. If the permission set isn't in the session, the user returns to the *ActivateRoles*. When the task is done, the user needs to deactivate the role timely to ensure the principle of least privilege. In the same session, the user may request one or more permissions to complete the task, and these permissions may be assigned to different roles, so after the user deactivating the roles, he or she must check whether the task is completed, if not then the user returns to *ActivateRoles* section to continue to complete the task.

- *DeleteSession*. *DeleteSession* releases the resource that is opened in the session, including some roles that the user doesn't deactivate.

As mentioned above, in the original RBAC model, the user activates the role to get the requested permissions. In the normal environment, the user is typically assigned to a small number of roles and he/she knows which roles are assigned to him/her, thus users can activate the role set on their own; however in the network environment, there are often hundreds of roles in the system, and it is difficult to find the role set to be activated to provide a particular set of permission requested by a user, and in a session, the user may need to constantly request permission, which make the role activation problem intractable, thus the original RBAC model cannot handle this situation.

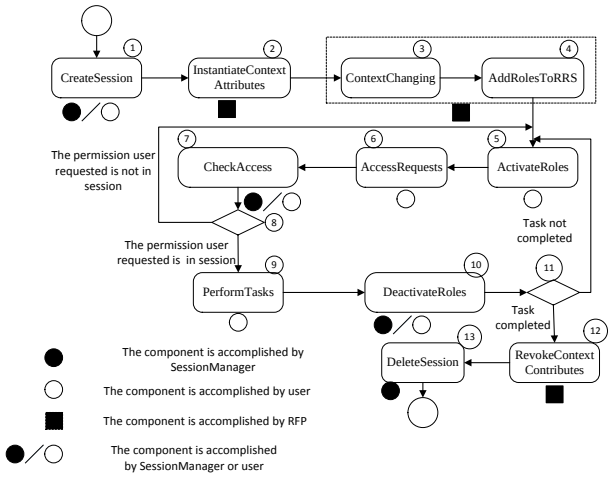


Fig. 3. The authorization process in R3BAC.

In order to solve the problem, we propose R^{3}BAC model to solve the problem. The authorization process of the R^{3}BAC model is listed in Fig. 3. Here the research only lists differences of components between the R^{3}BAC model and the RBAC model.

- *InstantiateContextAttributes*. When a user creates a session, the RFP instantiates the context attributes assigned to the user, where instantiation means that all the context attributes assigned to the user are monitored in real time (e.g. after Bob creates a session, the system will monitor his location).
- *ContextChanging* and *AddRolesToRRS*. The *ContextChanging* is not a real module that needs to be executed by the SessionManager, the user or the RFP components. In general, the user performs the task with changes in the context especially in the interconnected and collaborative environment. In order to illustrate clearly, this paper draws this module separately. When the RFP detects a change in the user's context attributes, it evaluates the context condition assigned to the role based on the context, and adds the role whose context condition's value is true to the RRS, adding the role whose context condition's value is true to the RRS is also called recommendation. This process is repeated during the session, although we put this module at the beginning. The specific RFP function is given in the following.
- *ActivateRoles*. One of the main differences between role activation in R^{3}BAC and RBAC is that the users need to find roles to activate in RBAC, while in R^{3}BAC model, the users or the system only need to find roles in RRS, and the number of roles in the RRS is significantly less than the number of roles in the system, because in current environment, the number of roles that satisfy the context condition will be very small. Thus, the user or the systems' workload of activating the role has been reduced a lot, and the role activation problem will be solved. When the required role is found in RRS, the user or the system adds the role to the session.
- *RevokeContextContributes*. At the end of the session, not only the remaining activated roles need to be deactivated, but also the instantiated context attributes need to be revoked, in other words, the RFP does not monitor the user's context in real time any more.

The above is the authorization process in R^{3}BAC: it uses context conditions to filter the role and adds the roles which satisfy the context condition to the RRS, in this way, the range of roles will be significantly reduced. In this paper, we do not intend to use algorithms to solve the UAQ problem. Instead, we propose a more general model based on the RBAC model to solve the problem of role activation under the web environment, so that users or systems can quickly and effectively locate the role set which provide the permission set user request.

B. R^{3}BAC Functional Description

In this section, this paper presents the functional description of R^{3}BAC for each component defined in the previous section. At the end of this section, the research shows the Role Filter Process function.

C. Administrative Functions

Administrative Functions are used to create and maintain element sets and relations. The basic elements in the R^{3}BAC model are ccla and catt. The administrator creates and deletes ccla/catt, and establishes the relation between ccla/role and catt/user. Administrative functions for creating ccla and deleting ccla are CreatCclas and DeleteCclas. The management functions for catt are CreateCatts and DeleteCatts. New relations in R^{3}BAC are role-to-ccla (RC) and user-to-catt (UC). The functions for creating and deleting RC are AssignCcla and DeassignCcla. For UC, the corresponding functions are AssignCatt and DeassignCatt.

D. Supporting System Functions

Supporting System Functions are used for session management and access control decisions. The supporting system functions in the R^{3}BAC model are almost the same as those in the NIST RBAC. The *AddActiveRole* function in NIST RBAC is used to activate roles, and its parameter space is the roles assigned to the user. Corresponding to

the R[3]BAC model, the parameter space is the roles in the RRS.

E. Review Functions

When the role-to-ccla relations and the user-to-catt relations are created, it should be possible to view the contents of those relations from ccla, user, catt and role perspectives. For example, for the RC relations, the administrator should be able to view all the roles associated with a ccla, and all of the cclas assigned to a role. At the same time the user can view the roles in the RRS attached to the session. The following are new review functions for the R[3]BAC model on the basis of the NIST RBAC model.

Assigned_cclas(r:ROLES):returns all cclas assigned the role;

*Assigned_roles(c:CCLAS):*returns all roles assigned the ccla;

Assigned_users(c:CATTS): returns all users assigned the catt;

Assigned_catts(u: USERS): returns all catts assigned the user;

*RRSroles(se:SESSIONS):*returns all roles in the RRS;

Role Filter Process Function.

Role Filter Process filters roles by the detection of the contexts in real time. The process of Role Filter Process is shown in Fig. 4, and it involves four functions:

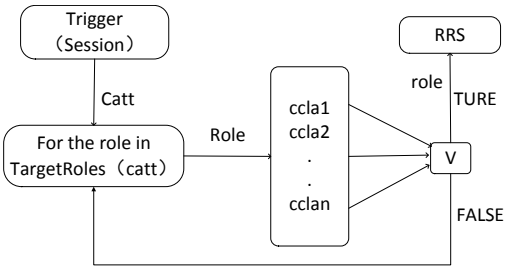


Fig. 4. Role filter process.

Trigger(se: SESSIONS):return the catt whose value is changed;

TargetRoles(ca: CATTS):return the set of catt-related roles;

cond.value(se:SESSIONS): calculates the true value of all the cclas assigned to the role according to the context of the current session;

Addroles(r: roles, RRS): add the filtered role to RRS;

The complete pseudo-code is shown in Fig. 5:

```
For each catt∈Trigger（session）
  For each role in Targetroles（catt）
    For each ccla in Assigned_cclas（role）
      If cond.value（se）=TURE
        AddRoles(role, RRS);
      End If;
      End For;
  End For;
Return RRS;
End For;
```

Fig. 5. The function of role filter process.

IV. XACML Profile for R[3]BAC

XACML is a general access control policy language for determining request/response and a framework for implementing authorization policies. Because of its reputation, considerable work has been done for XACML to implementing RBAC. XACML profile for RBAC [22] has been defined to guide the implementation of RBAC via XACML. In order to show the universality of the model, this research illustrates that R[3]BAC can be easily implemented in XACML. Consistent with the description of the previous R[3]BAC model, this paper proposes a XACML profile for R[3]BAC based on that for RBAC and gives a specific implementation example for this profile.

A. Proposed Profile

The XACML profile for R[3]BAC in this paper does not change the original XACML profile for RBAC. The paper only discusses those new components. The R[3]BAC profile is guided by the following.

- The role filtering process should be performed by an entity rather than the XACML PDP, where this paper refers this entity as the Role Filter Process entity. The Role Filter Process entity filters the roles after the context changes according to the Role Filter policies. These Role Filter policies are a different policy file from the Role<PolicySet> and Permission<PolicySet> files. Role Filter policies are to be used only when the XACML request comes from a Role Filter Process entity.
- The paper [22] includes four policy files: Role<PolicySet>, Permission<PolicySet>, Separation of Duty<PolicySet> and Role Assignment<PolicySet>, in practice, these four policy files and Role Filter Policy files are used by different entities, the order in which they are used is also different. Separation of Duty<PolicySet> and Role Assignment<PolicySet> will be used first when the system is initialized and the user registered. The roles in the session are dynamically changed, so the Role<PolicySet> and Permission<PolicySet> may be used at any time during the session lifetime. The Role Filter Policy file is used continuously during the session lifetime as well as the Role<PolicySet> and Permission<PolicySet>, and the Role Filter Policy files is used in advance of the Role <PolicySet> and Permission <PolicySet>.

B. An Illustrative Scenario

This paper uses the BYOD scenario to illustrate our model. An enterprise employee owns a mobile device, such as a mobile phone, which has two types of applications: one is the employee's own private application and the other is the enterprise office application. Here are two enterprise applications: one is the word application, the other is the video application, where the word application is used for enterprise day-to-day office, and the video application is the learning tool provided by the enterprise for employees to learn the professional information. Enterprise Security Administrators have

developed security policies for the both application: the word application can only be used inside the company and the video application can't be used with private screenshot application at the same time. Here we treat the application as a role. There always has a lot of this kind of enterprise application in the enterprise, for the sake of convenience, we only cite two. In this situation, when we use the traditional RBAC model to achieve access control will encounter the role activation problems, because the number of roles is large, it is difficult to determine whether there exists an optimum set of roles to be activated to provide a particular set of permissions requested by a user. It would be a lot simpler to take advantage of the model presented in this paper: mobile devices will get the user's context information in real time and will validate or invalid the applications based on the current context-visible or invisible to the user. In this way, we can easily determine the role set that provide a particular set of permissions requested by a user.

Here we use the XACML component defined above to represent the role filter policies, for convenience, we only cite the word office application example. Suppose here's an employee named Bob, who is assigned the role_word role. The XACML code is shown in Fig. 6:

```
<PolicySet>
<Target> <Subject>
<SubjectMatch MatchId="&function;string-equal">
<AttributeValue DataType="&xml;string">Bob</AttributeValue>
<SubjectAttributeDesignator
AttributeId="urn:someapp:attributes:name"
DataType="&xml;string"/>
</SubjectMatch></Subject>
<Resource>
<ResourceMatch MatchId="&function;string-equal">
<AttributeValue DataType="&xml;string"> role_wordapp
</AttributeValue>
<ResourceAttributeDesignator
AttributeId="urn:someapp:attributes:role"
DataType="&xml;string"/>
</ResourceMatch></Resource>
<Action>
<ActionMatch MatchId="&function;string-equal">
<AttributeValue
DataType="&xml;string">recommend</AttributeValue>
<ActionAttributeDesignator
AttributeId="urn:someapp:attributes:action-id"
DataType="&xml;string"/>
</ActionMatch></Action>
</Target>
```

Fig. 6. RoleFilter <PolicySet> policy file in example.

V. Conclusion

In this paper, we analyzed the UAQ problem and raised the idea that context should be considered in authorization query, especially in the interconnected and interactive environments. Then the paper proposes the R^{3}BAC model and in this model the system filters roles for the user by monitoring the context of the user in real time. In this way, the number of effective roles will be greatly reduced in the current environment, thus fundamentally solve the role activation problem. The paper illustrates the authorization process of the RBAC model, and gives the authorization process of the R^{3}BAC model. By comparing the differences, the research explains how the R^{3}BAC model filters roles based on context condition. The paper also extends the functional specification of the R^{3}BAC model. Functions provide developers with the design of flexibility and the ability to incorporate additional features to meet the needs of users. In the end the paper also presents the XACML implementation of the R^{3}BAC model with practical examples.

References

[1] R. J. Loomis and A. C. O'Connor, "2010 Economic analysis of role-based access control," *Final Report, Nist*, 2010.

[2] Y, Zhang and J. B. D. Joshi, "UAQ: A framework for user authorization query processing in RBAC extended with hybrid hierarchy and constraints," *ACM Symposium on Access Control MODELS and Technologies* ACM, 2008, pp. 83-92.

[3] J. Lu, J. B. Joshi, L. Jin, and Y. Liu, "Towards complexity analysis of user authorization query problem in RBAC," *Computers & Security*, vol. 48, pp. 116-130, 2015.

[4] J. Lu, Y. Xin, Z. Zhang, H. Peng, and J. Han, "Supporting user authorization queries in RBAC systems by role-permission reassignment," *Future Generation Computer Systems*, pp. 468-476, 2017.

[5] X. Ma, Y. Liu, L. Zhao, Y. Lan, and J. Lu, "Specification and enforcement of the general user authorization query problem in role based access control system," *Open Automation and Control Systems Journal*, vol. 6, pp. 692-698, 2014.

[6] G. T. Wickramaarachchi, W. H. Qardaji, and N. Li, "An efficient framework for user authorization queries in RBAC systems," in *Proc. ACM Symposium on Access Control Models and Technologies*, 2009, pp. 23-32.

[7] J. Lu, Z. Wang, D. Xu, C. Tang, and J. Han, "Towards an efficient approximate solution for the weighted user authorization query problem," *IEICE TRANSACTIONS on Information and Systems*, vol. E100.D, pp. 1762-1769, 2017.

[8] A. Samuel, M. I. Sarfraz, H. Haseeb, S. Basalamah, and A. Ghafoor, "A framework for composition and enforcement of privacy-aware and context-driven authorization mechanism for multimedia big data," *IEEE Transactions on Multimedia*, vol. 17, pp. 1484-1494, July 2015.

[9] D. Kulkarni and A. Tripathi, "Context-aware role-based access control in pervasive computing systems," in *Proc. ACM Symposium on Access Control Models and Technologies*, 2008, pp. 113-122.

[10] S. Hosseinzadeh, S. Virtanen, N. D áz-Rodr guez, and J. Lilius, "A semantic security framework and context-aware role-based access control ontology for smart spaces," *International Workshop on Semantic Big Data*, 2011.

[11] A. D. Wood, J. A. Stankovic, G. Viron, L. Selavo, and Z. He, "Context-aware wireless sensor networks for assisted living and residential monitoring," *IEEE Network*, vol. 22, pp. 26-33, July 2008.

[12] J. E. Bardram, T. R. Hansen, M. Mogensen, and M. Soegaard, "Experiences from real-world deployment of context-aware technologies in a hospital environment," in *Proc. International Conf. on Ubiquitous Computing*, pp. 2006, 369-386.

[13] N. Davies, K. Cheverst, K. Mitchell, and A. Efrat, "Using and determining location in a context-sensitive tour guide," *Computer*, vol. 34, pp. 35-41, 2001.

[14] Y. Shi, W. Xie, G. Xu, R. Shi, E. Chen, Y. Mao, and F. Liu, "The smart classroom: Merging technologies for seamless tele-education," *IEEE Pervasive Computing*, vol. 2, pp. 47-55, 2003.

[15] G. Neumann and M. Strembeck, "An approach to engineer and enforce context constraints in an RBAC environment," in *Proc. ACM Symposium on Access Control Models and Technologies*, pp. 65-79, June 2003.

[16] J. Bacon, K. Moody, and W. Yao, "A model of OASIS role-based access control and its support for active security," *ACM Transactions on Information and System Security (TISSEC)*, vol. 5, pp. 492-540, Nov 2002.

[17] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in

INFOCOM'10 Proc. of the 29th Conference on Information Communications, 2010, pp. 534-542.

[18] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, "Towards a better understanding of context and context-awareness," in *Proc. of the 1st International Symposium on Handheld and Ubiquitous Computing*, 1999, pp. 304-307.

[19] R. Nick, J. Pascoe, and D. Morse, "Enhanced reality fieldwork: The context aware archaeological assistant," *Bar International Series*, pp. 269-274, 1999.

[20] B. Schilit, N. Adams, and R. Want, "Context-aware computing applications," in *Proc. of the 1994 First Workshop on Mobile Computing Systems and Applications*, pp. 85-90, Dec 1994

[21] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based access control," *ACM Transactions on Information and System Security (TISSEC)*, vol. 4, pp. 224-274, Aug 2001

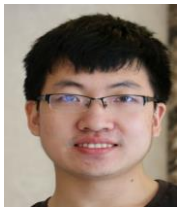
[22] A. Anne. "XACML profile for role based access control (RBAC), version 2.0, " *Specification]*, 2004.

Gang Liu received M.S. and Ph.D degrees in computer science and technology from Xi'an Jiaotong University, Xi'an, China, in 2001 and 2004 respectively.

He has been a faculty member of School of Computer Science and Technology at Xidian University since 2007, where he is currently an associate professor. His major research interests include embedded system, information security and trusted computing.

Shaomin Ji is currently a M.S at Xidian University, Xi'an, China.

His research interests include information security, access control and security model of integrity and confident in embedded system.



Runnan Zhang is currently a Ph.D at Xidian University, Xi'an, China.

His research interests include information security, access control and security model of integrity and confident in embedded system.

Guofang Zhang is currently a M.S. at Xidian University, Xi'an, China.

His research interests include information security, access control and security model of integrity and confident in embedded system.

Lu Fang is currently a M.S at Xidian University, Xi'an, China.

Her research interests include information security, access control and security model of integrity and confident in embedded system.

Quan Wang received the B.S., M.S. and Ph.D degrees in computer science from Xidian University in 1992, 1997 and 2008 respectively.

He is now a professor at the School of Computer Science and Technology and the institute director of the Institute of Computer Peripheral Equipment. His research interest includes embedded system.