International Journal of Grid Distribution Computing
Vol.7, no.3 (2014), pp.1-12
http://dx.doi.org/10.14257/ijgdc.2014.7.3.01

Role-based Access Control Using Ontology in Cloud Storage

Hong Sun[1], Xueqin Zhang[2] and Chunhua Gu[3]

[1, 2]School of Information Science and Engineering, East China University of Science and Technology, Shanghai 200237, China
[3]Shanghai University of Electric Power
[1]hsun2013@163.com, [2]zxq@ecust.edu.cn, [3]chgu@ecust.edu.cn

Abstract

With the development of cloud computing, and as the basis of data services, security problems of cloud storage are growing more attention. Based on distributed storage, multi-domain and multi-tenant characteristics, combined with access control technologies, this paper sets up the Role-based Access Control using Ontology and domians in Cloud Storage (DOnto_RBAC), which could provide a concise and effective strategy for service providers (isps). According to the characteristics of access control in cloud storage, based on the standards (called CDMI), this paper adds Domains and Time constraints of roles into RBAC. With ontology technologies and the OWL language, this paper establishes ontology model including entities' descriptions and strategies to realize reasoning of multi-domain access control permissions. We realized our system through Python and established Restful APIs. Experiments in campus-level cloud storage called Swift showed that, using requests with Restful format commands, DOnto_RBAC was proved to be effective to manage distributed and multi-domain data in cloud storage.

Keywords: *Cloud Storage, Cross-domain, RBAC, Access Control, Ontology*

1. Technical Background and Research Status of Access Control in Cloud Storage

As an important node of the cloud, cloud storage [1] is the underlying foundation and processing core of cloud computing. Thus, its security is a main point for cloud computing. For every user, only the ones who have permissions can manipulate data and the others without authorizations cannot. Secondly, how to share resources freely with certain users or groups in the recording time (Pay-As-You-Go), has become an issue. Lastly, how to refine the managed objects requested from large numbers of users and how to separate users and privileges in cloud storage to protect users' data should be considered.

While most IT service providers have provided cloud storage services and their own access control policies [2]. For example, Hadoop uses the classic DAC model [3], which is simple to realize. But DAC is difficult to solve authority migration. Amazon S3 proposed "bucket" (similar with folders) to manage data resources security [4] which is also combined with ACLs to authorize users. However the management is not flexible enough when used by millions of users in cloud. Additionally, Redhat, IBM and SUN are mainly through authentication and encryption mechanisms based on the Kerberos protocol [5] to ensure reliable and secure authentications. However all providers do not have strategies special for the distributed cloud.

At present, there are numbers of relevant researches about data ciphertext of access control [6-8], in which data owners could pre-encrypt before storing through distributing keys and authorities to users. For examples, ParaScale Cloud Storage (PCS2.5) ensures data replication and migration process safe [9] through providing encryption technologies. However, the growth of the amount of keys and distribution mechanisms will increase greatly which may lead to become a system bottleneck. There are studies using security tags in BLP model [10] to describe security attributes to enhance security between subjects and objects. However, there are no uniform descriptions and specifications for different concepts and descriptions to support security policies in cross-domains. Thus, people imported semantic ontology into access control [11, 12]. Though describing similar objects in different systems uniformly, ontologies could generalize concepts about permission management to unify the differences of concepts and terms in different applications. Therefore, ontologies bring an innovation to optimize security issues in distributed or multi-domain environment.

Focus on those above problems and analyses, we proposed an ontology-based cloud storage access control model called DOnto_RBAC, which is both aimed at campus-level cloud file storage applications and the public cloud. According to Cloud Storage Standards (CDMI), our system increased access domains and constraints about roles and privileges. Meanwhile, taking advantage of OWL and Ontologies, this paper established the ontology of DOnto_RBAC including entities and strategies to realize the reasoning on logical contradiction test, subclasses classifications and individual archives. Implemented by Python, system is called by Restful API and communicated through pipeline. With conducted experiments in campus-level cloud file storage applications, DOnto_RBAC is verified effectively in distributed and multi-domains cloud storage.

2. Improvements of DOnto_RBAC Access Control Model

In April 2010, American Storage Networking Industry Association (SNIA) released the Cloud Data Management Interface (CDMI), the first unified data and interfaces standards, which is special for accessing and managing cloud storage. In CDMI, the resource objects are divided into five categories: the logic management unit (Domain), data storage space (Container), Metadata, Queue and Capability which is used to describe the capacity of objects. Also CDMI makes specifications for users' accounts and safe access. It means that all of requesting services, redundant backup and calling data should follow CDMI. Thus, this paper introduced Domain into RBAC96 [13-14] to manage objects uniformly.

2.1. Definitions in DOnto_RBAC

Based on RBAC96, DOnto_RBAC imports the concepts of Domain and Capability: each Domain corresponds to a set of objects that can be unified in management logically which represents the ownership of data in the administration of cloud storage, and each Domain has its own capabilities to stipulate the pricing strategies, statistics and metadata that can be modified. There is the model of DOnto_RBAC in Figure 1 mainly including 10 objects and the specific contents are as follows.

Def. 2.1: $Users = \{u_1, u_2, ..., u_n\}$: A set of users that means users who initiates the requests.

Def. 2.2: $Roles = \{r_1, r_2, ..., r_n\}$: A set of roles that is attached with certain permissions. There are two subsets called $sysRoles$ and $posRoles$: $sysRoles$ refers to the only one system role special for each user which cannot be modified by users, and $posRoles$ is template roles which could be modified in their own environments.

Def. 2.3: $Sessions = \{s_1, s_2, ..., s_n\}$: A set of sessions to acquire and active roles.

Def. 2.4: $Operations = \{ops_1, ops_2, ..., ops_n\}$: Including GET, POST, PUT and DELETE in experiments.

Def. 2.5: $Objects = \{obs_1, obs_2, \ldots, obs_n\}$: A set of objects that can be operated. There are three kinds of objects in DOnto_RBAC including Domains, Capabilities and Data.

Def. 2.6: $Onto\_obj$: A set of abstract metadata of objects which also has three kinds according to the classification of objects to ensure the isolation of real resource.

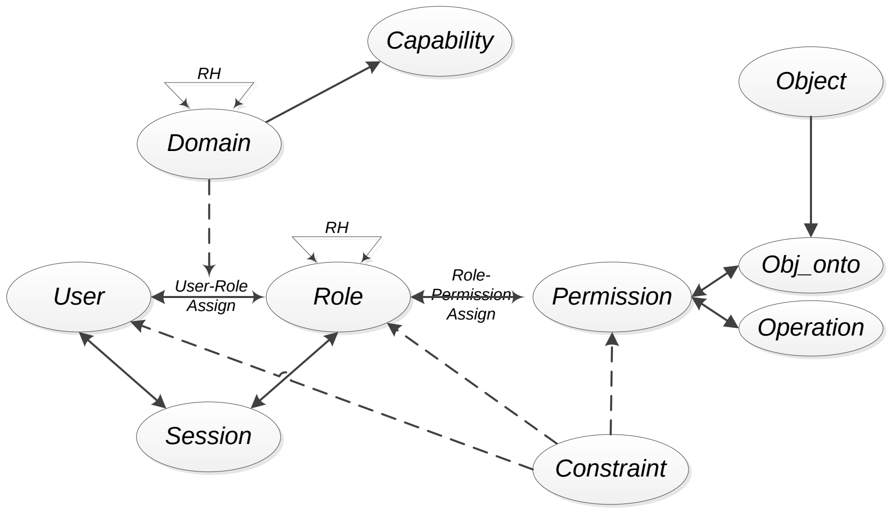


Figure 1. Model of DOnto_RBAC

Def. 2.7: $Domains = \{do_1, do_2, \ldots, do_n\}$: A set of objects that can be managed uniformly. Domain represents the concept of administrative ownership of stored data to manage conveniently, so there is no real data actually stored in domains. Users have their own domain and could inquire, combinate and redistribute their permissions of domains and data.

Def. 2.8: $Permissions = 2^{Operations \times obj\_onto} = \{per_1, per_2, \ldots, per_n\}$: A set of permissions that are represented by the Cartesian product.

Def. 2.9: $Constraints = \{cos_1, cos_2, \ldots, cos_n\}$: A set of constraints that are including the role activation constraints, run-time constraints, exclusive-role constraints and separation of rights etc., and the specific content will be described in detail in the next section.

Def.2.10: $Capabilities = \{cbs_1, cbs_2, \ldots, cbs_n\}$: A set of non-rights management policies that are including pay policies and statistical strategies.

2.2. Security policies in DOnto_RBAC

To divide individual users and groups, Domains have attributes to be classified named $D\_attribute$ which is a tuple of D_type and D_owner . D_type Means the type of Domains and D_owner means the owner of Domains. D_type Is divided into three kinds including $private, protected, public$:

- $private$ represents the private domains whose access members is only D_owner ;
- $protected$ represents the protected domains, in which the objects are shared for all members, while cannot be traversed in cloud storage;
- $public$ Represents those public domains that are registered in the system databases, which can be traversed for all users in cloud storage.

2.3. Constraints of DOnto_RBAC

To meet the demand of dynamically changings of roles and permissions, we improved time constraints in the part of user-to-role assignment and permission-to-role assignment. Therefore, paper proposed some improvements in DOnto_RBAC constraints, as follows:

a. *Exclusive-role constraints*. There are role that a user cannot be owned simultaneously in the same domain. $sysRoles$ are pairwise mutually exclusive events, and $posRoles$ can be set according to the actual use case applications.
b. *Capacity-role constraints and Cardinality-role constraints*. Capacity-role constraints mean limited numbers of users for each role, and Cardinality-role constraints mean limited numbers of roles for each user. For example, DOnto_RBAC provides only one domain administrator that is a Capacity-role constraint.
c. *State-role and time constraints.* Paper defined the state of role are active and disable to avoid repeated recycling roles, and defined time constraints as $rr\ (r_{id}, r_{bt}, r_{et}, r_{all}, RA)$, in which r_{id} means the IDs of roles. Also paper defined the total active time is RA and the actual active time is r_{all}. While we use (r_{bt}, r_{et}) to indicate the starting-time and the terminal-time of an active role, in which the time of active roles need to be activated by (r_{bt}, r_{et}) and r_{all}. Thus, we have two rules as follows.

Constrain 3.1. Roles those are earlier than r_{bt} and later than r_{et} cannot be activated.

Constrain 3.2. System records roles active time from the start and the end to calculate this duration r_{all}' for accumulating the total r_{all}, and make comparison with RA. If $r_{all} \geq RA$, then this role cannot be active.

3. Establishment and Reasoning of DOnto_RBAC Ontology

This paper established the Ontology of DOnto_RBAC to be as the basic axiom system. Through Ontology, we can facilitate relationships and unify different concepts in multi-domains including users, roles, permissions and other core concepts of objects, as well as user-to-role assignments and role-to-permission assignments, etc. On the basis of semantic ontology, paper also used descriptive-language to infer, which can make inspection for access control rules and realize the intelligent classification.

3.1. Establishment of DOnto_RBAC Ontology

In order to ensure the correctness and completeness of this ontology, we constructed the Ontology strictly following classical methods to build it.-Firstly, through existing ontologies, this paper determined the important terms and concepts in the ontology of cloud access control model. And then paper extracted core classes and their clear definitions from duplicate ones.

Table 1. Core Classes of Access Control

Names	Definitions
Policy	Specific access rules in Donto_RBAC, including user-to-role and role-to-permission distribution policies.
Role	Core elements connection between users and permissions including inheritance relations between parents and child classes.
Resource	Actual physical storage resources, ie. data.
Domain	A collection of objects that could be unified management which have inheritance relationships.
Capability	Other non-rights-managed strategies in cloud storage.
… …	

As shown in Figure 2, we use RDF and triples representation to articulate the concepts of classes, attributes and relationships in access control, in which contains domains, object-relational properties and ranges. There are 10 concepts to describe the classed in DOnto_RBAC, several specifically defined examples as follows in Table 1.

Secondly, this paper determined Data Properties, Object Properties and levels between parent and child classes, as while it gave a detailed interpretation. Through Data Properties, ontology could describe its own properties with classes, its subclasses and their Individuals. According to details in CDMI, DOnto_RBAC completed classes on the basis of supplements and improvements. As shown in Figure 2, relationships on lines represent the Object Properties between classes. There are 16 classes in DOnto_RBAC in total. For example, juniorRoleOf and seniorRoleOf represent inheritance relationships between roles.

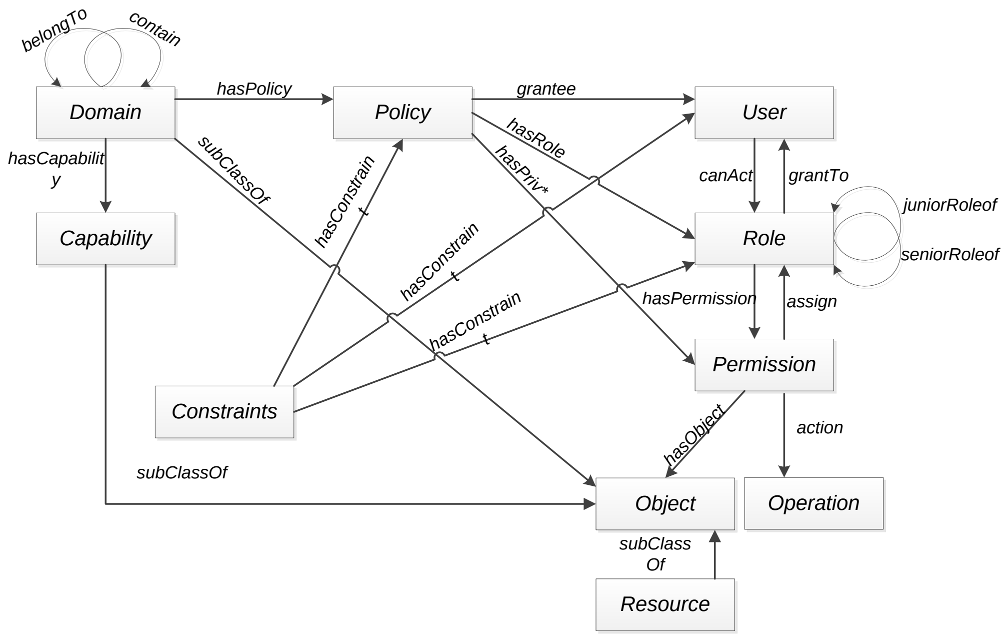


Figure 2. DOnto_RBAC Ontology

Lastly, we created the Individuals which reflected actual data and meaning of classes. Data Properties are the characteristics to connect classes and Individuals, so that Individuals are the values of corresponding Data Properties.

3.2. Reasoning of DOnto_RBAC Ontology

Through OWL ontology language, DOnto_RBAC could realize the descriptions by machine languages. Meanwhile using OWL DL reasoning language to code the clauses for inference rules, this paper formulated the application logic reasoning in classes, properties and individuals to solve certain problems through semantic methods.

(1) Contradiction Tests. According to logical relationships between classes including intersection and union, etc., when defining, OWL classes were default overlapping that need to distinguish by Disjoint Classes. Also to avoid logical contradictions, this paper conducted Contradiction Tests to examine. Specific inference rules are as follows:

Rule 1: Every user cannot simultaneously belong to the trial-user class (DemoUser) and the premium-user class (ChargeUser). As the length of paper, rest similar rules were not listed.

(2) Subclasses Classifications based on necessary and sufficient conditions. By using two important fields, ontologies could describe class attributes itself. The first is "Super classes" corresponding to essential conditions in mathematics which means "all subclasses belong to the current class should meet these necessary conditions" and on the contrary cannot be established. The other is "Equivalent classes" corresponding to necessary and sufficient conditions that means "all subclasses and individuals with these Equivalent classes of the current class should belong to current classes". Through this determination, this paper could solve errors, confusions, inefficiencies or omissions during building ontology hierarchy levels.

Rule 2: The resources authorities are Resource Permission which represented as the Cartesian product of Resource objects and the Operation. Its formalized expressed as:

$$Equivalent\ classes = \{Permission\ \text{and}\ action\ \text{some}\ (Operation\ \text{and}\ hasObject\ \text{some}\ Resource)\}$$

Classes satisfied the above condition can be judged as the subclass of Resource Permission.

(3) Individual Archives. Because of huge and uncountable numbers of individuals, DOnto_RBAC realized individual archives using logical reasoning judgments by individuals attribute tags and its corresponding values to decrease the consumption of manually classification. There is at least one condition about Data Properties in Individual Archives.

Rule 3: If there is an individual with fields of UserName and UserID, and also in line with the corresponding data format conditions, then this individual would be considered as an instance of the User class which formalizations are as follows:

$$Equivalent\ classes = \{UserName\ \text{some}\ string\ \text{and}\ UserID\ \text{some}\ integer[>\text{"0"\^{}\^{}integer}])\}$$

Similarly, when the user space is larger than 20G, this individual will be determined as the instance of the ChargeUser class:

$$Equivalent\ classes = \{User\ \text{and}\ UserSpace\ \text{some}\ integer[>\text{"20"\^{}\^{}integer}])\}$$

More similar rules are not repeated here. Nonetheless from above rules OWL DL can be defined as machine-readable rules in semantic Web to reduce the overhead from servers.

4. Realization and Experiments of DOnto_RBAC System

4.1. Overall design for DOnto_RBAC System

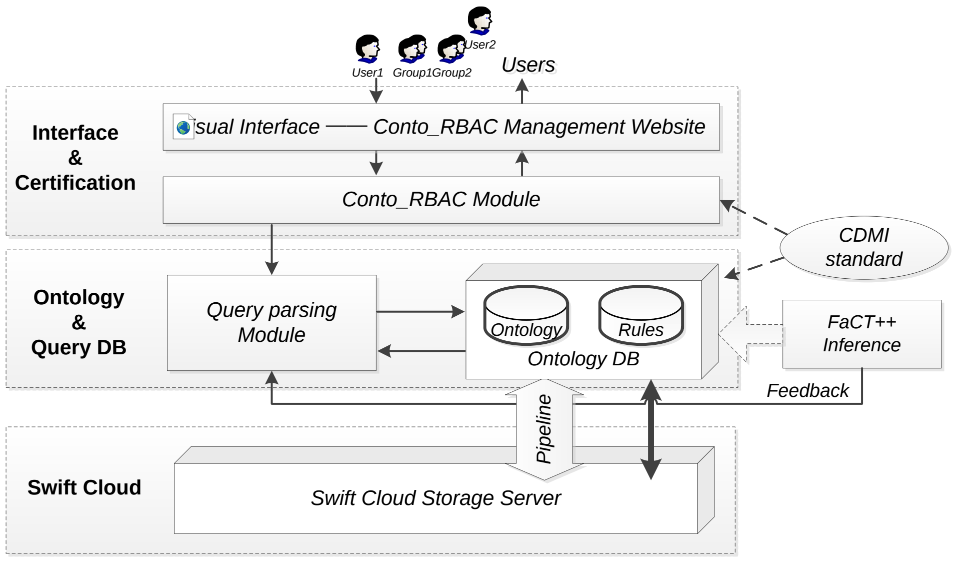
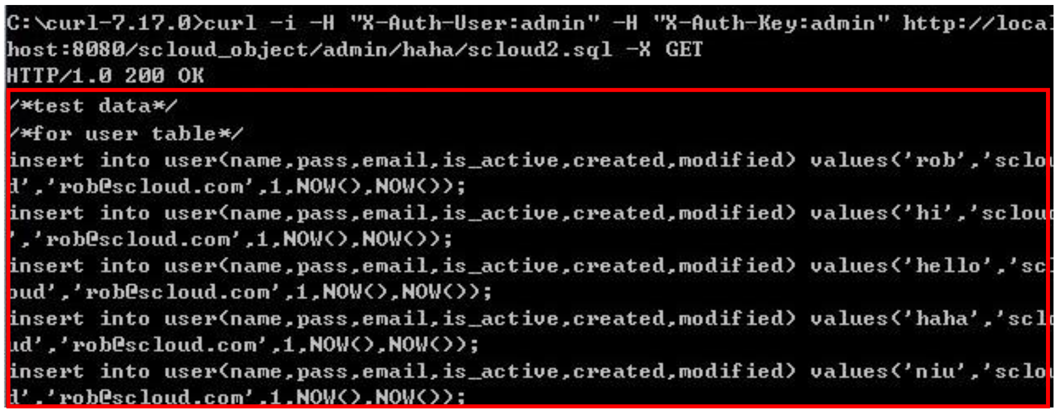


Figure 3. Framework for DOnto_RBAC Web

DOnto_RBAC goals to realize an access control system in public cloud storage file system, and the experimental environment is a campus-level cloud storage file system. This system is based on open source Swift in Open Stack framework which has achieved cloud files storage, sharing and downloading. For the tasks and environments, DOnto_RBAC designed three modules, six sub-modules and a communications pipe, and its designing diagram is shown in Figure 3 including the Interfaces and Authentication Section, the Ontology Library and Inquiries, Section and Swift Cloud Storage. Meanwhile, through FaCT++, system realized reasoning of ontology, and lastly, we used Pipeline as communication mechanism of system.

4.2. Resource Access Experiments

As shown above Figure 4, customers could use Restful style commands and URI which identify resources addresses to access DOnto_RBAC. And there were the requested content in the file named “scloud.sql” through the Swift's Pipeline in the box of Figure 4. It should be particularly noted that, due to the experimental environment was still in development, the response results had been currently only achieved in the dos environment. For further research, system would be saved as various formats.



```
C:\curl-7.17.0>curl -i -H "X-Auth-User:admin" -H "X-Auth-Key:admin" http://local
host:8080/scloud_object/admin/haha/scloud2.sql -X GET
HTTP/1.0 200 OK
/*test data*/
/*for user table*/
insert into user(name,pass,email,is_active,created,modified) values('rob','sclo
d','rob@scloud.com',1,NOW(),NOW());
insert into user(name,pass,email,is_active,created,modified) values('hi','sclou
','rob@scloud.com',1,NOW(),NOW());
insert into user(name,pass,email,is_active,created,modified) values('hello','sc
oud','rob@scloud.com',1,NOW(),NOW());
insert into user(name,pass,email,is_active,created,modified) values('haha','scl
ud','rob@scloud.com',1,NOW(),NOW());
insert into user(name,pass,email,is_active,created,modified) values('niu','sclo
d','rob@scloud.com',1,NOW(),NOW());
```

Figure 4. Restful Interface Realization in Cloud Storage

4.3. Security Experiments

In order to verify the security of DOnto_RBAC system, we configured two domians: TDomain and BDomain in Figure 5 which D_owner were respectively Tom and Bob.

Experiment 1: As the owner of TDomain, Tom could call all resources in it after checking and logining as well as Bob. Meanwhile, service provider also couldnot get their data. The ISP only has rights to change domain status to manage domain information effectively.

Experiment 2: Both Alice and Kate had the role of Operator, if Alice's request were:

-i -H "X-Auth-User" : Alice -H "X-Auth-Key" : alice_password http://ip/scloud_domain/BDomain -X GET

Experiments showed that even Alice also had the role of Operator, Alice could not access any information without permissions in BDomain from Bob.

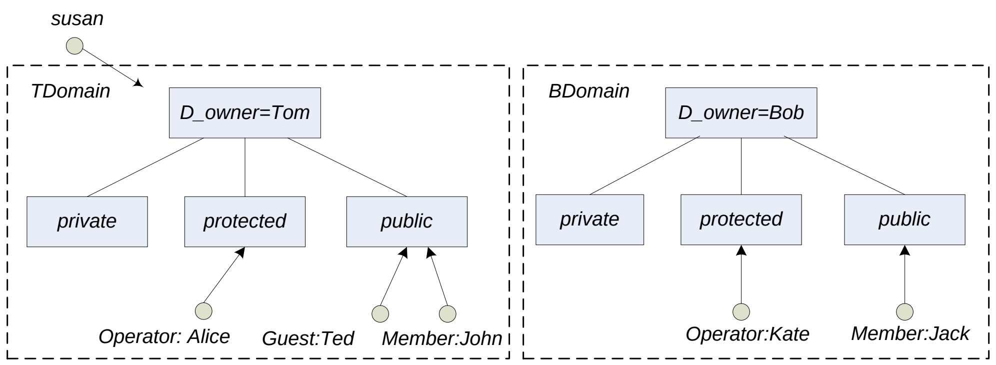


Figure 5. Logical Management Domains

Experiment 3: Ted and John in public domain of TDomain could be assigned different roles that were Guest and Member which were related to different permissions.

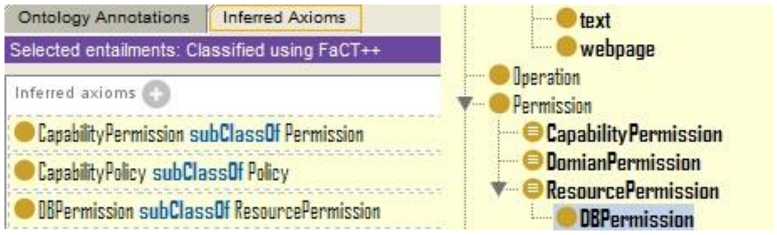
Experiment 4: susan who was not belong to TDomain or BDomain may request traverse to public-TDomian and public-BDomain, but susan could not get other further information.

4.4. Implementation and Experiments on Ontology and Reasoning

With Protégé and FaCT++, we conducted following experiments to establish ontologies and to verify the logic reasoning improvements in DOnto_RBAC.

Experiment 1: Contradiction Tests. According to reasoning rules in Section 3, because of the disjoint classes of Resource and Domain, experimental subclass named "webpage" which parent are both Resource and Domain caused the reasoned (called FaCT++) reported an error.

Experiment 2: Subclasses Classifications based on necessary and sufficient conditions. DOnto_RBAC defined different subsets of Permission based on roles and application scenarios. Experiments created a database permission named DBPermission, and then FaCT++ would classify automatically according to its necessary conditions. As shown in Fig. 6, DBPermission was inferred to the subclass of ResourcePermission, because DB was the subclass of Resource.



CapabilityPermission subClassOf Permission
CapabilityPolicy subClassOf Policy
DBPermission subClassOf ResourcePermission

Figure 6. Equivalent Classes Based of Classification Results

Experiment 3: Individual Archives. In order to facilitate entry of the individuals, this paper conducted individual archives experiments. Accroding to Rule 3 in 3.2, when an instance contains UserName and UserID with corrected data formats, and then it would be inferred as an individual of User, as well as ChargeUser. As shown in Figure 7, although this instance including UserName and its value, it was not inferred into User; and the individual of Bob was inferred into User and the subclass ChargeUser that achieved the automatic classification.

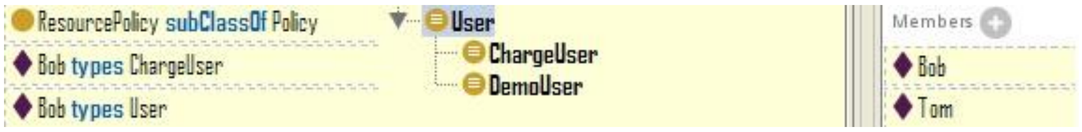
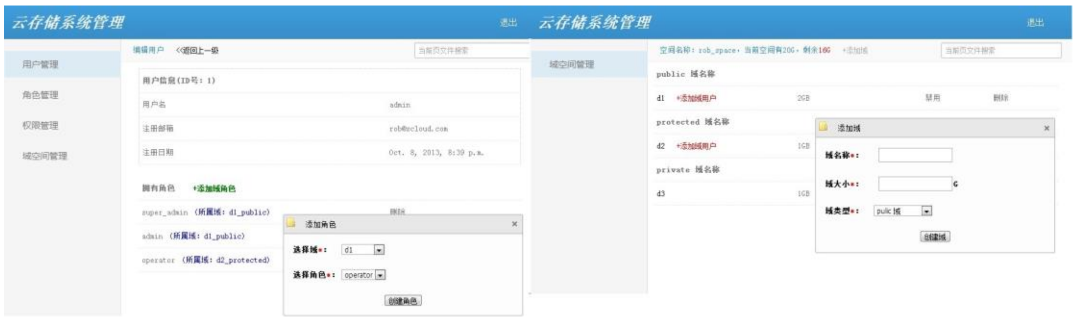

ResourcePolicy subClassOf Policy
Bob types ChargeUser
Bob types User

Figure 7. Automatic Classification of Individuals

The Contradiction Tests are used to be the verification during establishment to prevent contradictions with original models, and OWL configuration files of other reasoning rules could be modified in batch using JAVA, like Jena. Then Protégé will read OWL instance files and achieve reasoning.

4.5. Management Platform of DOnto_RBAC System

As the interactive interface between users and access policies, there are two kinds of users in DOnto_RBAC system that is isps and customers to manage classes of Domian, Roles and Permissions, etc. Aimed at users' demands, there were two interfaces in Fig. 8: the left is Administrator Interface, which achieved four modules; and the right is User Interface, which realize the management of his domain.



用户信息(ID号：1)

用户名	admin
注册邮箱	rob@zcloud.com
注册日期	Oct. 8, 2013, 8:39 p.m.

拥有角色

super_admin （所属域：d1_public）
admin （所属域：d1_public）
operator （所属域：d2_protected）

选择域*：d1
选择角色*：operator

空间名称：rob_space，当前空间有20G，剩余16G

public 域名称

d1	+添加域用户	2GB

protected 域名称

d2	+添加域用户	1GB

private 域名称

d3	1GB

域名称*：
域大小*： G
域类型*：pulic 域

Figure 8. Interface of Front Service and Users' Domains

5. Conclusion

Paper used the many-to-many user-to-role and role-to-permission assignment to simplify the architecture of roles in cloud storage. While according to CDMI, paper extracted the logical management unit called Domain to establish authorized relationships between users and domains in DOnto_RBAC, which greatly decreased expenses during authorization. DOnto_RBAC increased the fine-grained time constraints and state constraints about roles and Domains to facilitate billing and recall roles automatically. Moreover, system used ontologies and OWL to describe DOnto_RBAC, and designed inference rules based on the logical description language. Meanwhile paper established DOnto_RBAC Ontology by Protégé and achieved reasoning by FaCT++. Finally, through experiments and analysis, system verified its security and authorization reasoning. However, the management system and reasoning system in DOnto_RBAC merely realized correlation and synchronization at the level of database without the connector between designed system and FaCT++. Thus the issue of how to assist human-to-computer interaction by semantic reasoning technology should be optimized. Moreover, experiments in this paper were only specific to security of DOnto_RBAC. As the limitation of data levels, it should be further explored that the efficiency of reasoning tests.

References

[1] R. L. Krutz and R. D. Vines, "Editors. Cloud Security: A Comprehensive Guide to Secure Cloud Computing", Wiley Publishing, Inc., **(2010)**.
[2] S. C. Yu, C. Wang, K. Ren and W. J. Lou, "Attribute based data sharing with attribute revocation", Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, **(2010)** April, Beijing, China.
[3] K. Shvachko, H. Kuang, S. Radia and R. Chansler, "The hadoop distributed file system", Proceedings of the 26th IEEE Symposium on Mass Storage Systems and Technologies (MSST), **(2010)** May 3-7, Incline Village, NV.
[4] Amazon.com, Inc. AWS security whitepaper, **(2011)**, http://awsmedia.S3.amazonaws.com/pfd/AWS_Security_whiterpaper.pdf/.
[5] O. O'Malley, K. Zhang, S. Radia, R. Marti and C. Harrell, "Hadoop security design", Technical Report, **(2009)** October.
[6] E. Shi, J, Bethencourt, T. -H. H. Chan, D. Song and A. Perrig, "Multi-dimensional range query over encrypted data. Proc. of IEEE Symp of Security & Privacy, **(2007)** May 20-23, Piscataway, NJ.
[7] J. W. Shu, W. Xue, M. Xue and Z. R. Shen, "A realization of secure storage system in cloud storage", State Intellectual Property Office of P.R.China, Publication No.102014133A, **(2011)** April 13.
[8] C. Hong, M. Zhang and D. G. Feng, "AB-ACCS: a cryptographic access control scheme for cloud storage", Journal of Computer Research and Development, vol. 47, **(2010)**, pp. 259-265.
[9] "Hitachi Data Systems", Parascale cloud storage software-reference and whitepapers, **(2010)**, http://www.hds.com/index.php/technology/how-it-works/73-parascale-cloud-storage/138-parascale-cloud-storage-reference-papers.

[10] N. Kwak and C. Chong-Ho, "Input feature selection for classification problems", IEEE Transaction on Neural Network, vol. 3, no. 1, **(2002)**, pp. 143-157.
[11] M. Dong and R. Kothari, "Feature subset selection using a new definition of classifiability", Pattern Recognition Letters, vol.24, no. 9-10, **(2003)**, pp. 1215-1225.
[12] D. D. Li and J. L. Tan, "Research and implementation of an ontology-based privilege management system", Computer Engineering, vol.31, no.13, **(2005)**, pp. 43-45.
[13] W. Ji-yi, F. Jian-qing, P. Ling-di and X. Qi, "Study on the P2P Cloud Storage System", Acta Electronica Sinica, vol. 35, no.5, **(2011), pp.** 1100-1107.
[14] R. S. Sandhu, E. J. Coyne, H. L. Feinstein and C. E. Youman, "Role-based access control models", IEEE Computer, vol. 29, no. 2, **(1996)**, pp. 38-47.

Authors

Hong Sun, she has being studying the M.S. degree in Electric and Communications Engineering from East China University of Science and Technology (EC[illegible], from 2011. Her research interests are informatio[illegible]loud storage.

Xueqin Zhang, she received the Ph.D. degree in Detection Technology and Automation Devices from East China University of Science and Technology (ECUST), Shanghai, China, in 2007. Since 1998, she has been in the Electrical and Communication Engineering Department, ECUST, where she is currently an ASSOCIATE PROFESSOR. At 2006, she worked as a visiting scholar in University of Wisconsin Madison. Her research interests include pattern classification, information security and data mining etc.

Chunhua Gu, he received the Ph.D. degree in Control Science and Engineering from East China University of Science and Technology (ECUST), Shanghai, China, in 2007. From 1992 to 2013, he was in the Computer Science and Engineering Department, School of Information Science and Engineering, ECUST. Now he is a PROFESSOR at Shanghai University of Electric Power. At 2002, he worked as a visiting scholar in School of Computing and Information Sciences, Florida International University. His research interests include intelligent computing, software engineering and information security.