International Journal of Cloud Computing and Services Science (IJ-CLOSER)
Vol.2, No.2, April 2013, pp. 106~115
ISSN: 2089-3337

Scalable Multi-Tenant Authorization in Highly-Collaborative Cloud Applications

Samy Gerges, Sherif Khattab, Hesham Hassan, Fatma A Omara
Faculty of Computers and Information, Cairo University, Egypt

Article Info

Article history:

Received Oct 25th, 2012
Accepted Dec 30th, 2012

Keyword:

Cloud Computing
Security
Authorization
RBAC
Inter domain Mapping

ABSTRACT

Collaborative applications have lately gained extra momentum due to two recent phenomena: data explosion and cloud computing. With more and more data and applications being hosted in the "cloud", it becomes easier for organizations with varying levels of mutual trust to share and collaborate over resources. However, a pressing challenge remains with the need of each organization to control access to its resources. Authorization, usually implemented as role-based access control (RBAC), has been recently proposed as a consolidated, multi-tenant cloud service, whereby RBAC rules of the collaborating organizations are stored centrally with a trusted authorization provider to mask heterogeneity and to simplify management. A critical factor to the success of such aggregating approach to access control is the scalability of the rule store to the number of collaborating organizations and to the degree of collaboration. In this paper, we focus on the scalability of the *online* rule store, that is, the set of rules that are checked with every authorization request, and thus, needs to reside in fast storage (e.g., main memory). We propose an authorization system that scales well to the degree of collaboration and call our system highly-collaborative authorization service (HCAS). HCAS is based on role mapping, a well-known RBAC technique that maps roles across collaborating organizations. HCAS replaces the inter-domain RBAC rules with a more scalable set of role-mapping tuples. Using simulation, we show that HCAS achieves super-linear savings in the size of online rule store. HCAS exhibits a favorable behavior of a slightly decreasing rule set with increasing degree of collaboration in highly-collaborative settings. Scalability of online memory in RBAC multi-tenant authorization systems enables efficient software and hardware implementations.

Copyright © 2013 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

Samy Gerges
Faculty of Computers and Information, Cairo University, Egypt
Email : samygerges2010@hotmail.com

1. INTRODUCTION

Cloud computing services have been booming recently, and many organizations are moving to host their applications with cloud computing providers to reduce cost, to benefit from elasticity and dynamic on-demand provisioning, and to convert capital costs to operational costs with the cloud's pay-as-you go billing model. Current cloud instances can be classified into public, private, community, and hybrid [1].

Many cloud applications involve multiple collaborating organizations. For example, the federal US government cloud has different agencies collaborating together and sharing sensitive data among each other [2], and healthcare institutions are moving patients' sensitive data into the cloud and collaborating over shared resources [3]. A major concern with these two examples and with other collaborative cloud applications are security. New security challenges face the cloud providers and collaborating organizations to federate their own authentication and authorization systems and allow for cross-domain security mechanisms to protect the shared sensitive data from unauthorized access [4], [5]. One of the most popular authorization

methods is role-based access control (RBAC), whereby each organization maintains a list of roles with different permissions and different users that assume the roles [6]. In RBAC, a request to access a resource with a desired privilege is granted or denied after a lookup for a rule that allows the requested privilege. A well-known technique in RBAC is inter-domain role mapping, whereby a role in one organization is mapped to another role in another organization to grant the desired privileges [7], [8], [9].

Federated, multi-tenant authorization services have been recently proposed [10], [11], whereby RBAC rules of the collaborating organizations and the cross-domain rules are stored centrally with a trusted authorization provider to mask heterogeneity and to simplify management. A critical factor to the success of federated RBAC access control is the performance of rule lookup, which depends on the scalability of the rule store size to the number of collaborating organizations and to the degree of collaboration, that is, the number of cross-domain RBAC rules. Scalability of online memory in RBAC multi-tenant authorization systems enables efficient software and potentially hardware implementations. The main problem studied in this paper is the scalability of the online rule store, that is, the set of rules that are checked with every authorization request, and thus, needs to reside in fast storage (e.g., main memory).

We show that the size of the online rule store increases quadratically with the number of collaborating organizations in highly-collaborative cloud applications, applications in which resources are shared massively across organizations. In this paper, we propose an authorization system, which we call the highly-collaborative authorization service (HCAS), that scales well to the degree of collaboration. HCAS is based on role mapping and replaces the cross-domain RBAC rules with a more scalable set of role-mapping tuples. Using simulation, we show that HCAS achieves super-linear savings in the size of online rule store. HCAS exhibits a favorable behavior of a slightly decreasing rule-set size with increasing degree of collaboration in highly-collaborative settings. The contribution of this paper is two-fold. First, we define the online-memory scalability problem in the context of multitenant RBAC authorization systems and show that traditional role-to-object RBAC mapping suffers from quadratic size of online memory in terms of the number of participating organizations in highly-collaborative cloud applications. Second, we introduce a solution to the online-memory scalability problem that is based on the role mapping technique and that achieved super-linear saving in online memory compared to role-to-object mapping in many simulated highly-collaborative settings.

The rest of the paper is organized as follows. Section II discusses related work. In section III, we formulate the online memory scalability problem in highly-collaborative environments. Section IV describes the architecture and operation of the role mapping algorithm of our highly-collaborative authorization service (HCAS). In Section V, we explain our simulation-based evaluation of HCAS, in which we evaluate the savings in online-memory size achieved by role mapping in highly-collaborative environments. Section VI concludes the paper and discusses future work.

2. RELATED WORK

In this section we discuss related multi-tenant authorization services [10], [11] and role mapping approaches [7], [8], [9]. Calero *et al*. [10] has introduced a multi-tenancy authorization model suitable for cloud computing, which federates the management of hierarchical role-based access control for path-based object hierarchies. RBAC rules, both local to each organization and across domains, are stored centrally on behalf of the collaborating organizations. Every organization should represent all roles with all resources that need to be accessed locally or in any of the other collaborating organizations in statements (tuples) signed by an issuer from the organization hosting the resources. These tuples are stored in a central trusted database.

In Leandro *et al*. [11], a multi-tenancy authorization system using Shibboleth [12] has been proposed and implemented. Authorization is managed by the application (a collaborative Web authoring tool), and Shibboleth was used to implement cross-domain identity management without a trusted third party. Requests to the service provider for accessing a secure resource are redirected by Shibboleth to the organization to authenticate the requesting client against his organization and send a token again to the service provider, whereby the application's resource manager takes a decision for granting or denying the request.

In centralized, multi-tenant authorization services, the size of the authorization rule database for all statements for every organization can be prohibitively massive in a highly collaborative environment, in which a large portion of every organization's resources is shared with most of the other collaborating organizations. To the rescue comes the idea of role mapping to map user roles in each organization to the roles in the host organization, that is, the organization owning the shared resource. The role-mapping rules replace the rules that map roles to resources. Role mapping has the potential of reducing the number of RBAC statements to be stored and accessed by the authorization system [7], [8], [9].

Chen *et al*., [7] has proposed an inter-domain role mapping technique based on the principle of least privilege. They suggest a minimal cardinality for a role across a domain to avoid misuse of access. Their approach may cause the loss of some of desired privileges. Our HCAS adopts the approach of role splitting to

avoid privilege loss. Geethakumari et al. [9] has proposed a scalable role-mapping approach that is based on ranking roles and resources with specific scores and comparing the role and resource scores to decide on request granting or denial. The scores are granted by the authorization server of each domain and are updated as the request traverses the domain hierarchy. Setting the scores correctly is a non-trivial process and would have a drastic impact on security if the scores are not set carefully.

3. RESEARCH METHOD

In this section we lay down the definitions of highly-collaborative cloud applications, online and off-line authorization rule store, the online-memory scalability problem, which is the focus of this paper, and we highlight our proposed highly-collaborative authorization service (HCAS).

3.1. Collaboration Graphs

We assume that a group of organizations are running a set of collaborative applications hosted in a cloud provider. Each application is composed of services and accesses resources (e.g., data elements) provided by one or more of the collaborating organizations. Each organization has its own role-based access control (RBAC) authorization system that manages the privileges of each of its roles with respect to the organization's resources. Moreover, organizations grant privileges on their own resources to other organizations. We model collaboration as a random directed graph G = (V,E), whose vertices are all the organizations' roles and all the organizations' resources or objects. There are N_{org} organizations. Each organization has an expected number of roles N_{role} and an expected number of objects N_{object}. The set of resources of an organization org is denoted O_{org}. An edge in this graph goes from a role (in one organization) to a resource in the same or different organization. More specifically, an edge (i_m; r_n) means that role i in organization m has a certain privilege on resource r in organization n. Multiple edges may exist between the same role and resource to model different privileges. In the case that m = n, that is, the role and resource belong to the same organization, the edge is said to represent an **intra-domain** RBAC rule. In the case that m ≠ n, the edge is said to represent an **inter-domain** rule.

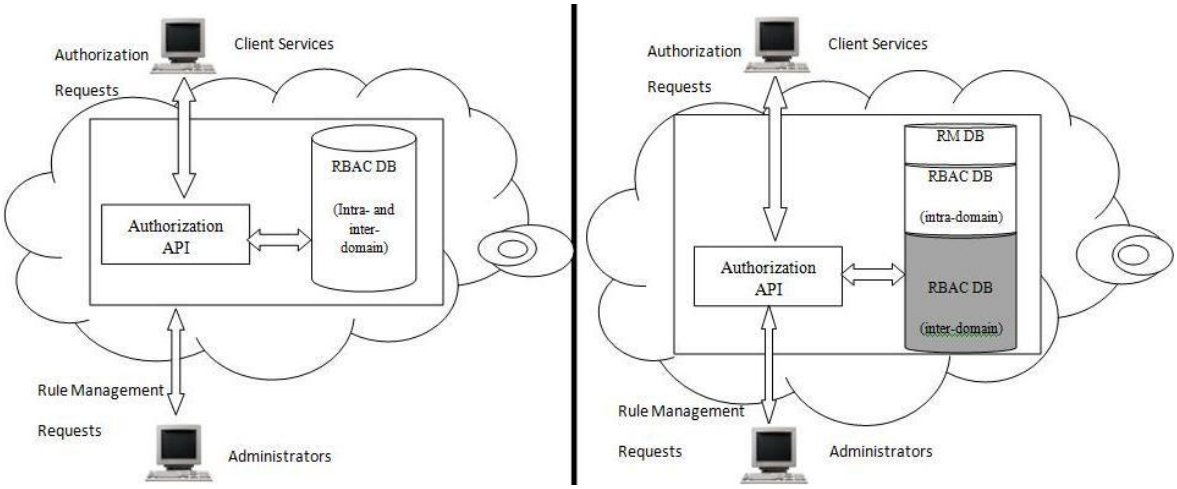

Fig. 1. Comparison between a typical multi-tenant authorization service (left) and our highly-collaborative authorization service (HCAS) (right). In a multitenant authorization service (e.g., [10]), all RBAC authorization rules are stored online (i.e., searchable by every authorization request). In a highly-collaborative environment, the number of RBAC rules is quadratic in the number of collaborating organizations. HCAS (right) replaces the quadratic inter-domain rules wit a set of role mapping tuples whose number is stable for many highly-collaborative scenarios. The inter-domain RBAC rule set if moved to off-line storage (shaded), which is accessed by domain administrators for adding, deleting, or modifying the inter-domain rules.

We assume that a group of organizations are running a set of collaborative applications hosted in a cloud provider. Each application is composed of services and accesses resources (e.g., data elements) provided by one or more of the collaborating organizations. Each organization has its own role-based access control (RBAC) authorization system that manages the privileges of each of its roles with respect to the organization's resources. Moreover, organizations grant privileges on their own resources to other organizations. We model collaboration as a random directed graph G = (V, E), whose vertices are all the organizations' roles and all the organizations' resources or objects. There are N_{org} organizations. Each organization has an expected number of roles N_{role} and an expected number of objects N_{object}. The set of

resources of an organization org is denoted O_{org}. An edge in this graph goes from a role (in one organization) to a resource in the same or different organization. More specifically, an edge (i_m; r_n) means that role i in organization m has a certain privilege on resource r in organization *n*. Multiple edges may exist between the same role and resource to model different privileges. In the case that $m = n$, that is, the role and resource belong to the same organization, the edge is said to represent an **intra-domain** RBAC rule. In the case that $m \neq n$, the edge is said to represent an **inter-domain** rule.

The average probability of an intra-domain edge, ′PL, is the probability of an edge between a role and a resource in the same organization averaged over all organizations. Similarly, the average probability of an inter-domain edge, ′PR, is the probability of an edge from a role in one organization and a resource in another organization averaged over all organization pairs *(n,m)* where $n \neq m$. We denote the expected total number of edges in the collaboration graph as T, the expected number of intra-domain (local) edges as $L = N_{org} \times N_{roles} \times N_{object} \times$ ′PL, and the expected number of inter-domain (remote) edges as $R = N_{org} \times N_{roles} \times (N_{org} - 1) \times N_{object} \times$ ′PR. The number of edges in a collaboration graph represents the number of RBAC rule tuples of the form *(i, r, p)*, indicating that role *i* has privilege *p* on resource *r*.

The probability ′PR of an inter-domain edge represents the **expected degree of collaboration**. A highly-collaborative application is an application that induces a collaboration graph with a high degree of collaboration $'PR \gg \frac{1}{N_{org}}$

3.2. Online-memory Scalability Problem

We assume that in a multi-tenant authorization service, all RBAC rules are stored centrally in a centralized rule store. Both inter- and intra-domain rules are stored in the centralized store. However, we differentiate between rules that need to be stored in **online** versus **off-line** stores. Online rules are searched with every authorization request, whereas off-line rules are only accessed when new rules are to be added or when old rules to be modified or deleted.

We define the ***online-memory scalability problem*** for a multi-tenant RBAC authorization service as the problem of minimizing the number of rule tuples that need to be online that is, searched for every authorization request.

The RBAC multi-tenant authorization system in [10] is an example of a centralized RBAC rule store. The system stores rule tuples for every shared resource in an organization, and these tuples are signed by an issuer–usually the organization's administrator. However, in a highly-collaborative environment, the number of inter-domain rules will be massive, and all these rules need to be stored online and processed by the authorization system for every authorization request.

If the system in [10] is to scale to increasing number of collaborative organizations, the number of rules would increase in two dimensions. The first dimension is the number of intra-domain RBAC tuples, which take the form *(role, object, permission, service)*, which means that *role* has *permission* to access *object* through *service*. The second dimension is the number of inter-domain rules, which take the form *(issuer, org1, role, org2, object, permission, service)*, which means that *issuer* has granted *role* of organization *org1* a certain *permission* on *object* in organization *org2* and this privilege is activated only through a certain *service*.

For example, consider two collaborating organizations, one with 20 resources r1 from *(1 to 20)*, the first 5 of them are shared and 3 roles *(i1 to i3)* and the other with 25 resources r2 from *(1 to 25)*, the first 10 of them are shared, and 4 roles *(j1 to j4)*. Every role in each organization has privileges to access one resource in its own organization and three resources in the other organization. Thus, the total number of intra-domain RBAC rules is 3+4 = 7 (e.g., (i1, r1 → 3) and (j4, r2 → 5)). The total number of inter-domain rules is 3×3+4=3 = 21 rules (e.g., (i3, r2 → 4) and (j4, r1 → 5)).

For a highly-collaborative application, a multi-tenant authorization service like [10] with a central RBAC rule store would suffer from quadratic (in number of organizations) number of rules in its online rule store. Indeed, the number of rules stored is $L+R = N_{org} \times N_{object} \times N_{role}('PL + (N_{org} - 1) \times 'PR) = O(N^2_{org})$.

3.3. Solution Highlights

In this paper, we address the online-memory scalability problem in a highly collaborative cloud environment, and we propose to replace the inter-domain rule tuples with a new rule store that contains role mapping [7], [8] tuples of the form *(in, jm)* indicating that role *i* in organization n maps to role *j* in organization *m*. A role *i* in organization *n* maps to role *j* in organization *m* if and only if role *i* is to be granted all privileges of role *j* within *j*'s local organization *m*.

Figure 1 highlights the basic idea behind our proposed solution to the online-memory scalability problem. Whereas, in a multi-tenant authorization service (e.g., [10]), all RBAC authorization rules are stored online (i.e., searchable by every authorization request). Our highly-collaborative authorization service (HCAS) replaces the quadratic inter-domain rules with a set of role mapping tuples whose number slightly decreases

with the degree of collaboration for many practical scenarios. The inter-domain RBAC rule set if moved to off-line storage (shaded in the figure), which is accessed by domain administrators for adding, deleting, or modifying the inter-domain rules.

3.4. Highly-Collaborative Authorization Service (HCAS)

This section describes the architecture and operation of our highly-collaborative authorization service. HCAS employs role mapping to reduce the size of online RBAC rule store by replacing the inter-domain rule tuples by role-mapping tuples. HCAS is composed of four main parts: an authorization service, a matching service, a mapping service, and off-line and online rule stores (Figure 2).

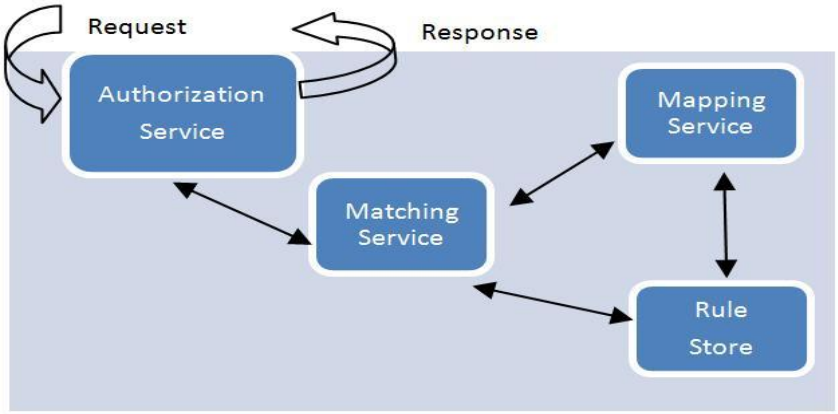

Fig. 2. HCAS System Architecture. HCAS is composed of four main parts: an authorization service (AS), a matching service (MX), a mapping service (MS), and off-line and online rule stores (RS).

3.4.1. HCAS Architecture

The **authorization service (AS)** is responsible for receiving authorization requests from the collaborative application's services and replying with granting or denying access. Each authorization request contains six parameters: requester organization ID, requester role, target organization ID, target resource ID, and requested permissions. The authorization service checks for hits in an internal request-response **cache**; otherwise, it forwards the request to the matching service MX. The **matching service (MX)** in turn looks up (in its online role-mapping tuple store) all rule tuples that indicate that the requester role has been mapped to roles in the target organization. For each such tuple, MX matches the permissions of the target organization's role with the requested permissions. If a match is found, MX replies back with access granted; otherwise MX replies with access-denied.

The **mapping service (MS)** is responsible for running the role-mapping algorithm (described in the next subsection) to deduce role-mapping rules from the inter-domain RBAC rules. The resulted role-mapping tuples are stored in the online rule store, whereas the inter-domain rules are stored in the offline rule store. MS runs with every insertion, deletion, or modification of an RBAC intra- or inter-domain rule. In each run of MS, the cache in AS is checked for rules to be invalidated. Finally, the **rule store (RS)** is divided into online and offline parts. The online rule store contains the intra-domain RBAC rules and the role-mapping rules. The off-line rule store contains the inter-domain RBAC rules.

3.4.2. Role Mapping Algorithm

The general role mapping problem has been shown to be an NP-hard problem and is similar in structure to the minimum set-cover problem [7], [8]. In HCAS, we adopted a greedy approach to solve the role-mapping problem. Our role-mapping algorithm is listed in Figure 3, and the notation used is summarized in Table I. The algorithm iterates over all ordered pairs of collaborating organizations.

For each pair of organizations *(host, guest)*, the algorithm maps each role in guest that has at least one inter-domain edge to a resource in host to one or more roles in host. To ensure correctness, two roles *(j, i)*, where *i* is a role in host and *j* a role in guest, are mapped if and only if all local privileges granted to role *i* are also granted (remotely) to role *j*. That is, the set of local resources mapped to role *i* in host is a subset of the set of remote resources mapped to role *j* from host's resources. That is, $\{oi : oi \in Ohost \text{ and } (i, oi) \in E\} \subseteq \{oj : oj \in Ohost \text{ and } (j, oj) \in E\}$. If such a role *i* does not exist in organization host, then two solutions are attempted.

First, role splitting is attempted. If there exists a "wider" role *î* from host, î is split into two roles *î'* and *î''*, one of them, *î'*, is locally mapped to the same set of resources that are mapped to role *j* from host. That is, {oj : oj Є Ohost and (j, oj) Є E} = {oî' : oî' Є Ohost and (î', oî') Є E} ⊂ {oî : oî Є Ohost and (î, oî) Є E}. Second, role insertion is attempted. If there was no wider role *î*, a new role i' is introduced to cover the needed resources, that is, {o_j : o_j Є O_{host} and (j, o_j) Є E} = {$o_{\hat{i}}$: $o_{\hat{i}}$ Є O_{host} and (î, oî) Є E}.

Table 1. Algorithm Notation

Symbol	Meaning
I	Set of roles in the host organization
O	Set of resources in the host organization
P	Set of access permissions (e.g., read, write, execute)
A	Set of role access rights in the host organization A ⊂ I × O × P
J	Set of roles in the guest organization
Req	Set of access rights granted to the guest organization by the host organization Req Є J × O × P
Î	Set of new roles to be added in the host organization after role mapping
Â	Set of access rights of the new roles Î that requested by the guest organization
M	Set of role mapping records between the guest organization and the host organization M Є J × (I U Î)

For each pair of organizations, the algorithm iterates over all guest roles *J* and for each guest role *j*, it iterates over all host roles *I*. For each nested-loop iteration *(i, j)*, we have three cases: (1) if all resources in current role *i* of *I* are needed by guest role *j*, then role *i* is mapped to role *j* by inserting the tuple *(j, i)* into the role-mapping rule store *M*; (2) if some resources in current role *i* — but not all — are needed by role *j*, then role *i* is split to role *î* with just the resources needed by role *j*, the new role *î* is inserted to a list of new roles *Î*, the privileges mapped to *î* are inserted into a new list of local RBAC rules *Â*, and the tuple *(j, î)* is inserted into *M*; (3) if some resources needed by role *j* are still not covered, a new role *role* is inserted with the remaining resources and added to list *Î*, its privileges added to *Â*, and the tuple *(j, role)* is inserted into *M*. Finally, we insert the new roles Î to the local RBAC rule store to keep track of the new roles, insert Â to the intra-domain rule store, and insert *M* to the role-mapping rule store. In HCAS, the inter-domain rule set *Req* between each two collaborating organizations is replaced by the set of role-mapping rules and the set of new roles added due to splitting or role insertion. That is, |Req| tuples are replaced by |M U Î U Â|.

4. RESULTS AND ANALYSIS

Our HCAS system aims at reducing the amount of rule tuples stored online, that is, in the path of each and every authorization request. The main tenet of HCAS is the adoption of the well-known role mapping technique to replace the inter-domain RBAC rule tuples by role mapping tuples. In this section we present and analyze the results of a simulation-based study to measure the effect of the degree of collaboration on the size of the online memory in both the traditional RBAC (i.e., role-object tuples) and our HCAS rule stores. In summary, our results show that the size of the online-memory in our HCAS is less in many highly-collaborative cloud applications. For applications with a low degree of collaboration or with very few objects per role, role-to-object mapping induces the same or slightly less online rules than HCAS.

4.1. Experiment Setup

The mapping service (the role mapping algorithm described in Figure 3) of HCAS has been implemented in Java, and we compared HCAS role mapping approach to the traditional role-to-object mapping. We have adopted a simple RBAC model of 5-tuple rules *(role, org1, object, org2, permission)* [10], where each rule indicates that role *role* from organization *org1* can access resource *object* in organization *org2* with permission *permission*. We generated a collaboration graph between two organizations, host and guest, as follows. For intra-domain edges, the number of resources assigned to each role was generated out of a normal distribution, and the IDs of the resources assigned to every role were generated uniformly at random from the set of local resource IDs. The inter-domain edges were generated in a similar way. We ran the experiments for a range of mean values for the intra-domain and inter-domain normal distributions, whereby the standard deviation is always 10% of the mean. Each experiment was run 10 times and the rule store sizes as described in Section IV were computed in each run. The average savings ratio in the size of online rule store was calculated as (# tuples in role-to-object RBAC / #tuples in HCAS) and reported in the figures below. Table II summarizes the parameter values for the experiments shown in this paper.

Table 2. Algorithm Notation

Parameter	Value
# roles in host organization	5, 15
# roles for guest organization	5, 20
# resources in host organization	20, 500
# resources in guest organization	20, 500
Number of resources per role (inter- and intra-domain)	mean = 1 - 500; standard deviation = 10% of mean

Input:
I: the set of roles in the host organization
R: the set of resources of the host organization
P: the set of access permissions (e.g., read, write, execute)
$A \subseteq$ IxRxP: the set of role access rights in the host organization
J: the set of roles in the requester organization
Req Є JxRxP: the set of access rights granted to the requester organization
Output:
Î : the set of new roles to be added in the host organization
Â : the set of access rights of the new roles Î
M ⊆ Jx(I ∪ Î): role mapping between the requester organization and the host organization
begin

```
M = Î = Â = ∅
for each role j in J
        K = ∅
        for each role i in I
                Temp = ∅
                for each (resource, permission) pair (r, p) in Req(j)
                        if( (r,p) Є A(i))
                            add tuple (r, p) to Temp
                        add tuple (r, p) to K
                end //loop on Req(j)
                if(|Temp| < |A(i)| ) then //role i not fully mapped
                    add i' to Î
                    for each (r, p) in Temp
                        add (i', r, p) to Â
                    end
                    add (j, i') to M
                else // |Temp(i)| = |A(i)|
                    add (j, i) to M
                end if

        if  K.size = Req(j) then //check if role j completely covered, if so, break
            break;
        end //loop on i
  if  K.size = Req(j) then //check if role j completely covered
      role = Req(j) – K
      add role to Î
      for each (r, p) in role
          add (role, r, p) to Â
      end
      add (j, role) to M
end //loop on j
return M
```

Fig. 3. The greedy role-mapping algorithm used in HCAS.

4.2. Low-collaboration Scenario

In the first set of experiments, we modeled a collaboration graph with a low degree of collaboration. More specifically, both collaborating organizations had five roles and twenty resources, and we varied the mean number of intra-domain and inter-domain resources per role between one and five inclusive. A mean of three, for example, means that each role was assigned an average of three intra-domain resources and and an average of three inter-domain resources. Figure 4 depicts the savings ratio with varying average number of resources per role. As the figure shows, there is no significant difference between the two methodologies. For a few cases, HCAS was slightly outperformed by role-to-object mapping because the overhead of role splitting was more than the saving due to role mapping.

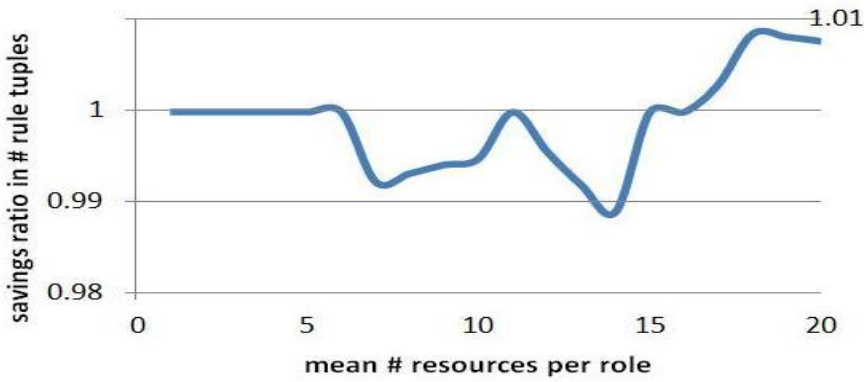

Fig. 4. Low-collaboration scenario. There was no significant difference between our HCAS and role-to-object mapping. For a few cases, however, HCAS was slightly outperformed by role-to-object mapping.

4.2. Low-collaboration Scenario

In the second set of experiments, we modeled a collaboration graph with a high degree of collaboration. More specifically, the host organization had 15 roles, and the guest organization had 20 roles. Both organizations have 500 resources, and we varied the mean number of intra-domain and inter-domain resources per role between 1 and 500 inclusive. Figure 5 shows that the savings ratio is super-linear in most of the simulated range of number of resources per role. It starts with a sub-linear behavior that turns into super-linear increase at mean = 70. Further investigation revealed that whereas the number of rule tuples in role-to-object mapping increased linearly with increasing number of resources per role, the number of tuples in HCAS started with a linear increase followed by a slight decrease at a turning point of mean = 70. With increasing number of intra- and inter-domain edges per role, it becomes more likely to match guest roles into host roles without the need to split host roles and without the need to create new host roles to cover unassigned objects, reducing the number of role-mapping rules.

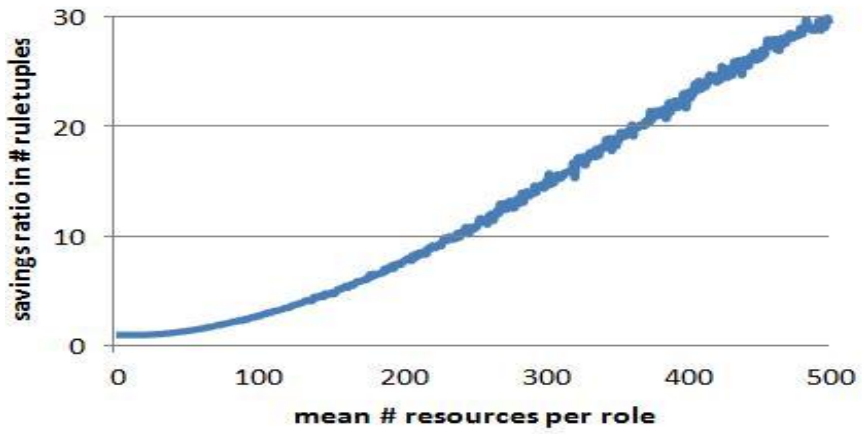

Fig. 5. Highly-collaborative scenario. HCAS achieved a super-linear savings ratio in most of the simulated range of number of resources per role.

5. CONCLUSION

Federated authentication and authorization systems have been gaining more ground recently due in part to the increasing nature of collaboration and resource sharing across administrative domains in today's applications. A corner stone, besides security, to successful adoption of such systems is performance. This work showed that it is possible to design multi-tenant RBAC services that scale, in terms of required online memory size, to the degree of collaboration among participant organizations. The key idea is to use the role mapping technique, which we found to result in a more stable online storage footprint than role-to-object mapping. Based on this observation, we have proposed the highly-collaborative authorization service (HCAS), described its architecture and operation, and evaluated its memory saving through simulation experiments.

An immediate next step to this work is the evaluation of a web service-based prototype of all HCAS component services: the authorization service (AS), the matching service (MX), the mapping service (MS), and the rule store (RS). Further investigation is warranted to efficient cache invalidation algorithms for the AS cache, efficient role-mapping algorithms with an incremental update mechanism, analysis of the characteristics of real-world collaboration graphs for current and projected collaborative cloud applications, and, last but not least, an adaptive feedback-based mechanism for switching between role-mapping and regular role-to-object mapping based on the observed degree of collaboration.

REFERENCES

[1] P. Mell and T. Grance. (2009, Jun.) Cloud Computing Definition. Last accessed on March 30, 2012. [Online]. Available: http://csrc.nist.gov/groups/SNS/cloud-computing/index.html

[2] L. Fair. (2010, Mar.) FOSE session: Security and risk management with cloud computing. Last accessed on March 30, 2012. [Online]. Available: http://blog.sciencelogic.com/fose-session-securityand-risk-management-with-cloud-computing/03/2010

[3] R. Zhang and L. Liu, "Security models and requirements for healthcare application clouds," in *Proceedings of the 2010 IEEE 3rd International Conference on Cloud Computing*, ser. CLOUD '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 268–275. [Online]. Available: http://dx.doi.org/10.1109/CLOUD.2010.62

[4] J. Howell and D. Kotz, "End-to-end authorization," in *Proceedings of the 4th conference on Symposium on Operating System Design & Implementation - Volume 4*, ser. OSDI'00. Berkeley, CA, USA: USENIX Association, 2000, pp. 11–11. [Online]. Available: http://dl.acm.org/citation.cfm?id=1251229.1251240

[5] H. Takabi, J. B. D. Joshi, and G.-J. Ahn, "Security and privacy challenges in cloud computing environments," *IEEE Security and Privacy*, vol. 8, no. 6, pp. 24–31, Nov. 2010. [Online]. Available: http://dx.doi.org/10.1109/MSP.2010.186

[6] R. Sandhu, D. Ferraiolo, and R. Kuhn, "The NIST model for role-based access control: towards a unified standard," in *Proceedings of the fifth ACM workshop on Role-based access control*, ser. RBAC '00. New York, NY, USA: ACM, 2000, pp. 47–63. [Online]. Available: http://doi.acm.org/10.1145/344287.344301

[7] L. Chen and J. Crampton, "Inter-domain role mapping and least privilege," in *Proceedings of the 12th ACM symposium on Access control models and technologies*, ser. SACMAT '07. New York, NY, USA: ACM, 2007, pp. 157–162. [Online]. Available: http://doi.acm.org/10.1145/1266840.1266866

[8] J. Hu, R. Li, and Z. Lu, "On role mappings for RBAC-based secure interoperation," in *Proceedings of the 2009 Third International Conference on Network and System Security*, ser. NSS '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 270–277. [Online]. Available: http://dx.doi.org/10.1109/NSS.2009.76

[9] G. Geethakumari, A. Negi, and V. N. Sastry, "A cross - domain role mapping and authorization framework for RBAC in grid systems," *IJCSA*, vol. 6, no. 1, pp. 1–12, 2009.

[10] J. M. A. Calero, N. Edwards, J. Kirschnick, L. Wilcock, and M. Wray, "Toward a multi-tenancy authorization system for cloud services," *IEEE Security and Privacy*, vol. 8, no. 6, pp. 48–55, Nov. 2010. [Online]. Available: http://dx.doi.org/10.1109/MSP.2010.194

[11] M. A. P. Leandro, T. J. Nascimento, D. R. dos Santos, C. M. Westphall, and C. B. Westphall, "Multi-tenancy authorization system with federated identity for cloud-based environments using shibboleth," in *ICN 2012,The Eleventh International Conference on Networks*. IARIA, Feb 2012, pp. 88–93.

[12] Shibboleth. Last accessed on March 30, 2012. [Online]. Available: http://shibboleth.internet2.edu/

BIOGRAPHY OF AUTHORS

Samy Gerges is a Master student in department of computer science, faculty of computers and information, Cairo University since 2008 in fields of Cloud Computing, SOA and Secuirty. Samy also working as a software engineer since 2007 in IT field and currently he is working at HP enterprise services Egypt as Senior service information developer.

Dr. Sherif Khattab is an assistant professor in the department of computer science, faculty of computers and information, Cairo University. Sherif obtained his PhD in computer science from the department of computer science, university of pittsburgh in 2008 under supervision of Rami Melhem and Daniel Mossé. He was a member of the Network Security group. Sherif received a Bachelor's degree in computer engineering from Cairo University in 1998 and a M.Sc. degree in computer science from the University of Pittsburgh in 2004.

Prof. Hesham A. Hassan is an Egyptian researcher born in Cairo in 1953. He sham's educational background is as follows: B.Sc in Agriculture, Cairo University, Egypt in 1975. Postgraduate diploma in computer science, from ISSR, Cairo University, Egypt in 1984. M.Sc I computer science, from ISSR, Cairo university, Egypt, in 1989. PhD in computer science from ISSR, Cairo University (dual supervision Sweden/Egypt) in 1995. He is now a PROFESSOR and HEAD of computer science department at the faculty of computers and Information, Cairo university. He is also IT Consultant at Central Laboratory of Agricultural Expert System, National Agricultural Research Center. He has published over than 51 research papers in international journals, and conference proceedings. He has served member of steering committees and program committees of several national conferences. Hesham has supervised over 27 PhD and M. Sc theses. Prof. Hesham interests are Knowledge modeling, sharing and reuse, Intelligent information retrieval, Intelligent Tutoring systems, Software Engineering. Cloud Computing and Service Oriented Architecture (SOA).

Prof. Fatma A. Omara is Professor in the Computer Science Department and Vice Dean for Community Affairs and Development in the Faculty of Computers and Information, Cairo University. She has published over 45 research papers in prestigious international journals, and conference proceedings. She has served as Chairman and member of Steering Committees and Program Committees of several national Conferences. She has supervised over 30 PhD and M.Sc thesis. Prof. Omara is a member of the IEEE and the IEEE Computer Society. Prof. Omara interests are Parallel and Distributing Computing, Parallel Processing, Distributed Operating Systems, High Performance Computing, Cluster, Grid, and Cloud Computing.