

Enhancing highly-collaborative access control system using a new role-mapping algorithm

Doaa Abdelfattah, Hesham A. Hassan, Fatma A. Omara

Department of Computer Science, Faculty of Computers and Artificial Intelligence, Cairo University, Cairo, Egypt

Article Info

Article history:

Received Feb 11, 2021

Revised Dec 24, 2021

Accepted Jan 10, 2022

Keywords:

Authorization

Cloud computing

Role based access control

Role-mapping

Security

ABSTRACT

The collaboration among different organizations is considered one of the main benefits of moving applications and services to a cloud computing environment. Unfortunately, this collaboration raises many challenges such as the access of sensitive resources by unauthorized people. Usually, role based access-control (RBAC) Model is deployed in large organizations. The work in this paper is mainly considering the authorization scalability problem, which comes out due to the increase of shared resources and/or the number of collaborating organizations in the same cloud environment. Therefore, this paper proposes replacing the cross-domain RBAC rules with role-to-role (RTR) mapping rules among all organizations. The RTR mapping rules are generated using a newly proposed role-mapping algorithm. A comparative study has been performed to evaluate the performance of the proposed algorithm with concerning the rule-store size and the authorization response time. According to the results, it is found that the proposed algorithm achieves more saving in the number of stored role-mapping rules which minimizes the rule-store size and reduces the authorization response time. Additionally, the RTR model using the proposed algorithm has been implemented by applying a concurrent approach to achieve more saving in the authorization response time. Therefore, it would be suitable in highly-collaborative cloud environments.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Doaa Abdelfattah

Department of Computer Science, Faculty of Computers and Artificial Intelligence, Cairo University

1 Gamaa Street, Giza, Cairo, Egypt

Email: doaaabdelfattah94@gmail.com

1. INTRODUCTION

Many cloud applications involve multiple collaborating organizations to share data, services, and resources [1]. For example, the federal United States Government cloud has different agencies that collaborate and share sensitive data, such as criminals' information and national security data [2]. Also, healthcare institutions upload patients' sensitive data onto the cloud and apply collaborating rules over shared data and services, such that patients can get their treatment across different healthcare branches, or even across different healthcare institutions [3]. Therefore, these examples and many others inspired us to go deeply into organizations' collaboration in cloud computing, and how these organizations control accessing the shared resources among each other.

A collaborative cloud computing environment means that many organizations host their services or resources on the same cloud computing environment. Each organization has its authorization mechanism and uses an access control model in its authorization. In a collaborative environment, all organizations communicate together and share resources. For each resource or service shared by any of the collaborative organizations, there are rules, which centrally stored in a third-party service, specify the permissions and roles for allowing

access to the shared resource. These rules are called cross-domain rules, which are stored centrally as an authorization service. A trusted authorization provider saves all rules of each organization and the cross-domain statements in a central rule store [4].

The authorization function is concerned with specifying access rights to certain resources [5]. Therefore, authorization aims to allow the authorized people to access certain resources. For example, an employee in a company can access his timesheet and cannot access any other employees' timesheets. While the system manager can access his employees' timesheets, but he cannot modify them, or view timesheets of employees who are not in his team. Therefore, the authorization procedure grants access to a specific resource if there is a stored rule specifying that the requester can access this resource with specific permission. Otherwise, the authorization procedure denies access to the requested resource [6].

Security is considered a major issue that affects collaborative cloud applications [7]. New security challenges face the cloud providers and collaborating organizations when they federate their authentication and authorization systems and allow cross-domain security mechanisms to protect the shared sensitive data from unauthorized access [8], [9]. In high collaboration environments, the rule store size is increased quadratically with increasing the number of the collaborating organizations. Also, the rule store size is affected by the number of shared resources of each organization. Therefore, the critical factor of a successful federated access control system is the performance of rule lookup, which depends on the scalability of the rule store size. This size depends on the number of shared resources among the collaborating organizations and the number of collaborating organizations. Therefore, the aim of this paper is to address the online-memory scalability problem.

The online-memory scalability problem for a multi-tenant authorization service has been defined as the problem of minimizing the number of rule tuples that need to be online and searched for every authorization request [10]. The goal of the work in this paper is to enlarge the scale of the collaboration along two dimensions. The first dimension is to enlarge the scale of shared resources among collaborating organizations, while the second one is to increase the number of collaborating organizations in the same environment. Therefore, a new algorithm has been proposed for the federated multi-tenant authorization services, which aims to reduce the number of rule tuples in the central rule store and improve the authorization request's response time.

The remaining sections of the paper are organized: section 2 discusses the related work. Section 3 describes the principles of the role-based access control (RBAC) system, its mapping models, and the SplitMap role-mapping algorithm. Section 4 describes the proposed DirectMap role-mapping algorithm. Section 5 presents the experimental evaluation of the proposed DirectMap role-mapping algorithm concerning the rule store size and the authorization response time compared to other existed algorithms. Section 6 introduces a concurrent approach for implementing the role-to-role (RTR) model using the proposed DirectMap role-mapping algorithm and also presents the experimental evaluation of that approach concerning the authorization response time. Section 7 concludes the paper.

2. RELATED WORK

In this section, the related work of Cloud computing access control services, multi-tenancy authorization models, and role-mapping approaches are discussed. Rao *et al.* [11] introduced algebra for fine-grained integration of language-independent policies which integrates access control policies of collaborating parties. This algebra can support the specification of a large variety of integration constraints specified in extensible access control markup language (XACML). Also, the authors proposed a framework that uses algebra for the fine-grained integration of policies. Unfortunately, they did not consider the impact of the presence of obligations when integrating policies using fine-grained integration algebra (FIA). Also, they did not clarify how to guarantee that a given algebraic expression will behave as expected.

Alansari *et al.* [12] proposed a novel identity and access management system for cloud federations. This system applies attribute-based access control policies on the federated organizations' data in a privacy-preserving manner. It grants users to access federated data when their identity attributes match the policies without revealing their attributes to the federated organization which owns the data. The integrity of the policy evaluation process is guaranteed by using blockchain technology and Intel software guard extensions (SGX) trusted hardware. The authors did not produce an extensive evaluation of the system concerning the performance and the cost of execution.

Younis *et al.* [13] proposed a novel access control model for cloud computing. It utilizes temporal (time and location) and delegation constraints. In that model, users are assigned to security domains that relate to their roles and actual jobs. Every role within the model is assigned the relevant tasks that allow them to practice their roles. The model secures access and flow of data by marking the data with security labels, which state the data sensitivity. Each task has a security classification for accessing the data or assets, and the

certain needed permissions for accomplishing this task. But this model needs a risk engine and its components, which are used to deal with dynamic behaviors.

Tang *et al.* [14] proposed a multi-tenant role-based access control (RBAC) Model for collaborative cloud environments. In this model, an authorization based on RBAC among collaborative organizations was built. The model architecture was built on a centralized policy decision point (PDP) and multiple policy enforcement points (PEPs), in which every organization has a PEP that contains all authorization request requirements and asks the centralized PDP to decide granting or denying the request. The proposed model solves the scalability problem among collaborating organizations which raised by increasing number of collaborating organizations and increasing number of shared resources by increasing- in run time- the number of tenants (i.e., increasing the number of PEPs), or by increasing PDP RAM and CPU cores taking the advantage of cloud platform scalability. On the other hand, the multi-tenant RBAC policies with different expressive power in the authorization as a service (AaaS) incur various policy evaluation delays.

Leandro *et al.* [15] proposed and implemented a multi-tenancy authorization system using Shibboleth [16]. Authorization is managed by the application (a collaborative web authoring tool), and Shibboleth is used to implement cross-domain identity management without a trusted third party. The 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 decides for granting or denying the request. Some specialized features for an identity management system include a single sign-on (SSO), where a user does not need to sign on many times to call various applications, and able to reuse the authenticated status of a preceding application in the same session. The risk of security is being considered as one of the main drawbacks of the Shibboleth system. The SSO interface creates a problem related to security as it is attached to both the manager and client's sides. By using this weak access system, various threats like malicious software, deviant, programs, and bad-bots can cause unwanted access to the secured resources.

Calero *et al.* [17] introduced a multi-tenancy authorization model suitable for cloud computing. This model federates the management of hierarchical role-based access control for path-based object hierarchies. Each organization of the collaborating organizations should represent all roles with all resources that need to be accessed locally or globally by the other collaborating organizations in statements (tuples) signed by an issuer from the organization hosting the resources. These tuples are stored in a trusted central database. For a highly collaborative application, a multi-tenant authorization service with a central RBAC rule store would suffer from a quadratic (in several organizations) number of rules in its online rule store.

Gopalan *et al.* [4] proposed a scalable role-mapping approach based on ranking roles and resources with specific scores and comparing the role and resource scores to decide request granting or denial. The scores are granted by the authorization server of each domain and 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.

Chen and Crampton [18] proposed an inter-domain role-mapping technique based on the principle of least privilege. They proposed a minimal cardinality for a role across a domain to avoid misuse of access. Their approach was trying to map already existing roles without creating new roles. This approach might cause losing some of the desired privileges.

Gerges *et al.* [10] proposed a scalable multi-tenant authorization system. This system replaces the role-to-object (RTO) mapping with role-to-role (RTR) mapping using the SplitMap role-mapping algorithm. This algorithm has a limitation of the cost of authorization request's response time. According to the work in this paper, this system has been enhanced by introducing a new role-mapping algorithm which is called DirectMap to save the authorization response time and achieve more reduction for the online rule store size in highly-collaborative environment.

3. ROLE-BASED ACCRSS CONTROL MODEL

Role-based access control (RBAC) model is considered one of the most well-known access control models that used in the most of the federated multi-tenant authorization services [19], [20]. According to the work in this paper, RTR model using the proposed DirectMap role-mapping algorithm is introduced. To evaluate the performance of our work, a comparative study is performed between the existed RBAC model and RTR model using the proposed DirectMap algorithm. Therefore, the RBAC model will be discussed in this section.

3.1. Role-to-object (RTO) mapping model

RBAC model is based on the RTO mapping technique which means that users can access resources or objects based on the roles they have been assigned [21]. This can be achieved by mapping the permissions

for certain resources (objects) to certain roles and then the users are assigned the roles and hence permitted to access the resources, as illustrated in Figure 1.

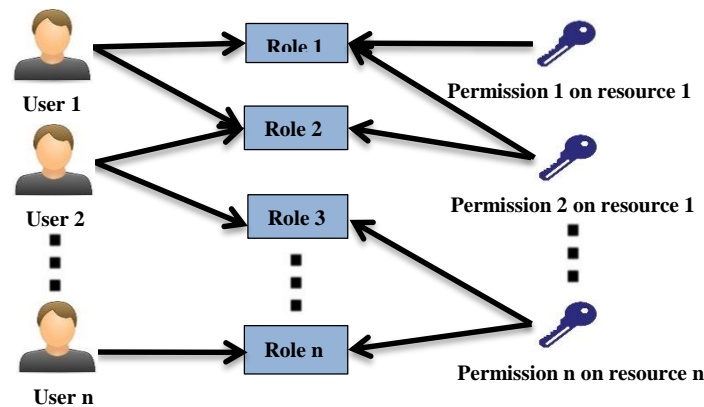


Figure 1. Principles of role-based access control model

As shown in Figure 1, a user can have multiple roles while each role can be assigned with multiple permissions on multiple resources. A request to access a certain resource with specific permission is granted or denied after a lookup for a rule that allows the requested permission. In the collaborative cloud environments, each collaborating organization has its security mechanism and uses the RBAC model for authorization. All the collaborating organizations communicate together and share some resources. For each resource shared by any organization to another, there is a rule stored in a third-party service that specifies the role and permission that can access that shared resource. These rules are called cross-domain rules. A trusted authorization provider stores all rules of each organization and the cross-domain rules in a central rule store. In the highly collaborative environments, the number of stored rules can be massive. So, the rule store size can be increased quadratically with increasing the number of collaborating organizations and/or the number of shared resources by each organization.

The RBAC multi-tenant authorization system has a centralized RBAC rule store [17]. It stores rule tuples for each shared resource in an organization, and these tuples are signed by an issuer—who is usually the administrator of the organization. However, in a highly collaborative environment, the number of cross-domain rules will be huge, and all these rules need to be stored online and processed by the authorization system for every authorization request. So, by increasing the number of collaborative organizations, the number of rules stored centrally will be increased in two dimensions. The first one is the number of local RBAC tuples for each organization, which is represented as (role, object, permission, service), which means that role has permission to access the object through service. The second dimension is the number of cross-domain rules, which is represented as (issuer, org1, role, org2, object, permission, service), which means that the issuer has granted the role of organization org1 a specified permission on the object in organization org2, and this privilege is activated only through a specified service. For a highly collaborative application, a multi-tenant authorization service with a central RBAC rule store would suffer from a quadratic number of rules (i.e., number of organizations) in its online rule store.

3.2. Role-to-role (RTR) mapping model

By increasing the rule-store size in highly collaborative environments, the time of rule lookup would be affected, which causes increasing the authorization request's response time. To solve this problem, the idea of role-mapping (i.e., role-to-role mapping) technique came out. It maps user roles in each organization to the roles in the host organization, which owns the shared resource [10].

The role-to-role (RTR) mapping technique replaces the cross-domain rule tuples with role-mapping rule tuples. For example, assume the role-mapping rule (i_n, j_m) . This mapping rule indicates that role i in organization n maps to role j in organization m , as shown in Figure 2(a). This mapping process is completed if and only if role i granted all privileges of role j within j 's local organization m . As shown in Figure 2(a), user X has a role i within organization n and role i maps to role j in organization m . therefore user X inherits all the privileges assigned to role j . On the opposite, according to the RTO model, role i in organization n maps to the privileges on certain shared resources in organization m . Therefore, a user Y has the privileges to access the mapped resources with the assigned permissions, as illustrated in Figure 2(b).

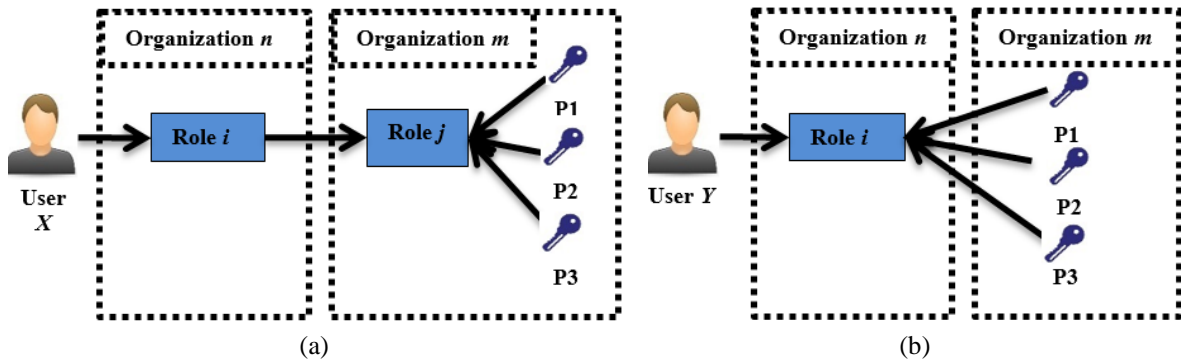


Figure 2. Comparison between (a) role-to-role mapping and (b) role-to-object mapping

By comparing the number of rule tuples in the mentioned example, it is found that there is only one stored mapping rule of (i_n, j_m) in the RTR mapping model, while in the RTO mapping model, there are three stored mapping rules: $(i_n, P1_m)$, $(i_n, P2_m)$, and $(i_n, P3_m)$. Therefore, RTR mapping has the potential for reducing the number of rule tuples that are searched in every authorization request and consequently enhancing the authorization response time.

RTR mapping tuples should be generated by a role-mapping algorithm. There are many algorithms have been proposed for role-mapping [4], [10], [18], [22]. According to the work in this paper, a new enhanced role-mapping algorithm, called DirectMap is introduced. To evaluate the performance of the proposed technique, a comparative study is performed to compare the rule store size and the authorization response time for the proposed role-mapping algorithm against the SplitMap role-mapping algorithm [10].

3.2.1. SplitMap role-mapping algorithm

SplitMap is a role-mapping algorithm that generates role-mapping rule tuples to be stored and searched for every authorization request [10]. This algorithm iterates over each ordered pair of all the collaborating organizations. For each pair of organizations (i.e., *host*, *guest*), where each role in *guest* which has at least one cross-domain edge to a resource in the *host*, is mapped to one or more roles in the *host*. For example, assume that there are two roles (i, j) , where i is a role in *host* and j is a role in *guest*. The mapping of such roles will be formed according to the following three cases:

- First case: if all local privileges are granted to role i and also granted (remotely) to role j , role i will be mapped directly to role j as shown in Figure 3(a). Therefore, the set of local resources' privileges of role i in the *host* could be a subset of the set of remote resources' privileges mapped to role j from the *host*'s privileges.
- Second case: the host role i splitting is attempted as shown in Figure 3(b). If i is a role that has resources' privileges more than the requested privileges from role j , then, a new role i' is created as a subset of role i . This new role i' is locally mapped to the same set of privileges that are mapped to role j . Then, role i' is mapped to role j .
- Third case: role insertion is attempted as shown in Figure 3(c). If all the *host* roles are checked and role j is not fully covered (i.e., there are one or more privileges of role j is not mapped to any *host* role), a new role i' is introduced to cover the needed privileges, and then it is mapped to role j .

All new invented roles and their implications on the privileges are stored centrally in the same database. So, all the updated data would be recognized through all the organizations. The pseudo-code and used notations of the SplitMap algorithm are listed in [10]. Although, this algorithm saves the online rule store size in high collaboration environments, it costs more time for the authorization request's response time than the RTO model. In addition, this algorithm would not be effective by considering the following scenario: Assume that each requested role will consume N distributed resources' privileges, so N splitting of the *host* roles will be needed (i.e., each resource's privilege will make split to one different role). Therefore, the number of roles in the host organization will increase by N roles. In this case, the online rule store of the RTO model and the RTR model will contain N number of rule tuples between the *guest* and the *host* organizations. Therefore, the RTR mapping model will behave exactly as the RTO mapping model.

On the other hand, increasing the number of tuples will affect the RTR mapping model by increasing the number of roles and its associated privileges in the local database of the host organization by N number of records due to the invention of N new roles. Therefore, in this case, by using the SplitMap algorithm, the RTO mapping model could behave better than the RTR mapping model concerning the size of the *host*'s local database.

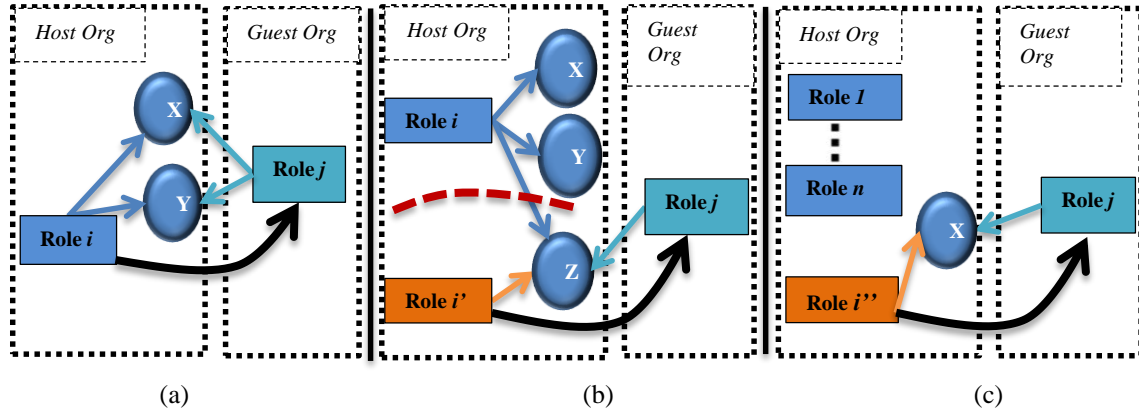


Figure 3. Principles of the SplitMap algorithm, (a) first case, (b) second, and (c) third

4. THE PROPOSED ALGORITHM

To enhance the performance of the SplitMap algorithm, a proposed role-mapping algorithm called DirectMap is introduced. The proposed algorithm aims to reduce the number of generated role-mapping tuples which reduces the online rule store size and enhances the authorization response time.

4.1. Formal definition

The basic sets which are used in the proposed DirectMap algorithm are listed in Table 1. The algorithm's formal definition is described:

- $\forall j \in J \rightarrow \exists Req(j) \subset O \times P$
For each guest role j in the guest organization, take its access rights which granted to the guest organization by the host organization.
- $\forall RP \subset Req(j) \rightarrow Temp = Temp \cup RP, RP: \langle r, p \rangle$
For each requested pair of (resource, permission) in role j , which is an access right on a certain resource with a specific permission, add it to an empty set, Temp.
- $\hat{I} = \hat{I} \cup i'$ where $i' : \text{New Role}$
Create a new role i' in the host organization and add it to \hat{I} set.
- $\forall RP \subset Temp \rightarrow \hat{A} = \hat{A} \cup \langle i', RP \rangle$
For each access right pair of (resource, permission) in Temp set, assign it to the new role i' and add them to \hat{A} set.
- $M = M \cup \langle j, i' \rangle$
Map the new host role i' to the guest role j and add this mapping rule to the set M.
Then, after iterating over all the guest roles, insert all the updated sets (\hat{I} , \hat{A} , and M) to the central Rule-Store.

Table 1. DirectMap algorithm notations

Symbol	Meaning
I	Set of roles in the host organization
O	Set of resources in the host organization
P	Set of access permissions in the host organization (e.g., read, write, execute)
A	Set of role access rights in the host organization $A \subset I \times O \times P$
J	Set of roles in the guest organization
Req	Set of access rights granted to the guest organization by the host organization $Req \subset J \times O \times P$
\hat{I}	Set of new roles to be added in the host organization after role mapping
\hat{A}	Set of access rights of the new roles \hat{I} that requested by the guest organization $\hat{A} \subset \hat{I} \times O \times P$
M	Set of role mapping rules between the guest organization and the host organization $M \subset J \times (I \cup \hat{I})$

The pseudo-code of the proposed DirectMap role-mapping algorithm is listed in Figure 4. The proposed DirectMap algorithm iterates over each ordered pair of all the collaborating organizations. For each pair of organizations (i.e., host, guest), the algorithm iterates over the roles of the guest organization.

As shown in Figure 5, assume that $Role_j$ is a role in the *guest* organization Org_x and requests to access some shared resources with certain permissions (i.e., P_1 and P_2) on the *host* organization Org_z .

Regardless of which roles these resources' permissions belong to, the DirectMap algorithm directly creates a new role $Role_{jx}$ in the *host* organization Org_z and assigns all requested resources' permissions to that role inside the *host* organization. Then, it maps $Role_{jx}$ to $Role_j$.

```

Input:
I: set of roles in host organization
O: set of objects in host organization
P: set of access permissions (e.g., read, write, execute)
A  $\subset$  IxOxP: set of role access rights in host organization
J: set of roles in guest organization
Req  $\subset$  JxOxP: set of access rights granted to the guest organization
Output:
 $\hat{I}$ : set of new roles to be added in the host organization
 $\hat{A}$ : set of access rights of the new roles  $\hat{I}$ 
M  $\subset$  Jx(I  $\cup$   $\hat{I}$ ): set of generated Role-Mapping rules between the guest organization and
the host organization

begin
M =  $\hat{I}$  =  $\hat{A}$  =  $\emptyset$ 
for each role j in J
Temp =  $\emptyset$ 
Req' = Req(j)
for each (resource, permission) pair (r, p) in Req'
add pair (r, p) to Temp
end //loop of access rights in Req'
add i' to  $\hat{I}$  //create new role i'
for each pair (r, p) in Temp
add (i', r, p) to  $\hat{A}$ 
end //loop of access rights in Temp
add (j, i') to M //Map role j to new role i'
end //loop on role j
end

```

Figure 4. Pseudo code of the proposed DirectMap algorithm

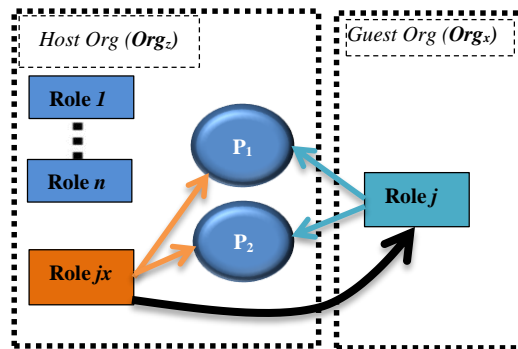


Figure 5. Principles of DirectMap algorithm

In other words, this algorithm always creates a new role in the *host* organization for each role in the *guest* organization. It assigns the requested resources' permissions of the *guest* role to the new *host* role and then directly maps the newly created role to the *guest* role. Assume that the *guest* organization has N number of roles. Then, N roles will be created in the *host* organization. If H is the number of all roles of the *host* organization, after the mapping, the number of *host* organization roles will be $(H+N)$ and the associated privileges will be created for the newly created N Roles.

4.2. Difference between SplitMap and DirectMap algorithms

The difference between role-mapping techniques of the SplitMap and DirectMap algorithms could be clarified by considering the following scenario. As shown in Figure 6, assume that a *guest* role (i.e., $Role_j$) requests to consume n resources' permissions (i.e., P_1 to P_n), these permissions are distributed over n *host* roles (i.e., $Role_1$ to $Role_n$).

By using the SplitMap role-mapping algorithm as shown in 7(a), each *host* role (from $Role_1$ to $Role_n$) that has requested resource permission is split. In other words, each resource's permission (from P_1 to P_n) will make split into one different new role (i.e., $Role_{1'}$ to $Role_{n'}$). Therefore, this algorithm creates n new roles in the *host* organization which increases its local database size. Then, the algorithm maps the n created roles to the guest role, $Role_j$. So, n role mapping tuples are generated which increases the online rule store size.

By considering $Role_1$ as an example to clarify the mapping technique, assume that $Role_1$ has resources' permissions X and P_1 as shown in Figure 7(a). While $Role_j$ has the cross-domain permission, P_1 . The SplitMap algorithm will split the permissions of $Role_1$ by assigning P_1 to a newly created role $Role_{1'}$. Then it maps $Role_{1'}$ to $Role_j$.

As shown in Figure 7(b), by using the proposed DirectMap role mapping algorithm, there is no need to split different roles, but it creates (for the guest role, $Role_j$) only one new role (i.e., $Role_{ix}$) in the *host* organization. The algorithm assigns all the requested resources' permissions (i.e., P_1 to P_n) to the newly created role, $Role_{ix}$, and maps it to the guest role, $Role_j$. So, only one role mapping tuple is generated which minimizes the online rule store size and the local database size of the *host* organization.

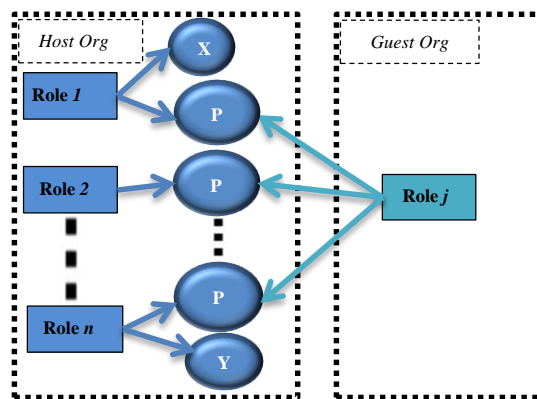


Figure 6. Distributed resources' privileges scenario

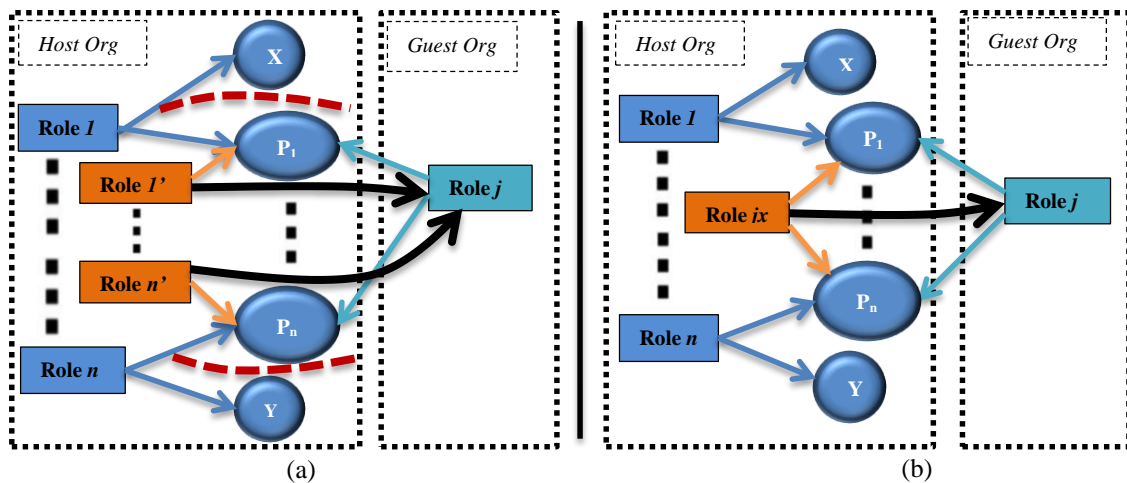


Figure 7. Difference between SplitMap and DirectMap algorithms (a) role-mapping using SplitMap and (b) role-mapping using DirectMap

5. RESULTS AND DISCUSSION

In this section, the performance evaluation of the proposed DirectMap algorithm is presented, starting by describing the experiments' setup. Then, the experiments will be introduced to measure the rule-store size and the authorization response time of the DirectMap algorithm against the SplitMap algorithm and against the traditional RBAC model (RTO).

5.1. Experiments setup

The experiments are implemented in Java using sequential query language (SQL) for communication with database management system (DBMS). MySQL server is used as the database server that contains the created inter-domain and intra-domain databases of communicating organizations. The intra-domain database, a central database schema that contains the local RBAC rules of all the collaborating organizations, is represented using five tables; *Employees* table contains all the organization employees' IDs, *Roles* table contains all organization roles, *Resources* table contains all resources, services, or data of the organization, *Permissions* table contains all the possible permissions over resources, and *Role-Resource-Permission* table contains relations over the *Roles*, *Resources* and *Permissions* tables. While the inter-domain database, a central database schema that contains relations among the collaborating organizations, is represented using three tables; *Organizations* table contains all the collaborating organizations that sharing resources in the collaborative environment, *Role-to-Object* table contains resources shared between every two organizations specifying the accessing permission (cross-domain rules), and *Role-to-Role* table contains all role-mapping rules between every two organizations.

The experiments are performed between two organizations, *host* and *guest*. For intra-domain rules, the number of resources assigned to each role is generated using the normal distribution, and the IDs of the resources assigned to every role are generated randomly from the local resource IDs set. The inter-domain rules are generated in a similar way as the intra-domain rules. The experiments are done for a range of mean values of the intra-domain and inter-domain, where the standard deviation is always 10% of the mean. The mean value ranged between 1 and the maximum number of resources in the host organization. For each mean value, the experiment will be executed. Table 2 summarizes the used parameters' values for the experiments in three different collaboration scales; low, intermediate, and high collaboration.

Table 2. Parameters' values for the experiments

Parameter	Values
# roles in host organization	5, 7, 15
# roles for guest organization	5, 10, 20
# resources in host organization	20, 250, 500
# resources in guest organization	20, 250, 500
#resources per role (inter- and intra-domain)	mean = 1 – 500 standard deviation = 10% of mean

5.2. Evaluation of RTR model using DirectMap vs SplitMap algorithms

A comparative study has been done to measure the effect of the collaboration degree on the size of the online rule-store and the authorization response time in the RTR model using the proposed DirectMap role-mapping algorithm compared with SplitMap role-mapping algorithm. The online rule-store size in the RTR model is defined as the number of role-mapping tuples among the collaborating organizations. While the average authorization response time is calculated by running 100 different authorization requests and measuring the authorization response time for each request. After that, the average response time of all the requests is calculated. To implement the experiment, three levels of simulated scenarios are used (low, middle, and high collaboration) using the parameters in Table 2. In each scenario, the rule-store size and the average authorization response time are measured.

5.2.1. Low-collaboration scenario

In the first scenario, a low degree of collaboration is considered using two collaborating organizations; both of them have five roles and twenty resources and using different values between one and twenty for the mean number of intra-domain and inter-domain resources per role. For example, if the mean value is three, it means that each role is assigned three intra-domain resources in average, and three inter-domain resources in average. Figure 8 depicts the number of rule tuples in the RTR model using SplitMap role-mapping algorithm and RTR model using the proposed DirectMap role-mapping algorithm for a range from one to twenty resources per role.

The average number of rule tuples in RTR model using SplitMap algorithm is 18, while the average number of rule tuples in RTR model using the proposed DirectMap algorithm is 5. As shown in Figure 8, by using SplitMap algorithm, the generated number of rules increases with increasing the number of resources per role. This is because the SplitMap algorithm maps each guest role to multiple split roles based on the number of resources per the guest's role. On the other hand, the DirectMap algorithm generates only 5 tuples with increasing the number of resources per role. This is because the guest organization has 5 roles and the algorithm maps each one of such roles to one newly created role in the host organization. Thus, the average number of generated rule tuples in the RTR model using DirectMap algorithm is lower than the average

number of generated rule tuples in the RTR model using SplitMap by 72.2%. Figure 9 depicts the average authorization response time (in milliseconds) for RTR model using SplitMap role-mapping algorithm and the proposed DirectMap role-mapping algorithm. As shown in Figure 9, the average response time of RTR model using SplitMap algorithm is 103 milliseconds. While by using DirectMap algorithm, the average response time is 70 milliseconds. Therefore, the proposed DirectMap algorithm achieves better response time than SplitMap algorithm by 32%.

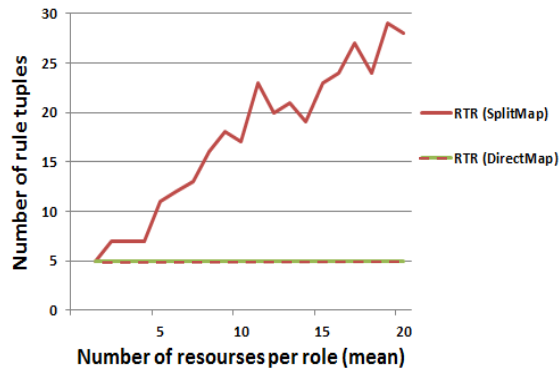


Figure 8. RTR using SplitMap vs RTR using DirectMap in low collaboration scenario

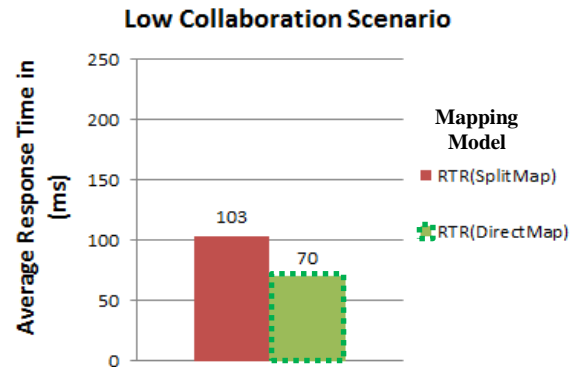


Figure 9. Average response time of RTR using SplitMap vs RTR using DirectMap in low collaboration scenario

5.2.2. Middle-collaboration scenario

In the second scenario, a Middle degree of collaboration is considered using two collaborating organizations; host organization has 7 roles, guest organization has 10 roles, both organizations have 250 resources, and the mean number of intra-domain and inter-domain resources per role is varied between 1 and 250 inclusive. Figure 10 depicts the number of generated role-mapping tuples in the RTR model using SplitMap role-mapping algorithm and the proposed DirectMap role-mapping algorithm for a range of mean values from 1 to 250 resources per role.

According to the results in Figure 10, it is found that the number of generated tuples using the proposed DirectMap algorithm is constant (i.e., for 10 roles in the guest organization, there are 10 mapping tuples in the online rule store). While, the number of generated tuples using SplitMap algorithm is increased by increasing the mean value (i.e., resources per role), until the maximum point which is 80 tuples for the mean value of 97 resources per role. Therefore, by increasing the mean value, the number of tuples does not exceed the maximum value (i.e., 80). For the maximum mean value (i.e., 250), the number of tuples generated using SplitMap algorithm is 80 tuples, while the proposed DirectMap algorithm generates only 10 tuples. Thus, the average number of generated rule tuples in the RTR model using DirectMap is lower than the average number of generated rule tuples in the RTR model using SplitMap by 85.9%.

Figure 11 depicts the average authorization response time (in milliseconds) for RTR model using SplitMap role-mapping algorithm and the proposed DirectMap role-mapping algorithm. As shown in Figure 11, the average response time of RTR model using SplitMap algorithm is 130 milliseconds. While by using DirectMap algorithm, the average response time is 67 milliseconds. Therefore, the proposed DirectMap algorithm achieves better response time than SplitMap algorithm by 48.5%.

5.2.3. High-collaboration scenario

In the third scenario, a high degree of collaboration is considered using two collaborating organizations; host organization has 15 roles, guest organization has 20 roles, both organizations have 500 resources, and the mean number of intra-domain and inter-domain resources per role is varied between 1 and 500 inclusive. Figure 12 depicts the number of generated role-mapping tuples in the RTR model using the SplitMap role-mapping algorithm vs RTR model using the proposed DirectMap role-mapping algorithm for a range of mean values from 1 to 500 resources per role.

According to the results in Figure 12, it is found that the number of generated tuples using the proposed DirectMap algorithm is constant (i.e., for 20 roles in the guest organization, there are 20 mapping tuples in the online rule store). While, the number of generated tuples using SplitMap algorithm is increased by increasing the mean value (i.e., resources per role), until the maximum point which is 320 tuples for the

mean value of 151 resources per role. Therefore, by increasing the mean value, the number of tuples does not exceed the maximum value (i.e., 320). For the maximum mean value (i.e., 500), the number of tuples generated using the SplitMap algorithm is 320 tuples, while the proposed DirectMap algorithm generates only 20 tuples. Thus, the average number of generated rule tuples in the RTR model using DirectMap is lower than the average number of generated rule tuples in the RTR model using SplitMap by 93.1%. Figure 13 depicts the average authorization response time (in milliseconds) for RTR model using SplitMap role-mapping algorithm and the proposed DirectMap role-mapping algorithm.

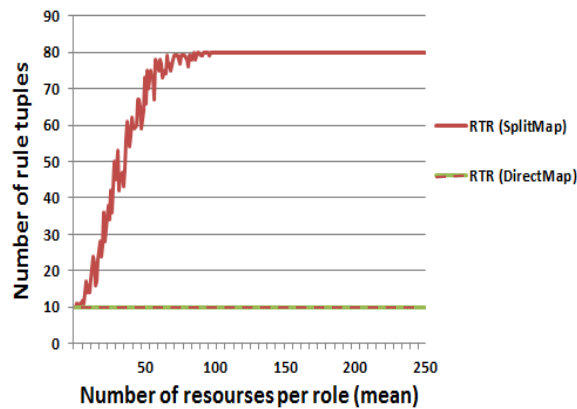


Figure 10. RTR using SplitMap vs RTR using DirectMap in middle collaboration scenario

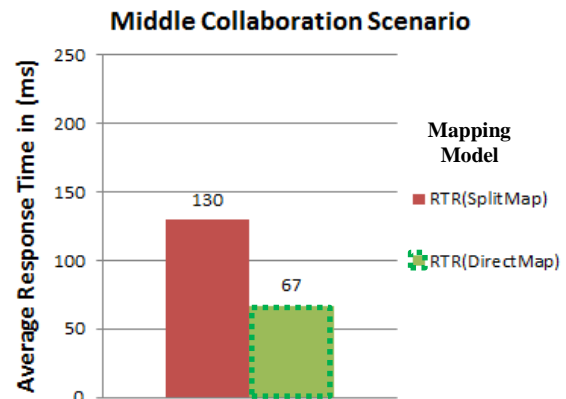


Figure 11. Average response time of RTR using SplitMap vs RTR using DirectMap in middle collaboration scenario

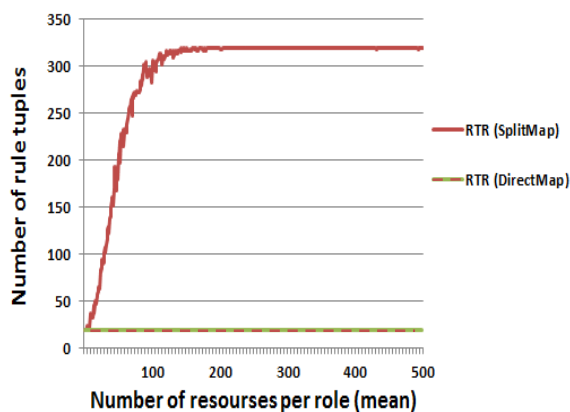


Figure 12. RTR using SplitMap vs RTR using DirectMap in high collaboration scenario

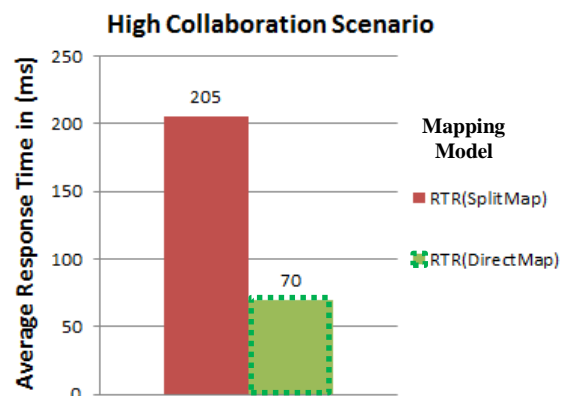


Figure 13. Average response time of RTR using SplitMap vs RTR using DirectMap in high collaboration scenario

As shown in Figure 13, the average response time of RTR model using SplitMap algorithm is 205 milliseconds. While using DirectMap algorithm, the average response time is 70 milliseconds. Therefore, the proposed DirectMap algorithm achieves better response time than SplitMap algorithm by 65.8%. Because the proposed DirectMap role-mapping algorithm achieves more saving in the rule-store size and the authorization response time relative to the SplitMap role-mapping algorithm in all the simulated collaboration scenarios, it will be more suitable to be used for role-mapping in RTR model as it generates minimal number of rule tuples and consequently minimal authorization response time.

5.3. Evaluation of RTR model using DirectMap vs RTO model

Another comparative study has been done to measure the effect of the collaboration degree on the size of the online Rule-Store and the authorization response time in the RTR model using the proposed DirectMap role-mapping algorithm compared to the RTO model. The online Rule-Store size in the RTO

model is defined as the number of tuples of each shared resource and permission assigned for every role among the collaborating organizations. While the online rule-store size in the RTR model is defined as the number of role-mapping tuples among the collaborating organizations. The average authorization response time is calculated by running 100 different authorization requests and measuring the authorization response time for each request. After that, the average response time of all the requests is calculated.

5.3.1. Low-collaboration scenario

Figure 14 depicts the number of rule tuples in the RTO model vs RTR model using the proposed DirectMap Role-Mapping algorithm for a range from one to twenty resources per role. As shown in Figure 14, the number of rule tuples in the RTO model increases with the increasing number of resources per role. On the other hand, the DirectMap role-mapping algorithm generates only 5 tuples with increasing the number of resources per role. The average numbers of rule tuples using RTO and RTR (using DirectMap) are 103 and 5 tuples respectively. Thus, the average number of rule tuples in the RTR model using DirectMap is lower than the average number of rule tuples in the RTO model by 95.1%. Figure 15 depicts the average authorization response time (in milliseconds) for RTO model and RTR model using the proposed DirectMap role-mapping algorithm.

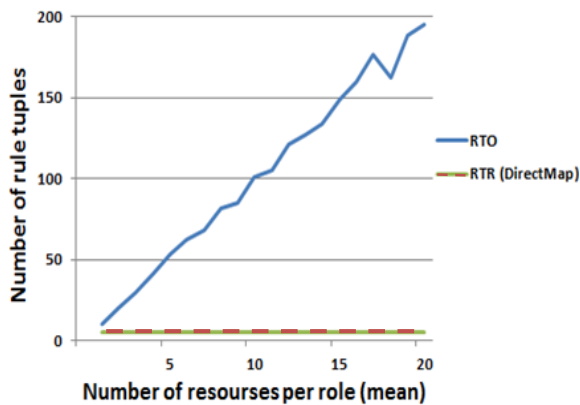


Figure 14. RTO vs RTR using DirectMap algorithm in a low collaboration scenario

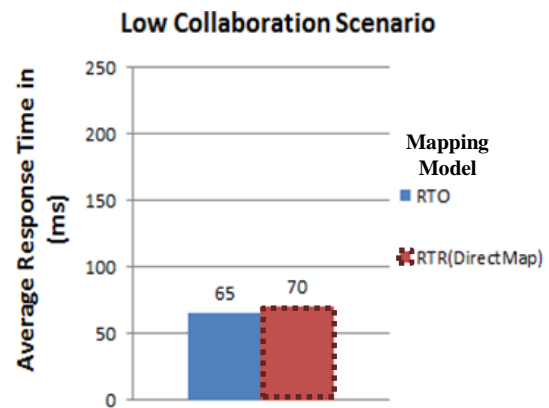


Figure 15. Average response time in RTO vs RTR using DirectMap in low collaboration scenario

As shown in Figure 15, the average response time in RTR model using the proposed DirectMap is 70 milliseconds while in RTO model is 65 milliseconds. So, the RTR mapping model using DirectMap is not better than RTO model in low collaboration scenario. This is because in the low collaboration scenario, the number of shared resources is low which causes low number of rule tuples in the RTO model. While the DirectMap role-mapping algorithm generates role-mapping tuples equal to the number of guest roles which could be more than the number of rule tuples in RTO model. Consequently, the searching time for matching in RTR model using DirectMap role-mapping algorithm will be larger than the RTO model. Therefore, the average authorization response time of the RTO model is better than the RTR model using DirectMap algorithm in low collaboration scenario.

5.3.2. Middle-collaboration scenario

Figure 16 depicts the number of rule tuples in RTO model vs RTR model using the proposed DirectMap role-mapping algorithm for a range from 1 to 250 resources per role. As shown in Figure 16, the number of rule tuples in RTO model increases with increasing the number of resources per role. On the other hand, the DirectMap role-mapping algorithm generates only 10 tuples with increasing the number of resources per role. The average numbers of rule tuples using RTO and RTR (using DirectMap) are 2109 and 10 tuples respectively. Thus, the average number of rule tuples in RTR model using DirectMap role-mapping algorithm is lower than the average number of rule tuples in RTO model by 99.5%. Figure 17 depicts the average authorization response time (in milliseconds) for RTO model and RTR model using the proposed DirectMap role-mapping algorithm. As shown in Figure 17, the average response time in RTO model is 81 milliseconds while in RTR model using DirectMap role-mapping algorithm is 67 milliseconds. Therefore, the RTR model using the proposed DirectMap role-mapping algorithm achieves better response time than the RTO model by 17.2%.

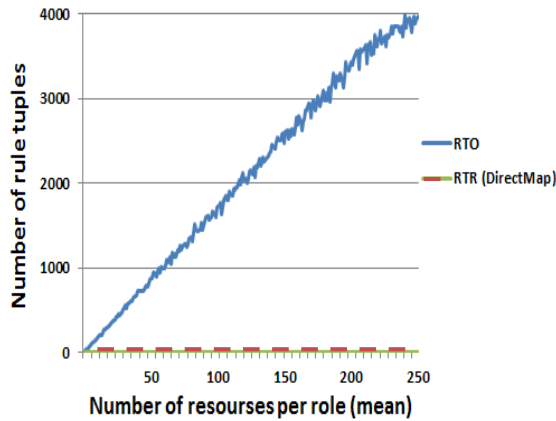


Figure 16. RTO vs RTR using DirectMap algorithm in middle collaboration scenario

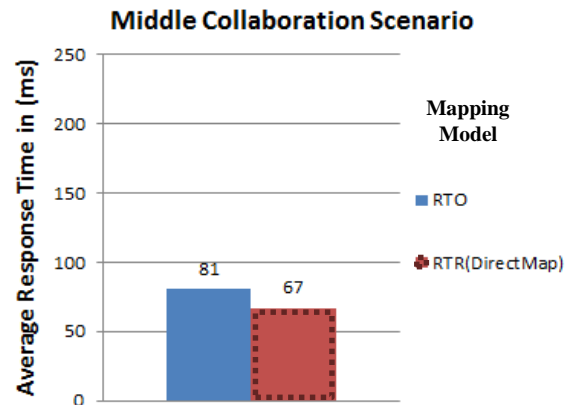


Figure 17. Average response time in RTO vs RTR using DirectMap in middle collaboration scenario

5.3.3. High-collaboration scenario

Figure 18 depicts the number of rule tuples in RTO model vs RTR model using the proposed DirectMap role-mapping algorithm for a range from 1 to 500 resources per role. As shown in Figure 18, the number of rule tuples in RTO model increases linearly with increasing number of resources per role. On the other hand, the DirectMap algorithm generates only 20 tuples with increasing the number of resources per role. The average numbers of rule tuples using RTO and RTR (using DirectMap) are 8674 and 20 tuples respectively. Thus, the average number of rule tuples in RTR model using DirectMap role-mapping algorithm is lower than the average number of rule tuples in RTO model by 99.7%. Figure 19 depicts the average authorization response time (in milliseconds) for RTO model and RTR model using the proposed DirectMap role-mapping algorithm.

As shown in Figure 19, the average response time in RTO model is 98 milliseconds. While in RTR model using DirectMap is 70 milliseconds. Therefore, the RTR model using the proposed DirectMap role-mapping algorithm achieves better response time than the RTO model by 28.5%.

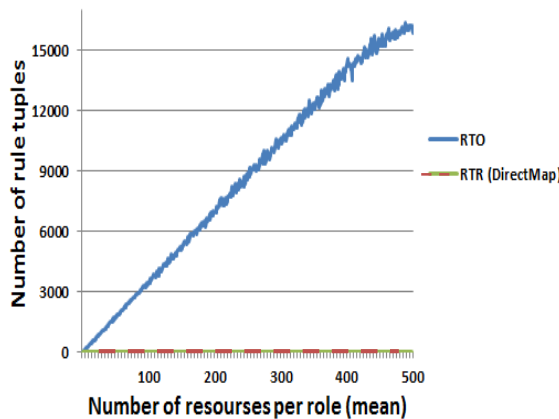


Figure 18. RTO vs RTR using DirectMap algorithm in high collaboration scenario

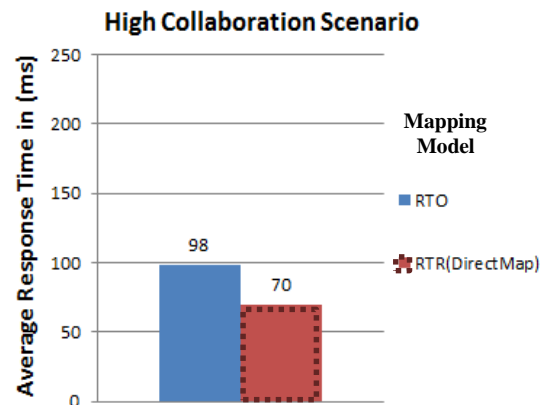


Figure 19. Average response time in RTO vs RTR using DirectMap in high collaboration

6. CONCURRENT IMPLEMENTATION OF RTR MODEL USING THE PROPOSED DIRECTMAP ALGORITHM

In order to enhance the authorization request’s response time of the RTR model in all collaboration scenarios, the RTR mapping model using the proposed DirectMap role-mapping algorithm has been implemented concurrently [23], [24]. This type of concurrency called single program multiple data (SPMD) computational model [25]–[27]. Therefore, the searching time decreases and consequently the authorization request’s response time decreases.

The flowchart of the concurrent implementation of the RTR searching technique using the DirectMap role-mapping algorithm is presented in Figure 20. By implementing the RTR mapping model using DirectMap role-mapping algorithm concurrently, the following request parameters are assumed; a requester *UserX* from organization *Org1* requests to access a resource *r* with permission *p* in the target organization *Org2*. The role-mapping rules are generated by the DirectMap role-mapping algorithm (i.e., each role in *Org1* is mapped to only one new role in *Org2*).

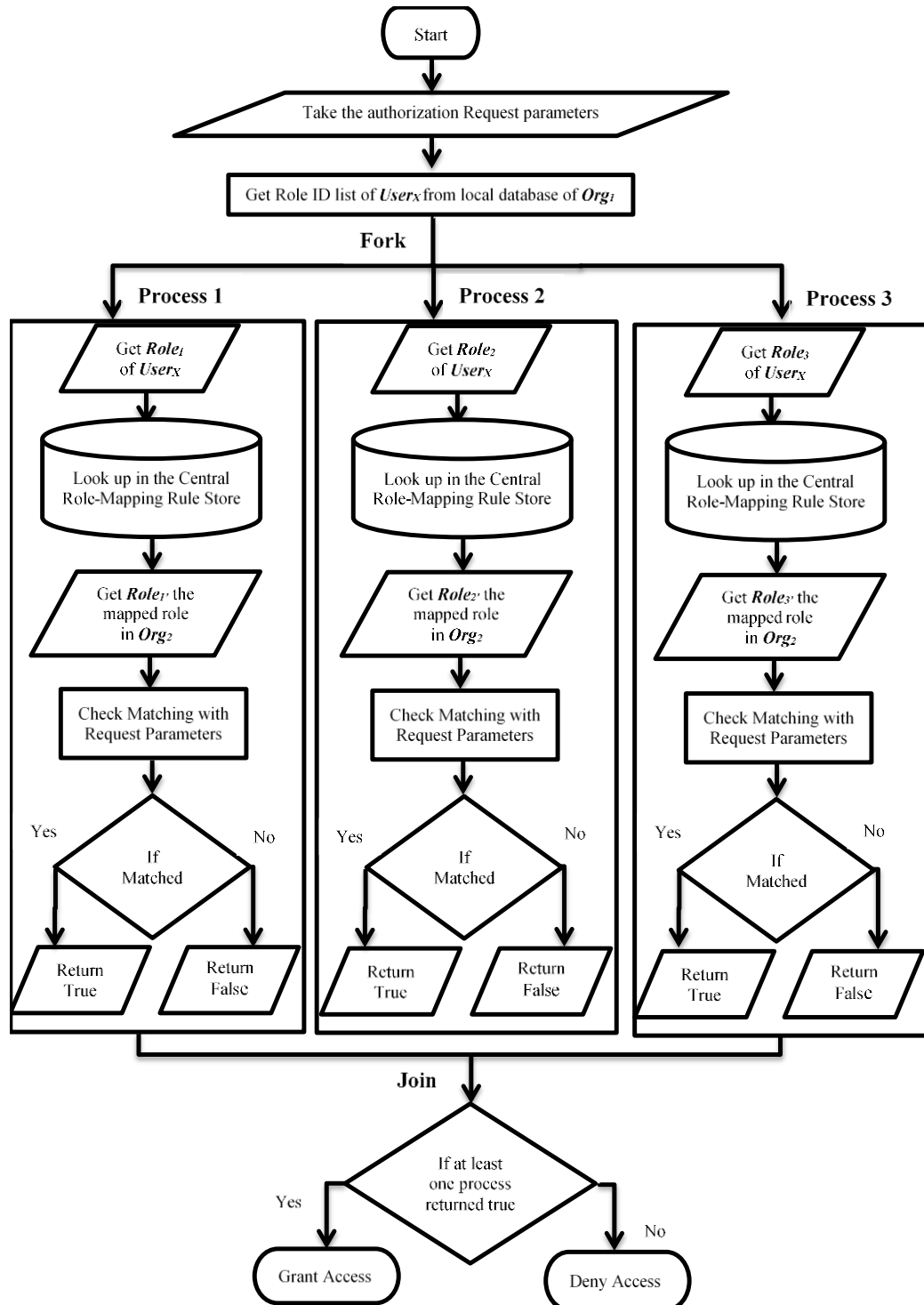


Figure 20. Flow chart of applying SPMD concurrent model on RTR searching technique with DirectMap role-mapping algorithm

Assume that *UserX* has three roles (i.e., *Role1*, *Role2*, and *Role3*) in his organization *Org1*. By applying SPMD computational model, multiple processes execute the same function concurrently, as shown in Figure 20. Each process takes one user's Role, gets its mapped one in the target organization from the RTR rule store, and then checks for matching with the requested parameters. After the processes are finished, if there is a hit in one or more of the processes, the system will grant *UserX* to access the requested resource with the specified permission. If there are no hits, the system will deny *UserX* from accessing the requested resource.

6.1. The evaluation of the RTR mapping model by applying SPMD concurrent model

To evaluate RTR model using DirectMap algorithm with applying SPMD concurrent model, its authorization request's response time is measured and compared with the RTO model in low and high collaboration scenarios.

6.1.1. Low-collaboration scenario

Figure 21 depicts the average authorization response time (in milliseconds) for the RTR using DirectMap algorithm with applying SPMD concurrent model against the RTO model. As shown in Figure 21, the average authorization response time of RTO model is 65 milliseconds while in RTR model using the proposed DirectMap with concurrent implementation is 59 milliseconds. So, RTR model using DirectMap algorithm with concurrent implementation is better than the RTO model by 9.2%.

6.1.2. High-collaboration scenario

Figure 22 depicts the average authorization response time (in milliseconds) for the RTR using DirectMap algorithm with applying SPMD concurrency model against the RTO model. As shown in Figure 22, the average authorization response time in RTO model is 98 milliseconds while in RTR model using the proposed DirectMap with concurrently implementation is 61 milliseconds. So, the RTR using the proposed DirectMap algorithm better than the RTO model by 37.7%. Therefore, by applying SPMD concurrent model of the RTR model using the proposed DirectMap algorithm achieves more saving in the authorization response time in all collaboration scenarios.

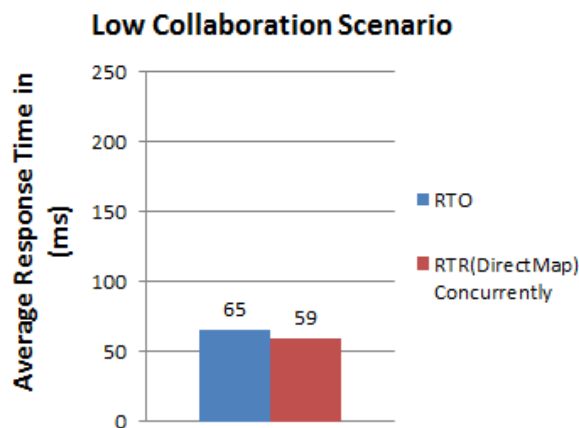


Figure 21. Average response time of RTO vs RTR (DirectMap) concurrently in low collaboration scenario

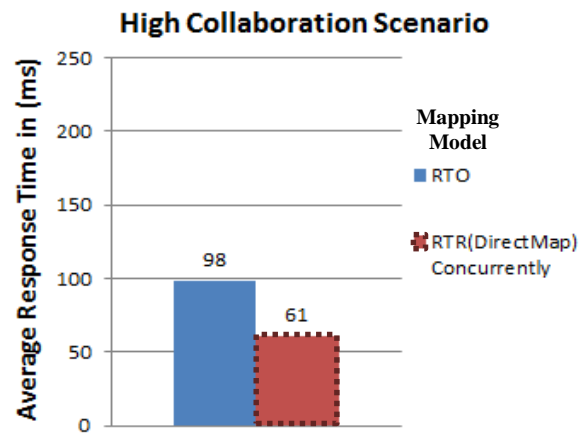


Figure 22. Average response time of RTO vs RTR (DirectMap) concurrently in high collaboration scenario

6.2. Summary

The goal of the paper is to introduce a role-mapping algorithm by using the role-to-role mapping technique for the RBAC model instead of role-to-object mapping technique to save the online rule store size and also improve the authorization request's response time in different collaboration cloud environments (i.e., low, middle, and high). In our previous work, the RTR technique has been implemented using the SplitMap role-mapping algorithm. The main limitation of SplitMap is the cost of the authorization request's response time. According to this work, a new role-mapping algorithm, called DirectMap, has been introduced as an enhancement of SplitMap algorithm. To evaluate the performance of the proposed DirectMap algorithm, a comparative study has been done between the RTR mapping technique using the proposed

DirectMap algorithm and the SplitMap algorithm concerning the rule-store size and the authorization response time in three different collaboration scenarios (i.e., low, middle, and high). The summary of the comparative results of DirectMap algorithm against SplitMap algorithm is presented in Table 3.

Table 3. Comparative results of DirectMap algorithm vs SplitMap algorithm

	Collaboration Scenario	RTR using SplitMap	RTR using DirectMap	DirectMap Improvement Percentage Relative to SplitMap
Average Rule Store Size (in number of rule tuples)	Low	18	5	72.2%
	Middle	71	10	85.9%
	High	291	20	93.1%
Average Authorization Response Time (in ms)	Low	103	70	32%
	Middle	130	67	48.5%
	High	205	70	65.8%

According to the results in Table 3, it is found that DirectMap algorithm generates minimal number of rule tuples than SplitMap algorithm which decreases the rule-store size. Also, DirectMap algorithm reduces the authorization response time relative to SplitMap algorithm in all collaboration scenarios. Another comparative study was performed to evaluate the proposed DirectMap algorithm against the traditional RBAC based on role-to-object mapping technique concerning the rule-store size and the authorization response time in the different collaboration scenarios. The summary of the comparative results of the RTR model using DirectMap algorithm against the RTO model is presented in Table 4.

Table 4. Comparative results of RTR using DirectMap vs RTO model

	Collaboration Scenario	RTO	RTR using DirectMap	DirectMap Improvement Percentage Relative to RTO
Average Rule Store Size (in number of rule tuples)	Low	103	5	95.1%
	Middle	2109	10	99.5%
	High	8674	20	99.7%
Average Authorization Response Time (in ms)	Low	65	70	-
	Middle	81	67	17.2%
	High	98	70	28.5%

According to the results in Table 4, it is found that the proposed DirectMap algorithm in the RTR model achieves quadratic saving in the online rule store size which outperforms the RTO model. In spite of reducing the response time by the RTR using DirectMap algorithm in middle and high collaboration, it has a limitation in low collaboration scenario. So, it has been enhanced by implementing the RTR model using the proposed role-mapping algorithm (i.e., DirectMap) concurrently to achieve more saving in the authorization request's response time.

The last comparative study has been done to evaluate the effect of applying SPMD concurrency model on the RTR using DirectMap algorithm against the RTO model concerning the authorization response time in the different collaboration scenarios. The summary of the comparative results of the RTR model using DirectMap algorithm with concurrently implementation against the RTO model is presented in Table 5. According to the results in Table 5, it is found that implementing RTR model with DirectMap algorithm using SPMD concurrency model achieves more saving in the authorization response time in all collaboration scenarios. Therefore according to the experimental results, The DirectMap algorithm will be more suitable to be used for role-mapping in the RTR model as it generates the minimal number of rule tuples, as well as, minimal authorization response time. In addition, applying concurrency in the searching method of the RTR model contributed in achieving more saving in the authorization response time in different collaboration cloud environments.

Table 5. Comparative results of RTR using DirectMap concurrently vs RTO model

	Collaboration Scenario	RTO	RTR using DirectMap Concurrently	DirectMap Concurrently Improvement Percentage Relative to RTO
Average Authorization Response Time (in ms)	Low	65	59	9.2%
	Middle	81	57	29.6%
	High	98	61	37.7%

7. CONCLUSION

Federated authorization systems have been gaining more ground recently due to the increasing nature of collaboration and resource sharing across administrative domains in today's applications. A cornerstone, besides security, for the successful adoption of such systems is performance. The work in this paper focuses on improving the authorization performance in high collaboration cloud environments by reducing the online rule store size and the authorization request's response time. The work in this paper proves 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.

According to the experimental results, it is found that using the role-to-role mapping model instead of the role-to-object mapping model achieves more stable online storage footprint. The drawback of using RTR mapping model is that the role-mapping algorithm produces time overhead in the role-mapping process. However, role-mapping is done as a background process, so it has minimal effect in running services.

It is also found that the proposed DirectMap role-mapping algorithm achieves more saving in the online rule store size and also more reduction in the authorization response time relative to the SplitMap algorithm. The main drawback of the DirectMap algorithm is the overhead on the intra-domain database of the host organization as a result of the creation of new roles and their implications subsequently. The overhead that affects the intra-domain database is calculated by the summation of requester resources for each role. By implementing the proposed role-mapping algorithm (i.e., DirectMap) concurrently in the RTR model, more saving in the authorization response time is achieved. Therefore, it is better to be used in high collaboration cloud environments.




REFERENCES

- [1] L. Wang *et al.*, "Cloud computing: a perspective study," *New Generation Computing*, vol. 28, no. 2, pp. 137–146, Apr. 2010, doi: 10.1007/s00354-008-0081-5.
- [2] Office of Management and Budget, "Federal cloud computing strategy," *Office of Management and Budget*, 2019. <https://cloud.cio.gov/> (accessed May 01, 2019).
- [3] R. Zhang and L. Liu, "Security models and requirements for healthcare application clouds," in *2010 IEEE 3rd International Conference on Cloud Computing*, Jul. 2010, pp. 268–275, doi: 10.1109/CLOUD.2010.62.
- [4] G. Gopalan, A. Negi, and V. N. Sastry, "A cross - domain role mapping and authorization framework for RBAC in grid systems," *International Journal of Computer Science and Applications*, vol. 6, no. 1, pp. 1–12, 2009.
- [5] U. Kaur and D. Singh, "Comparative analysis of access control models," *International Journal of Computer Trends and Technology*, vol. 29, no. 3, pp. 132–135, Nov. 2015, doi: 10.14445/22312803/IJCTT-V29P123.
- [6] T. Mudarri and S. A. Al-Rabeei, "Security fundamentals: access control models," *International Journal of Interdisciplinarity in Theory and Practice*, vol. 7, pp. 259–262, 2015.
- [7] K. Hashizume, D. G. Rosado, E. Fernández-Medina, and E. B. Fernandez, "An analysis of security issues for cloud computing," *Journal of Internet Services and Applications*, vol. 4, no. 1, p. 5, 2013, doi: 10.1186/1869-0238-4-5.
- [8] J. Howell and D. Kotz, "End-to-end authorization," *Proceedings of the 4th conference on Symposium on Operating System Design & Implementation*. 2000.
- [9] H. Takabi, J. B. D. Joshi, and G.-J. Ahn, "Security and privacy challenges in cloud computing environments," *IEEE Security & Privacy Magazine*, vol. 8, no. 6, pp. 24–31, Nov. 2010, doi: 10.1109/MSP.2010.186.
- [10] S. Gerges, S. Khattab, H. Hassan, and F. A. Omara, "Scalable multi-tenant authorization in highly-collaborative cloud applications," *International Journal of Cloud Computing and Services Science (IJ-CLOSER)*, vol. 2, no. 2, Jan. 2013, doi: 10.11591/closer.v2i2.1665.
- [11] P. Rao, D. Lin, E. Bertino, N. Li, and J. Lobo, "Fine-grained integration of access control policies," *Computers & Security*, vol. 30, no. 2–3, pp. 91–107, Mar. 2011, doi: 10.1016/j.cose.2010.10.006.
- [12] S. Alansari, F. Paci, and V. Sassone, "A distributed access control system for cloud federations," in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, Jun. 2017, pp. 2131–2136, doi: 10.1109/ICDCS.2017.241.
- [13] Y. A. Younis, K. Kifayat, and M. Merabti, "An access control model for cloud computing," *Journal of Information Security and Applications*, vol. 19, no. 1, pp. 45–60, Feb. 2014, doi: 10.1016/j.jisa.2014.04.003.
- [14] B. Tang, Q. Li, and R. Sandhu, "A multi-tenant RBAC model for collaborative cloud services," in *2013 Eleventh Annual Conference on Privacy, Security and Trust*, Jul. 2013, pp. 229–238, doi: 10.1109/PST.2013.6596058.
- [15] 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*, 2012, pp. 88–93.
- [16] InCommon, "Shibboleth," *InCommon*. <http://shibboleth.internet2.edu/> (accessed Dec. 09, 2019).
- [17] J. M. A. Calero, N. Edwards, J. Kirschnick, L. Wilcock, and M. Wray, "Toward a multi-tenancy authorization system for cloud services," *IEEE Security & Privacy Magazine*, vol. 8, no. 6, pp. 48–55, Nov. 2010, doi: 10.1109/MSP.2010.194.
- [18] 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 - SACMAT '07*, 2007, doi: 10.1145/1266840.1266866.
- [19] D. F. Ferraiolo and D. R. Kuhn, "Role-based access controls," in *15th National Computer Security Conference*, Mar. 2009, pp. 554–563.
- [20] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman, "Role-based access control models," *Computer*, vol. 29, no. 2, pp. 38–47, 1996, doi: 10.1109/2.485845.
- [21] R. Sandhu, D. Ferraiolo, and R. Kuhn, "The NIST model for role-based access control," in *Proceedings of the fifth ACM workshop on Role-based access control - RBAC '00*, 2000, pp. 47–63, doi: 10.1145/344287.344301.
- [22] J. Hu, R. Li, and Z. Lu, "On role mappings for RBAC-based secure interoperation," in *2009 Third International Conference on Network and System Security*, 2009, pp. 270–277, doi: 10.1109/NSS.2009.76.
- [23] Joseph JáJá, *An introduction to parallel algorithms*. United States: Addison Wesley Longman Publishing Co., Inc., 1992.




- [24] M. Tahir, "Programming concurrency in C++ - part one," *C#Corner*, 2018. <https://www.c-sharpcorner.com/article/programming-concurrency-in-cpp-part-1> (accessed May 01, 2021).
- [25] F. Darema, D. A. George, V. A. Norton, and G. F. Pfister, "A single-program-multiple-data computational model for EPEX/FORTRAN," *Parallel Computing*, vol. 7, no. 1, pp. 11–24, Apr. 1988, doi: 10.1016/0167-8191(88)90094-4.
- [26] S. K. Meesala, D. P. M. Khilar, and D. A. K. Shrivastava, "The parallel architecture approach, single program multiple data (SPMD) implementation on clusters of terminals using Java Rmi," *International Journal of Computational Engineering Research (IJCER)*, vol. 6, no. 3, pp. 16–23, 2016.
- [27] D. Bradley, "SPMD: single program multiple data streams," *Tufts University*. 2016, Accessed: Jul. 30, 2021. [Online]. Available: <https://slideplayer.com/slide/7559656>.

BIOGRAPHIES OF AUTHORS






Doaa Abdelfattah    is a Master Student in the Department of Computer Science, Faculty of Computers and Artificial Intelligence, Cairo University since 2016 in fields of Cloud Computing and Security. She obtained bachelor's degree in Computer Science from faculty of Computers and Artificial Intelligence, Ain Shams University in 2015. She also works as a software quality engineer since 2015 in IT field. She can be contacted at email: doaaabdelfattah94@gmail.com.



Hesham A. Hassan    is an Egyptian researcher born in Cairo in 1953. Hesham's educational background is as follows; BSc in Agriculture, Cairo University, Egypt in 1975. Postgraduate diploma in computer science, from ISSR, Cairo University, Egypt in 1984. MSc in 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 a PROFESSOR and HEAD of Computer Science department at the faculty of Computers and Artificial Intelligence, 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. Prof. Hesham has supervised over 27 PhD and MSc 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). He can be contacted at email: drhesham2007@gmail.com.



Fatma A. Omara    is a Professor in the Computer Science Department and Vice Dean for Community Affairs and Development in the Faculty of Computers and Artificial Intelligence, 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 MSc 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. She can be contacted at email: fatma_omara@hotmail.com.