A novel role-mapping algorithm for enhancing highly collaborative access control system

Doaa Abdelfattah1 · Hesham A. Hassan1 · Fatma A. Omara1

Accepted: 10 March 2022 / Published online: 31 March 2022 © The Author(s) 2022

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. This paper addresses the scalability problem of the online stored rules. This problem affects the performance of the access control system due to increasing number of shared resources and/or 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 is performed to evaluate the proposed algorithm’s performance with concerning the Rule-Store size and the authorization response time. According to the results, it is found that the proposed algorithm reduces the number of stored rules which minimizes the Rule-Store size and reduces the authorization response time. Additionally, this paper proposes applying a concurrent approach on the RTR mapping model using the proposed Role-Mapping algorithm to achieve more savings in the authorization response time. Therefore, it will be suitable in highly-collaborative cloud environments.

Keywords Cloud computing · Authorization · Role-mapping · Role-based access control

---

Doaa Abdelfattah
doaaabdelfattah94@gmail.com

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

Cloud computing, which was modeled in 2007, currently became a hot topic due to its abilities to offer flexible dynamic IT infrastructures and deliver many resources like computing and storage to users on demand regardless of any maintenance of extra resources [1]. User can store large amount of data on cloud and access it from anywhere on any device. Cloud storage service is cheap and provides services at long term access with low cost and scalable high performance storage architecture. Because of the dramatic and continuous increases in the amount of digital information that needs to be stored, there is an obvious motivation for the service providers to explore outsourcing of users’ data to the cloud. Many cloud applications involve multiple collaborating organizations to share data, services, and resources. For example, the federal US 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]. Many large companies like Google, IBM and HP have multiple business units which are hosted in private cloud and sharing secured data among each other. 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 specify the permissions and roles for allowing access to the shared resource. These rules are called cross-domain rules, which are centrally stored in a third-party access control service. A trusted authorization provider saves all the local rules of each organization and the cross-domain statements in a central rule store [4].

The authorization function is responsible of 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. Whereas, 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]. Cloud providers and collaborating organizations face many new security challenges, 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 exponentially 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, this paper addresses the scalability problem of the online Rule-Store which contains the set of rules that are checked with every authorization request in run-time. Also, this paper discusses the effect of that problem on the authorization performance such as the authorization request’s response time.

There are many schemas were introduced for access control over cloud computing environments [10–20]. In [12], Calero introduced RBAC multi-tenant authorization system using Role-to-Object mapping technique for access control management. It suffers from a quadratic number of rules in its online rule store in highly collaborative applications. Also, Tang [11] solved the scalability problem by increasing infrastructure parameters like RAM and CPU cores. However, the multi-tenant RBAC policies with different expressive power incur various policy evaluation delays. On the other hand, Gerges et al. [10] introduced a solution by replacing the Role-to-Object mapping model with Role-to-Role (RTR) mapping model using a Role-Mapping algorithm called, SplitMap. This Role-Mapping algorithm has a limitation in terms of the cost of authorization request’s response time. Therefore, it motivated us to enhance the previously introduced SplitMap Role-Mapping algorithm in [10] by adding the following contributions:

- Introducing a new Role-Mapping algorithm 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.
- Performing comparative studies between the proposed algorithm against the traditional RBAC model and against the previously introduced SplitMap Role-Mapping algorithm.
- Applying SPMD concurrent approach on the RTR model using the proposed Role-Mapping algorithm to achieve more reduction in the authorization request’s response time in all degrees of collaboration.

The remaining sections of the paper are organized as follows; Sect. 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 introduces 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 against other existing algorithms. Section 6 introduces applying SPMD concurrent approach on the RTR mapping 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 and discusses the future work.
2 Related work

In this section, the related work of Cloud Computing Access Control models, Multi-Tenancy RBAC models, and Role-Mapping approaches are discussed.

Rao et al. [21] 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 didn’t consider the impact of the presence of obligations when integrating policies using Fine-grained Integration Algebra (FIA). Also, they didn’t clarify how to guarantee that a given algebraic expression will behave as expected.

Alansari et al. [22] 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 didn’t produce an extensive evaluation of the system concerning the performance and the cost of execution.

Younis et al. [23] 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. [11] 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 AaaS (Authorization as a service) incur various policy evaluation delays.
Leandro et al. [24] proposed a Multi-Tenancy authorization system using Shibboleth [25]. 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 Single Sign-On 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. [12] 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 [26] 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 mapping with RTR mapping using a proposed Role-Mapping algorithm called, SplitMap. This algorithm has a limitation in terms of the cost of authorization request’s response time. According to the work in this paper, this system is enhanced by introducing a new Role-Mapping algorithm called, DirectMap. The proposed algorithm aims to save the authorization response time and achieve more reduction on the online rule store size in highly-collaborative environment. Table 1 summarizes the major findings from the existing schemes.
| Scheme | Key finding                                                                 | Advantage                                                                 | Disadvantage                                                                 |
|--------|-----------------------------------------------------------------------------|---------------------------------------------------------------------------|------------------------------------------------------------------------------|
| [21]   | Introducing algebra for fine-grained integration of language-independent     | This algebra can support the specification of a large variety of integration | Ignoring the impact of the presence of obligations when integrating policies   |
|        | policies which integrates access control policies of collaborating parties    | constraints specified in XACML                                             | using Fine-grained Integration Algebra (FIA)                                   |
|        | Proposing a framework that uses algebra for the fine-grained integration of  | Granting users to access federated data when their identity attributes match | Also, there is no clarification of how to guarantee that a given algebraic     |
|        | policies                                                                 | the policies without revealing their attributes to the federated organization which owns the data | expression will behave as expected                                             |
|        | This algebra can support the specifica-                                        | Securing access and flow of data by marking the data with security labels,  | There is no extensive evaluation of the system concerning the performance and |
|        | tion of a large variety of integration constraints specified in XACML          | which state the data sensitivity                                           | the cost of execution                                                        |
| [22]   | Introducing a novel identity and access management system for cloud          |                                                                          |                                                                              |
|        | access management system for cloud federations                               |                                                                          |                                                                              |
|        | Applying attribute-based access control policies on the federated organizations’ data in a privacy-preserving manner |                                                                          |                                                                              |
| [23]   | Introducing a novel access control model for cloud computing. It utilizes     |                                                                          |                                                                              |
|        | temporal (time and location) and delegation constraints                        |                                                                          |                                                                              |
| [11]   | Introducing a Multi-Tenant Role-Based Access Control (RBAC) Model for        |                                                                          |                                                                              |
|        | collaborative cloud environments                                              |                                                                          |                                                                              |
|        | The model architecture was built on a centralized Policy Decision Point (PDP) | S solving the scalability problem among collaborating organizations by      | The multi-tenant RBAC policies with different expressive power in the AaaS      |
|        | and multiple Policy Enforcement Points (PEPs)                                | increasing the number of tenants (i.e., increasing the number of PEPs), or by increasing PDP RAM and CPU cores | (Authorization as a service) incur various policy evaluation delays            |
| [12]   | introducing a Multi-Tenancy authorization model suitable for cloud computing  | Each organization of the collaborating organizations represent all roles in statements 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 number of rules in its online rule store |
| Scheme                                                                 | Key finding                                                                 | Advantage                                                                 | Disadvantage                                                                 |
|-----------------------------------------------------------------------|-----------------------------------------------------------------------------|---------------------------------------------------------------------------|--------------------------------------------------------------------------------|
| [4] Introducing a scalable Role-Mapping approach based on ranking roles and resources with specific scores | Deciding request granting or denial is based on comparing the role and resource scores | 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 |
| [26] Introducing an inter-domain Role-Mapping technique based on the principle of least privilege | Setting minimal cardinality for a role across a domain to avoid misuse of access | Trying to map already existing roles without creating new roles | This approach might cause losing some of the desired privileges |
| [10] Introducing a scalable Multi-Tenant Authorization system          | Replacing the Role-to-Object mapping model with Role-to-Role mapping model using a proposed Role-Mapping algorithm | Minimizing the number of rule tuples in the online Rule-Store | This algorithm has a limitation in terms of the cost of authorization request’s response time |
3 Preliminary studies

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 [27, 28]. According to the work in this paper, RTR mapping 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 Role-to-Object mapping technique, which means that users can access resources or objects based on the roles they have been assigned [29]. This can be achieved by mapping the permissions for certain resources (objects) to certain roles. Then, the roles are assigned to users and hence permitted to access the resources (see Fig. 1).

As shown in Fig. 1, RBAC has three main entities:

- **User**: An individual that has access to the organizational resources. Each individual has an associated user ID.
- **Role**: Job function within the organization. Typically, a description of the authority and responsibility conferred on this role is associated with each role.
- **Permission**: An approval of a particular mode of access to an object. Equivalent terms are privilege and access right.

The relationship between User and Role is many-to-many relationship; one user can have multiple roles and multiple users may be assigned to a single role. Similarly, there is a many-to-many relationship between Role and Permission; one role

![Fig. 1 Principles of role-based access control model](image)
can be assigned with multiple permissions on multiple resources while, one permission may be assigned to multiple roles.

A rule is created for each role, resource and permission to specify the access policy. The authorization process checks if there is an existing rule that fulfils the specified request to decide granting or denial of the request. In other words, 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 or inter-domain rules. While the local RBAC rules of an organization are called intra-domain rules. A trusted authorization service provider stores the intra-domain rules 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.

In [12], the introduced RBAC multi-tenant authorization system has a centralized RBAC rule store. 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. It 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. It is represented as $(issuer, org_1, role, org_2, object, permission, service)$ which means that the issuer has granted the role of organization org_1 a specified permission on the object in organization org_2, 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. In order to solve this problem, the idea of Role-Mapping (i.e., RTR mapping) technique came out. It maps roles in each organization to the roles in the host organization, which owns the shared resource [10]. The 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 \(m\) (See Fig. 2a). 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 Fig. 2a, 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 Role-to-Object model, role \(i\) in organization \(n\) maps to the privileges on certain shared resources in organization \(m\). Therefore, user \(Y\) has the privileges to access the mapped resources with the assigned permissions (See Fig. 2b).

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. 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.

We can define the mapping logic for each model as follows; The Role-to-Object mapping logic is done based on the organization’s requirements. In other words, the organization specifies the access rights of each role inside it. Therefore, each employee can be assigned his dedicated role. According to the RTR mapping, it is done based on the used Role-Mapping algorithm. The mapping of the used algorithm depends on the *intra-domain* rules and the *cross-domain* rules of each organization.

The RTR mapping tuples should be generated by a Role-Mapping algorithm. There are many algorithms were proposed for Role-Mapping [10, 30–35]. According to the work in this paper, a new 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 previously introduced 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]. The principles of
the algorithm are illustrated in Fig. 3. 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 \(j\) (See Fig. 3a). 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 (see Fig. 3b). If \(i\) is a role that has resources’ privileges more than the requested privileges from role \(j\), 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 (see Fig. 3c). If all the host roles are checked and role \(j\) isn’t 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. Then, it is mapped to role \(j\). Therefore, SplitMap algorithm performs the Role-Mapping through these three cases. The algorithm iterates over each ordered pair of all the collaborating organizations (i.e., host and guest). Each role in the guest organization is mapped to a role in the host organization based on its case. So, it checks (from the three cases) the case of each guest role, and makes its mapping to a host role based on its case.

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, the authorization request’s response time is increased relative

![Fig. 3 Principles of the SplitMap algorithm](image)
to the RTO model. In addition, this algorithm wouldn’t be effective by considering the following scenario:

Assume that each requested role consumes $N$ 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.

### 4 The proposed DirectMap role-mapping algorithm

In order to enhance the performance of the SplitMap algorithm, a new 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.

The basic sets which are used in the proposed DirectMap algorithm are listed in Table 2.

The algorithm’s formal definition is described as follows;

- \( \forall j \in J \rightarrow \exists \text{Req}(j) \subseteq O \times P \)
  
  For each guest role \( j \) in the guest organization, take its access rights which granted to the guest organization by the host organization.

| 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 \( A \subseteq I \times O \times P \) |
| \( J \) | Set of roles in the guest organization |
| \( \text{Req} \) | Set of access rights granted to the guest organization by the host organization \( \text{Req} \subseteq J \times O \times P \) |
| \( \hat{I} \) | 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 \( \hat{A} \subseteq \hat{I} \times O \times P \) |
| \( M \) | Set of role mapping rules between the guest organization and the host organization \( M \subseteq J \times (I \cup \hat{I}) \) |
\[ \forall_{RP \subset \text{Req}(j)} \rightarrow \text{Temp} = \text{Temp} \cup \text{RP}, \text{RP}: \langle r, p \rangle \]
For each requested pair of (resource, permission) \( \text{RP} \) 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 \( i' \): New Role
Create a new role \( i' \) in the host organization and add it to \( \hat{I} \) set.

- \( \forall_{\text{RP} \subset \text{Temp}} \rightarrow \hat{A} = \hat{A} \cup \langle i', \text{RP} \rangle \]
For each access right pair of (resource, permission) \( \text{RP} \) in Temp set, assign it to the new role \( i' \) and add them to \( \hat{A} \) set.

- \( \hat{M} = \hat{M} \cup \langle j, i' \rangle \)
Map the new host role \( i' \) to the guest role \( j \) and add this mapping rule to the set \( \hat{M} \).

Then, after iterating over all the guest roles, insert all the updated sets (\( \hat{I}, \hat{A}, \) and \( \hat{M} \)) to the central Rule-Store.

Temp is a vector of pairs. It is initially empty. It is used to temporarily store the pairs of (resource, permission) of the requested guest role. Then, the algorithm takes its stored pairs and assigns them to the newly created role in the host organization. Then it maps this new host role to the requested guest role. This process is repeated with all the guest roles. The pseudo-code of the proposed DirectMap Role-Mapping algorithm is listed in Fig. 4.

We can discuss the proposed DirectMap algorithm with the following steps:

1. Iterate over each guest role \( j \) in the set of guest roles \( J \).
2. Initiate the Temp set as an empty set.
3. Get the requested access rights of the guest role \( j \) and store them in \( \text{Req}' \).
4. Iterate over each ordered pair of (resource, permission) in \( \text{Req}' \) and add it to Temp.
5. Create a new host role \( i' \) and add it to the set of new roles \( \hat{I} \).
6. Get the (resource, permission) pairs from Temp and add them to the new role \( i' \) then store them in \( \hat{A} \).
7. Map the guest role \( j \) to the new host role \( i' \).
8. After finishing the guest roles set \( J \), insert the set of \( \hat{I} \) to the local roles of the host organization, add the set of \( \hat{A} \) to local roles’ access rights of the host organization, and finally add the Role-Mapping rules of \( \hat{M} \) to the inter-domain database.

The algorithm’s principles are illustrated in Fig. 5. 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 each ordered pair of all the collaborating organizations. As shown in Fig. 5, assume that \( \text{Role}_j \) is a role in the guest organization \( \text{Org}_x \) and requests to access some shared resources with certain permissions (i.e., \( P_1 \) and \( P_2 \)) on the host organization \( \text{Org}_z \). Regardless of which roles these resources’ permissions belong to, the DirectMap algorithm directly creates a new role \( \text{Role}_{jx} \) in the host organization \( \text{Org}_z \) and assigns all requested resources’ permissions to that role inside the host organization. Then, it maps \( \text{Role}_{jx} \) to \( \text{Role}_j \). 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. Then, it 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.

In the collaborative environment, there are multiple organizations are registered and share their resources among each other in the same cloud service. The proposed algorithm iterates over each ordered pairs of the collaborating organizations. For example, assume we have two collaborating organizations (Org1, and Org2). The algorithm takes Org1 as a host organization and Org2 as a guest organization to perform the mapping from the guest roles of Org1 to the host roles of Org2. Then, it takes Org2 as a host and Org1 as a guest to perform the mapping from the guest roles of Org2 to the host roles of Org1. Similarly, this process is performed for each ordered pairs of the collaborating organizations if there are more than two organizations.

The relationship between roles and users inside the organization is the responsibility of the organization itself. The organization assigns its roles to its users as per its internal policy. The same role can be assigned to different users. We handled these relationships and simulated the real life data in order to perform our experiments which evaluate the proposed algorithm. This simulation will be discussed in the following section.

5 Performance evaluation of RTR model using the proposed DirectMap algorithm

In this section, the performance evaluation of the proposed DirectMap algorithm is presented, starting by describing the experimental environment. Then, the experiments are 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 Experimental Environment

In order to implement the RTO model and the RTR mapping model using Role-Mapping algorithms, a full-structured access control system is implemented. The used open-source tools for implementation are as follows:

- Tomcat web container is used as the deployment server where the system is deployed on.
- Java/Servllet is the controller of handling the authorization request life cycle.
- SQL is the query language that used to communicate with DBMS.
- MySQL server is used as the database server that contains the created inter-domain and intra-domain databases of communicating organizations.

a. Database Design
The implemented database has two schemas; inter-domain and intra-domain. The intra-domain database is a central database schema that contains the local RBAC rules of all the collaborating organizations. It is represented using six tables:

- **Organizations** table contains all the collaborating organizations that sharing resources in the collaborative environment.
- **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.
- **Role-Resource-Permission** table contains relations over the Roles, Resources and Permissions tables.

The inter-domain database is a central database schema that contains relations among the collaborating organizations. It is represented using three tables:

- **Organizations** table contains all the collaborating organizations that sharing resources in the collaborative environment.
- **Role-to-Object** table (representing the RTO mapping model) contains resources shared between every two organizations specifying the accessing permission (cross-domain rules). This table is very important and is used by the Role-Mapping algorithm to generate the Role-Mapping rule tuples.
- **Role-to-Role** table (representing the RTR mapping model) contains all Role-Mapping rules between every two organizations.

Therefore, the inter-domain database is used while applying the RTO model by using the Role-to-Object table. Also, it is used while applying the RTR model by using the Role-to-Role table.

b. Access Control System Architecture

Figure 6 demonstrates the implemented access control system architecture. The implemented RBAC system consists of four main components; Authorization Service, Matching Service, Role-Mapping Service, and Rule Store.

- **Authorization Service**

The Authorization service receives the authorization request from the collaborative application's services and replies with access granted or denied. Each authorization request contains five parameters; Requester Organization ID, Requester ID, Target Organization ID, Target Resource ID, and Requested Permission.

This service initially checks for hits in its internal request-response Cache. In case of no hits, it forwards the request to the Matching Service. In other words, the cache saves all the incoming requests associated with the decided system.
responses. So, for each new request, the cache is checked first before executing the authorization process. If the current request exists in the Cache, the system will respond rapidly with the associated response. Otherwise, the authorization process will be held by forwarding the request to the Matching Service.

- **Matching Service**

The Matching Service gets the requester role from the intra-domain database of the requester’s organization using the requester ID. If the requester doesn’t have any roles, it replies with access denied. Otherwise, it looks up—in the Role-Mapping tuples in inter-domain database—all rule tuples that indicate that the requester role has been mapped to roles in the target organization. For each tuple, Matching Service matches the permissions of the target organization’s role with the requested permission. If there is a match, it replies back to the Authorization Service with access granted. Otherwise, it replies with access denied.

- **Role-Mapping Service**

This service is responsible for running the Role-Mapping algorithm to generate the Role-Mapping rule tuples. The generated Role-Mapping tuples are stored in the RTR table in the inter-domain database. These Role-Mapping tuples are searched with every authorization request.

For every insertion, deletion, or any modification of a RBAC intra-domain or inter-domain rule, the Role-Mapping service will be executed by the system administrator. Thus, the Role-Mapping Service will run the Role-Mapping algorithm in
order to re-generate the Role-Mapping rules. Once this service runs, the Cache in the Authorization Service is totally cleared because of the updated rules. This service is considered an offline service which runs on the background of the system. Through this service, SplitMap and the proposed DirectMap Role-Mapping algorithms will be implemented and executed to explore the generated Role-Mapping tuples from each one and perform the experiments.

- Rule-Store

The Rule-Store is divided into two database schemas; inter-domain and intra-domain.

c. Experiments Setup

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 is ranged between 1 and the maximum number of resources in the host organization. The experiment is executed for each mean value. Table 3 summarizes the used parameters’ values for the experiments in three different collaboration scales; Low, Intermediate, and High Collaboration.

5.2 Evaluation of RTR model using DirectMap Vs SplitMap algorithms

A comparative study is performed to measure the effect of the collaboration degree on the online Rule-Store size and the authorization response time in the RTR model using the proposed DirectMap algorithm against the SplitMap algorithm.

The online Rule-Store size in the RTR model is defined as the number of Role-Mapping tuples among the collaborating organizations (number of generated Role-Mapping rules in the RTR table). Whereas, the average authorization response time is calculated by running 100 different authorization requests and measuring the

| Table 3 | Parameters’ values for the experiments |
|---------|--------------------------------------|
| Parameter | Values |
| # roles in the host organization | 5, 7, 15 |
| # roles for the guest organization | 5, 10, 20 |
| # resources in the host organization | 20, 250, 500 |
| # resources in the guest organization | 20, 250, 500 |
| #resources per role | Mean = 1–500 |
| (inter-and intra-domain) | Standard deviation = 10% of mean |
authorization response time for each request. After that, the average response time of all the requests is calculated.

We tried to simulate multiple collaboration scenarios to simulate different real-life environments and execute the experiments over them. So, we made our simulation to implement the experiments through three different levels of simulated scenarios; Low, Middle, and High collaboration, using the parameters in Table 3. In each scenario, the Rule-Store size and the average authorization response time are measured.

5.2.1 Low-collaboration scenario

In the first level of collaboration, we modelled a simulated low collaboration scenario between two organizations. As shown in Table 3, both of them have five roles and twenty resources. Number of resources per role in the intra-domain and inter-domain rules is varied according to the normal distribution generation, where the mean is varied between one and twenty and the standard deviation is always 10% of the mean. 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 7 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 Fig. 7, 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. The mapping technique of the DirectMap algorithm is to map each guest role to one newly created role in the host organization. Therefore, the number of generated tuples is equal to the number of guest roles. Hence,
as shown in Fig. 7, with increasing number of resources per role in each execution, the generated number of tuples is constant (simply, equals to number of guest roles). Thus, the average number of generated rule tuples in the RTR model using DirectMap algorithm is reduced by 72.2% in average relative to SplitMap algorithm. Figure 8 depicts the average authorization response time (in milliseconds) for RTR model using SplitMap Role-Mapping algorithm against the proposed DirectMap Role-Mapping algorithm.

As shown in Fig. 8, the average response time of RTR model using SplitMap algorithm is 103 ms. By using DirectMap algorithm, the average response time is 70 ms. Therefore, the proposed DirectMap algorithm achieves better response time than SplitMap algorithm by 32%.

5.2.2 Middle-collaboration scenario

In the middle level of collaboration using the parameters in in Table 3, the host organization has seven roles, the guest organization has ten roles, and both organizations have 250 resources. Number of resources per role in the intra-domain and inter-domain rules is varied according to the normal distribution generation, where the mean is varied between one and 250 and the standard deviation is always 10% of the mean. Figure 9 depicts the number of generated Role-Mapping tuples in the RTR model using SplitMap Role-Mapping algorithm against the proposed DirectMap Role-Mapping algorithm for a range of mean values from one to 250 resources per role.

According to the results in Fig. 9, 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). Whereas, 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 doesn’t 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 10 depicts the average authorization response time (in milliseconds) for RTR model using SplitMap Role-Mapping algorithm against the proposed DirectMap Role-Mapping algorithm.

As shown in Fig. 10, the average response time of RTR model using SplitMap algorithm is 130 ms. By using DirectMap algorithm, the average response time is 67 ms. Therefore, the proposed DirectMap algorithm reduces the response time by 48.5% in average relative to the SplitMap algorithm.

5.2.3 High-collaboration scenario

In High level of collaboration, using the parameters in Table 3, the host organization has fifteen roles, the guest organization has twenty roles, and both organizations have five hundred resources. Number of resources per role in the intra-domain and inter-domain rules is varied according to the normal distribution generation, where
the mean is varied between one and five hundred, and the standard deviation is always 10% of the mean. Figure 11 depicts the number of generated Role-Mapping tuples in the RTR model using the SplitMap Role-Mapping algorithm against the proposed DirectMap Role-Mapping algorithm for a range of mean values from 1 to 500 resources per role.

According to the results in Fig. 11, 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). Whereas, the number of generated tuples using SplitMap algorithm is increased with 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 doesn’t 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 12 depicts the average authorization response time (in milliseconds) for RTR model using SplitMap Role-Mapping algorithm against the proposed DirectMap Role-Mapping algorithm.

As shown in Fig. 12, the average response time of RTR model using SplitMap algorithm is 205 ms. By using DirectMap algorithm, the average response time is 70 ms. Therefore, the proposed DirectMap algorithm reduces the authorization response time by 65.8% in average relative to the SplitMap algorithm.

Therefore, the proposed DirectMap Role-Mapping algorithm achieves more reduction in the Rule-Store size and the authorization response time relative to the SplitMap Role-Mapping algorithm in all the simulated collaboration scenarios. So, 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

The second comparative study is performed 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 against 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 (number of rules in the Role-to-Object table). Whereas, the online Rule-Store size in the RTR model is defined as the number of Role-Mapping tuples among the collaborating organizations (number of generated Role-Mapping rules in the RTR table). 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 13 depicts the number of rule tuples in the RTO model against the RTR model using the proposed DirectMap Role-Mapping algorithm for a range from one to twenty resources per role. As shown in Fig. 13, 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 14 depicts the average authorization response time (in milliseconds) for RTO model against the RTR model using the proposed DirectMap Role-Mapping algorithm.

As shown in Fig. 14, the average response time in RTR model using the proposed DirectMap is 70 ms while in RTO model is 65 ms. So, the RTR mapping model
using DirectMap isn’t better than RTO model in low collaboration scenario. Our investigation revealed that, in the low collaboration scenario, the number of shared resources is low which causes low number of rule tuples in the RTO model. On the other hand, 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 longer 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 15 depicts the number of rule tuples in RTO model against the RTR model using the proposed DirectMap Role-Mapping algorithm for a range from 1 to 250 resources per role.
As shown in Fig. 15, 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 16 depicts the average authorization response time (in ms) for RTO model against RTR model using the proposed DirectMap Role-Mapping algorithm.

As shown in Fig. 16, the average response time in RTO model is 81 ms while in RTR model using DirectMap Role-Mapping algorithm is 67 ms. Therefore, the RTR model using the proposed DirectMap Role-Mapping algorithm reduces the authorization response time by 17.2% in average relative to the RTO model.
5.3.3 High-collaboration scenario

Figure 17 depicts the number of rule tuples in RTO model against RTR model using the proposed DirectMap Role-Mapping algorithm for a range from 1 to 500 resources per role.

As shown in Fig. 17, 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 18 depicts the average authorization response time (in milliseconds) for RTO model against RTR model using the proposed DirectMap Role-Mapping algorithm.

As shown in Fig. 18, the average response time in RTO model is 98 ms while in RTR model using DirectMap is 70 ms. Therefore, the RTR model using the
proposed DirectMap Role-Mapping algorithm reduces the authorization response time by 28.5% in average relative to the RTO model.

Thus, the proposed DirectMap Role-Mapping algorithm is much suitable to be used in high collaboration environments. It achieves more savings in the online Rule-Store size and the authorization response time with increasing the number of shared resources among the collaborating organizations. Unfortunately, the response time in low collaboration scenario in RTR model using the proposed DirectMap algorithm is longer than the RTO model. To overcome this limitation, a concurrent implementation of the RTR model using the proposed algorithm is introduced in the following section.

6 Concurrent implementation of RTR model using the proposed DirectMap Algorithm

According to the performed comparative studies, it is found that the proposed DirectMap algorithm achieves better results than SplitMap algorithm with respect to the Rule-Store size and the authorization request’s response time. But, it doesn’t reduce the response time relative to the traditional RTO mapping model in low collaboration environments. Based on these results, the RTR mapping model is enhanced by implementing it in a concurrent manner to overcome the limitation of the DirectMap Role-Mapping algorithm and reduce the authorization request’s response time in all levels of collaboration [36, 37]. The used concurrent approach is called Single-Program Multiple-Data (SPMD) computational model [38–40].

The SPMD Concurrent approach is applied on the RTR mapping model with DirectMap Role-Mapping algorithm using Multi-Threading technique on Single Machine. We used Multi-Threading in java for concurrent implementation. The host machine is a Quad-Core machine. The machine’s RAM capacity is 4 GB. This approach is applied on different virtual machines in order to measure the improvement of response time with different resource power. VMware Workstation tool is used for creating Virtual Machines and configuring the number of cores for each machine. The created virtual machines are as follows:

- Virtual machine with 2 cores.
- Virtual machine with 3 cores.
- Virtual machine with 4 cores.

In each virtual machine, we applied this approach and compared its average Authorization Response Time against the RTO Mapping Model. The flowchart of the concurrent implementation of the RTR mapping model with DirectMap Role-Mapping algorithm using Multi-Threading technique is presented in Fig. 19. By using this approach, assume the following request parameters:

- A requester User\textsubscript{x} from organization Org\textsubscript{1} requests to access a resource r with permission p in the target organization Org\textsubscript{2}.  

 Springer
The Role-Mapping rules (i.e., Role-to-Role table) are generated by the Direct-Map Role-Mapping algorithm. Therefore, each role in Org$_1$ is mapped to only one new role in Org$_2$. The Matching Service gets the Role ID list of User$_X$ from the local database of Org$_1$. Assume that User$_X$ has three roles (i.e., Role$_1$, Role$_2$, and Role$_3$) in his organization Org$_1$. By applying SPMD concurrent approach using Multi-Threading technique, searching for matching task is divided into sub-tasks. The sub-tasks are distributed over multiple processes with different input (see Fig. 19). The processes are generated based on the requester’s roles.

In the mentioned example, User$_X$ has three roles in his organization. Thus, three processes are generated to execute the main task. Each process takes one user’s role and executes the following sub-task:

![Flow Chart of Applying SPMD Concurrent Model with Multi-Threading technique on the RTR Mapping Model with DirectMap Role-Mapping algorithm](image)

**Fig. 19** Flow Chart of Applying SPMD Concurrent Model with Multi-Threading technique on the RTR Mapping Model with DirectMap Role-Mapping algorithm
1. Getting its Mapped Role in the target organization $Org_2$ from the Role-to-Role table in the central Rule-Store.
2. Checking if the Mapped Role has the privilege to access the resource $r$ with the permission $p$, by looking up in the Role-Resource-Permission table in the local database of the target organization $Org_2$.
3. If there is matching, the process returns true. Else, it returns false.

After all the processes finished, the system checks if there is a hit in one or more of the processes. In such case, the system will grant $User_X$ to access the requested resource $r$ with the specified permission $p$. If there are no hits, the system will deny $User_X$ from accessing the requested resource.

All the processes execute the tasks in parallel at the same time, which reduces the searching time of the authorization request. Therefore, the authorization request’s response time is improved. On the other hand, number of processes in the introduced procedure doesn’t cause a big problem with the computation power. For each authorization request, number of processes will be equal to the number of requester’s roles in his organization, which will not be a large number even in the high collaboration environments. In addition, the computation power is minimized with increasing number of cores in the used machine.

6.1 Performance evaluation of authorization response time

After introducing the proposed enhanced procedure of the RTR model with the DirectMap Role-Mapping algorithm using Multi-Threading technique, the next step is to evaluate its performance for all collaboration environments. The Authorization Response Time Measurement Experiment is performed to measure the average authorization response time for the RTR model using DirectMap Role-Mapping algorithm with applying SPMD concurrent approach using Multi-Threading technique against the RTO model in different collaboration scenarios. This comparison is performed in different virtual machines with different resource power to evaluate the performance with increasing number of machine’s cores. The first used virtual machine has 2 cores, the second one has 3 cores, and the third one has 4 cores.

6.1.1 Low-collaboration scenario

Figure 20 depicts the average authorization response time (in milliseconds) of the RTR model using DirectMap Role-Mapping algorithm with applying SPMD concurrent approach using Multi-Threading technique against the RTO model for different virtual machines in Low collaboration scenario.

As shown in Fig. 20, in a machine with 2 cores, the average authorization response time in RTO model is 79 ms. While in RTR model using SPMD with Multi-threading is 74 ms. Therefore, this approach reduces the authorization response time by 6.3% relative to the RTO model.

In a machine with 3 cores, the average authorization response time in RTO model is 76 ms. While in RTR model using SPMD with Multi-threading is 61 ms.

 Springer
Therefore, this approach reduces the authorization response time by 19.7% relative to the RTO model.

In a machine with 4 cores, the average authorization response time in RTO model is 70 ms. While in RTR model using SPMD with Multi-threading is 59 ms. Therefore, this approach reduces the authorization response time by 15.7% relative to the RTO model.

6.1.2 Middle-collaboration scenario

Figure 21 depicts the average authorization response time (in milliseconds) of the RTR model using DirectMap Role-Mapping algorithm with applying SPMD concurrent approach using Multi-Threading technique against the RTO model for different virtual machines in Middle collaboration scenario. As shown in Fig. 21, in a machine with 2 cores, the average authorization response time in RTO model is 110 ms. While in RTR model using SPMD with Multi-threading is 76 ms. Therefore, this approach reduces the authorization response time by 30.9% relative to the RTO model.

In a machine with 3 cores, the average authorization response time in RTO model is 94 ms. While in RTR model using SPMD with Multi-threading is 63 ms. Therefore, this approach reduces the authorization response time by 32.9% relative to the RTO model.

In a machine with 4 cores, the average authorization response time in RTO model is 90 ms. While in RTR model using SPMD with Multi-threading is 57 ms. Therefore, this approach reduces the authorization response time by 36.6% relative to the RTO model.
6.1.3 High-collaboration scenario

Figure 22 depicts the average authorization response time (in ms) of the RTR model using DirectMap Role-Mapping algorithm with applying SPMD concurrent approach using Multi-Threading technique against the RTO model for different virtual machines in High collaboration scenario.

As shown in Fig. 22, in a machine with 2 cores, the average authorization response time in RTO model is 127 ms. While in RTR model using SPMD with Multi-threading is 77 ms. Therefore, this approach reduces the authorization response time by 39.3% relative to the RTO model.
In a machine with 3 cores, the average authorization response time in RTO model is 120 ms. While in RTR model using SPMD with Multi-threading is 73 ms. Therefore, this approach reduces the authorization response time by 39.1% relative to the RTO model.

In a machine with 4 cores, the average authorization response time in RTO model is 110 ms. While in RTR model using SPMD with Multi-threading is 61 ms. Therefore, this approach reduces the authorization response time by 44.5% relative to the RTO model.

7 Discussion

The goal of this paper is to introduce the RTR mapping technique for the RBAC model using a new Role-Mapping algorithm which aims 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.

The difference between Role-Mapping techniques of the SplitMap and DirectMap algorithms could be clarified by considering the following scenario; assume that a guest role (i.e., Rolej) requests to consume n resources’ permissions (i.e., P1 to Pn), these permissions are distributed over n host roles (i.e., Role1 to Role n). By using the SplitMap Role-Mapping algorithm, each host role (from Role1 to Role n) that has requested resource permission is split. In other words, each resource’s permission (from P1 to Pn) 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, Rolej. So, n role mapping tuples are generated which increases the online rule store size. By using the proposed DirectMap Role-Mapping algorithm, there is no need to split different roles, but it creates (for the guest role, Rolej) only one new role (i.e., Rolejx) in the host organization. The algorithm assigns all the requested resources’ permissions (i.e., P1 to Pn) to the newly created role, Rolejx, and maps it to the guest role, Rolej. So, only one role mapping tuple is generated which minimizes the online rule store size and the local database size of the host organization.

To evaluate the performance of the proposed DirectMap algorithm, a comparative study is performed between the RTR mapping technique using the proposed DirectMap algorithm against 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 4.

According to the results in Table 4, it’s found that DirectMap algorithm generates minimal number of rule tuples which decreases the Rule-Store size by 83.7% in average relative to the SplitMap algorithm. Also, DirectMap algorithm reduces the authorization response time by 48.7% in average relative to SplitMap algorithm.
Another comparative study is 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 5.

According to the results in Table 5, it is found that the RTR model using DirectMap algorithm reduces the online Rule-Store size by 98.2% in average relative to the traditional RTO model. Also, it reduces the authorization request’s response time by 12.6% in average relative to the RTO model. But in low collaboration

### Table 4 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 | 48.7 |
| Middle | 130 | 67 | 48.5 |
| High | 205 | 70 | 65.8 |

### Table 5 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 | 98.2 |
| Middle | 2109 | 10 | 99.5 |
| High | 8674 | 20 | 99.7 |
| Average authorization response time (in ms) |  |  |  |
| Low | 65 | 70 | − 7.7 | 12.6 |
| Middle | 81 | 67 | 17.2 |
| High | 98 | 70 | 28.5 |
scenario, it is found that the RTR model using DirectMap algorithm has a delay in the response time as it is increased by 7.7% relative to the RTO model.

Based on the observed results, it is found that the performance of DirectMap algorithm is better than the SplitMap algorithm. However, it needs more enhancement in the authorization response time relative to the RTO model. Thus, the second contribution of this paper is enhancing the RTR model using the DirectMap algorithm in order to overcome its limitation by applying the Single-Program Multiple-Data (SPMD) computational approach on the RTR mapping model with DirectMap algorithm using Multi-Threading technique.

The last comparative study is performed in order to evaluate the performance of applying SPMD approach on RTR mapping model using Multi-Threading in multiple virtual machines with different number of cores. The summary of the comparative results of this proposed approach against the RTO model is presented in Table 6.

According to the results in Table 6, it is found that this approach achieves great results by reducing the authorization request’s response time by 25.5% in 2 cores machine, 30.6% in 3 cores machine, and 32.3% in 4 cores machine in average relative to the RTO model. Thus, it is found that the improvement percentage of this approach is increased with increasing number of cores of the used machine.

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 SPMD approach on RTR mapping model with the proposed DirectMap

| Number of cores | Collaboration scenario | RTO | SPMD with multi-threading | SPMD with multi-threading Improvement percentage relative to RTO (%) |
|-----------------|------------------------|-----|--------------------------|------------------------------------------------------------------|
| Average authorization response time (in ms) | Low | 79 | 74 | 6.3 | 25.5 |
|                  | Middle | 110 | 76 | 30.9 |
|                  | High | 127 | 77 | 39.3 |
| 2 cores          | Low | 76 | 61 | 19.7 | 30.6 |
|                  | Middle | 94 | 63 | 32.9 |
|                  | High | 120 | 73 | 39.1 |
| 3 cores          | Low | 70 | 59 | 15.7 | 32.3 |
|                  | Middle | 90 | 57 | 36.6 |
|                  | High | 110 | 61 | 44.5 |
algorithm contributed in achieving more savings in the authorization response time in different collaboration cloud environments.

8 Conclusions and future work

In today’s collaborative applications, the involved parties have to share their resources across the administrative domains. The increasing nature of such applications makes the federated access control systems gaining more ground recently. Besides security, the performance is considered a cornerstone for successful adoption of such access control systems. The work in this paper focuses on improving the authorization performance in collaborative cloud environments by reducing the online rule store size and the authorization request’s response time. It addresses the authorization performance of the RBAC system with increasing the number of collaborating organizations and/or increasing the number of shared resources in collaborative cloud environments. Also, it studies two different RBAC mapping techniques; Role-to-Object (RTO) model and RTR model. According to the experimental results, it is found that using the RTR 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. The first contribution of this paper is introducing a new Role-Mapping algorithm, DirectMap, to be used in the RTR model. The first comparative study is performed to evaluate the proposed algorithm against the SplitMap algorithm. According to the results, it is found that the proposed DirectMap Role-Mapping algorithm achieves more savings in the online rule store size and also more reduction in the authorization response time relative to the SplitMap algorithm. The second comparative study is performed to evaluate the proposed algorithm against the RTO model. According to the results, it is found that the RTR model using DirectMap algorithm achieves more savings in the online Rule-Store size and reduces the authorization request’s response time relative to the RTO model. But in low collaboration scenario, it is found that the RTR model using DirectMap algorithm has a delay in the response time relative to the RTO model. Another 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. The second contribution of this paper is enhancing the RTR model using the DirectMap algorithm in order to overcome its limitation by improving the authorization request’s response time in all the simulated levels of collaboration. The proposed way is to perform concurrent implementation of the RTR mapping model with DirectMap algorithm using the Single-Program Multiple-Data (SPMD) computational approach. The last comparative study is performed to evaluate this proposed approach against the RTO model with respect to the authorization request’s response time. According to the results, it is found that this approach achieves great results by reducing the authorization request’s response time in all the simulated levels of collaboration. Also, it
is noticed that the performance is improved with increasing number of cores in the used machine. Therefore, the proposed approach is suitable to be used in all collaborative cloud environments.

As a future work, we need to study the efficient cache invalidation algorithms for the request-response cache in the Authorization Service component instead of clearing it. Another issue that could be considered in the future work is the need of getting a real-life RBAC from the industry. Obtaining a real-life RBAC data of organizations was very hard. We couldn't find RBAC rules from the industry as they are considered confidential data. We tried to simulate the real-life RBAC rules by using some statistical and probability procedures. Different means and standard deviation values are used in the Normal Distribution generation of RBAC rules (i.e., mapping objects to roles). The used parameters in the introduced simulated scenarios can be changed in the future work in order to generate different RBAC rules and simulate other uncovered cases in this paper. The used parameters in the introduced simulated scenarios (i.e., Mean values, Standard Deviation values, number of collaborating organizations) can be changed in the future work in order to generate different RBAC rules and simulate other uncovered cases in this paper.

Acknowledgements This research did not receive any specific grant from funding agencies in the public, commercial or not-for-profit sectors.

Author contributions DA: Software, Methodology, Validation, Investigation, Project administration. HAH: Writing—Review & Editing, Supervision. FAO: Resources, Writing—Review & Editing, Supervision.

Funding Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB). No funding was received to assist with the preparation of this manuscript.

Data availability Not applicable.

Code availability Not applicable.

Declarations

Conflict of interest The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

Springer
References

1. Wang, L., et al.: Cloud computing: a perspective study. N. Gener. Comput. 28(2), 137–146 (2010). https://doi.org/10.1007/s00354-008-0081-5

2. Office of Management and Budget, “Federal cloud computing strategy,” Office of Management and Budget, 2019. https://cloud.io.gov/. Accessed 1 May 2019

3. Zhang, R., Liu, L.: Security models and requirements for healthcare application clouds. In 2010 IEEE 3rd international conference on cloud computing, pp. 268–275 (2010). https://doi.org/10.1109/CLOUD.2010.62

4. Gopalan, G., Negi, A., Sastry, V.N.: A cross-domain role mapping and authorization framework for RBAC in grid systems. Int. J. Comput. Sci. Appl. 6(1), 1–12 (2009)

5. Kaur, U., Singh, D.: Comparative analysis of access control models. Int. J. Comput. Trends Technol. 29(3), 132–135 (2015). https://doi.org/10.1445/22312803/IJCTT-V29P123

6. Madarri, T., Al-Rabeei, S.A.: Security fundamentals: access control models. Int. J. Interdiscip. Theory Pract. 7, 259–262 (2015)

7. Hashizume, K., Rosado, D.G., Fernández-Medina, E., Fernandez, E.B.: An analysis of security issues for cloud computing. J. Internet Serv. Appl. 4(1), 5 (2013). https://doi.org/10.1186/1869-0238-4-5

8. Howell, J., Kotz, D.: End-to-end authorization. In: Proceedings of the 4th conference on symposium on operating system design & implementation (2000)

9. Takabi, H., Joshi, J.B.D., Ahn, G.-J.: Security and privacy challenges in cloud computing environments. IEEE Secur. Priv. Mag. 8(6), 24–31 (2010). https://doi.org/10.1109/MSP.2010.186

10. Gerges, S., Khattab, S., Hassan, H., Omara, F.A.: Scalable multi-tenant authorization in highly-collaborative cloud applications. Int. J. Cloud Comput. Serv. Sci. (2013). https://doi.org/10.11591/closer.v22.i2.1665

11. Tang, B., Li, Q., Sandhu, R.: A multi-tenant RBAC model for collaborative cloud services. In: 2013 eleventh annual conference on privacy, security and trust, pp. 229–238 (2013). https://doi.org/10.1109/PST.2013.6596058

12. Calero, J.M.A., Edwards, N., Kirschnick, J., Wilcock, L., Wray, M.: Toward a multi-tenancy authorization system for cloud services. IEEE Secur. Privacy Mag. 8(6), 48–55 (2010). https://doi.org/10.1109/MSP.2010.194

13. Xinchaoy, Z., Rui, L., Xingquan, Z.: Advances on QoS-aware web service selection and composition with nature-inspired computing. CAAI Trans. Intell. Technol. 4(5), 159–174 (2019)

14. Namasudra, S., Roy, P.: Time saving protocol for data accessing in cloud computing. IET Commun.-Inst. Eng. Technol. 11(10), 1558–1565 (2017). https://doi.org/10.1049/ietc.2016.0777

15. Namasudra, S., Chakraborty, R., Majumder, A., Moparthi, N.R.: Securing multimedia by using DNA-based encryption in the cloud computing environment. ACM Trans. Multimedia Comput. Commun. Appl. (2020). https://doi.org/10.1145/3392665

16. Namasudra, S., Roy, P.: A new secure authentication scheme for cloud computing environment. Concur. Comput. (2016). https://doi.org/10.1002/cpe.3864

17. Algiluyev, R.M., Algiluyev, R.M., Sukhostat, L.V.: Efficient algorithm for big data clustering on single machine. CAAI Trans. Intell. Technol. 5(1), 9–14 (2020)

18. Namasudra, S., Roy, P., Vijayakumar, P., Audithan, S., Balasamy, B.: Time efficient secure DNA based access control model for cloud computing environment. Futur. Gener. Comput. Syst. 73, 90–105 (2017). https://doi.org/10.1016/j.future.2017.01.017

19. Namasudra, S.: Fast and secure data accessing by using DNA computing for the cloud environment. IEEE Trans. Serv. Comput. India (2020). https://doi.org/10.1109/TSC.2020.3046471

20. Shuai, L., Guoyin, W., Jie, Y.: Survey on cloud model based similarity measure of uncertain concepts. CAAI Trans. Intell. Technol. 4(4), 223–230 (2019)

21. Rao, P., Lin, D., Bertino, E., Li, N., Lobo, J.: Fine-grained integration of access control policies. Comput. Secur. 30(2–3), 91–107 (2011). https://doi.org/10.1016/j.cose.2010.10.006

22. Alansari, S., Paci, F., Sassone, V.: A distributed access control system for cloud federations. In: 2017 IEEE 37th international conference on distributed computing systems (ICDCS), pp. 2131–2136 (2017). https://doi.org/10.1109/ICDCS.2017.241

23. Younis, Y.A., Kifayat, K., Merabti, M.: An access control model for cloud computing. J. Inf. Secur. Appl. 19(1), 45–60 (2014). https://doi.org/10.1016/j.jisa.2014.04.003
24. Leandro, M.A.P., Nascimento, T.J., dos Santos, D.R., Westphall, C.M., Westphall, C.B.: Multi-tenancy authorization system with federated identity for cloud-based environments using shibboleth. In: ICN 2012: the eleventh international conference on networks, pp. 88–93 (2012)

25. InCommon, “Shibboleth,” InCommon. http://shibboleth.internet2.edu/. Accessed 09 Dec 2019

26. Chen, L., Crampton, J.: Inter-domain role mapping and least privilege. In: Proceedings of the 12th ACM symposium on access control models and technologies - SACMAT ’07 (2007). https://doi.org/10.1145/1266840.1266866

27. Ferraiolo, D.F., Kuhn, D.R.: Role-based access controls. In: 15th national computer security conference, pp. 554–563 (2009)

28. Sandhu, R.S., Coyne, E.J., Feinstein, H.L., Youman, C.E.: Role-based access control models. Computer 29(2), 38–47 (1996). https://doi.org/10.1109/2.485845

29. Sandhu, R., Ferraiolo, D., Kuhn, R.: The NIST model for role-based access control. In: Proceedings of the fifth ACM workshop on role-based access control - RBAC ’00, pp. 47–63 (2000). https://doi.org/10.1145/344287.344301

30. Hu, J., Li, R., Lu, Z.: On role mappings for RBAC-based secure interoperaion. In: 2009 third international conference on network and system security, pp. 270–277 (2009). https://doi.org/10.1109/NSS.2009.76

31. Kamath, A., Liscano, R., El Saddik, A.: User-credential based role mapping in multi-domain environment. In: The 2006 international conference on privacy, security and trust: bridge the gap between PST technologies and business services, Canada, October 30–November 1 (2006)

32. Lv, B., Zhang, D., Mao, R., Yang, H.: A multi-level cross-domain access control model based on role mapping. In: 2016 4th international conference on mechanical materials and manufacturing engineering (2016)

33. Du, S., Joshi, J.B.D.: Supporting authorization query and inter-domain role mapping in presence of hybrid role hierarchy. In: The eleventh ACM symposium on access control models and technologies, pp. 228–236 (2006)

34. Shehab, M., Bertino, E., Ghafoor, A.: SERAT: SEcure role mApping technique for decentralized secure interoperabilty. In: The10th ACM symposium on access control models and technologies, pp. 159–167 (2005)

35. Shafiq, B., Joshi, J.B.D., Bertino, E., Ghafoor, A.: Secure interoperation in a multidomain environment employing RBAC policies. IEEE Trans. Knowl. Data Eng. 17(11), 1557–1577 (2005)

36. Ják, J.: An Introduction to Parallel Algorithms. Addison Wesley Longman Publishing Co., Inc., Redwood City (1992)

37. Tahir, M.: Programming concurrency in C++ - part one, C#Corner (2018). https://www.c-sharp-corner.com/article/programming-concurrency-in-cpp-part-1. Accessed 01 May 2021

38. Darema, F., George, D.A., Norton, V.A., Pfister, G.F.: A single-program-multiple-data computational model for EPEX/FORTRAN. Parallel Comput. 7(1), 11–24 (1988). https://doi.org/10.1016/0167-8191(88)90094-4

39. Meesala, S.K., Khilar, D.P.M., Shrivastava, D.A.K.: The parallel architecture approach, single program multiple data (SPMD) implementation on clusters of terminals using Java RMI. IJCER 6(3), 16–23 (2016)

40. Bradley, D.: SPMD: single program multiple data streams. Tufts University. 2016. [Online]. https://slideplayer.com/slide/7539656. Accessed 30 Jul 2021

Publisher’s Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.