Towards Collaboration-Aware Resource Sharing in Research Computing Infrastructures

Conference Paper · November 2025

CITATIONS
0

READS
25

5 authors, including:

Souradip Nath
Arizona State University
6 PUBLICATIONS **64** CITATIONS
SEE PROFILE

Jaejong Baek
Arizona State University
19 PUBLICATIONS **93** CITATIONS
SEE PROFILE

Carlos Rubio Medrano
Texas A&M University – Corpus Christi
47 PUBLICATIONS **314** CITATIONS
SEE PROFILE

Towards Collaboration-Aware Resource Sharing in Research Computing Infrastructures

Souradip Nath[†], Ananta Soneji[†], Jaejong Baek[†], Carlos Rubio-Medrano[‡], and Gail-Joon Ahn[†]
[†]Arizona State University, [‡]Texas A&M University – Corpus Christi
[†]{snath8, asoneji, jbaek7, gahn}@asu.edu, [‡]carlos.rubiomedrano@tamucc.edu

***Abstract*—As scientific research grows increasingly collaborative, data-driven, and computation-intensive, Research Computing Infrastructures (RCIs) have emerged as critical platforms for enabling scientific collaboration across diverse disciplines. These infrastructures offer advanced computational and storage solutions essential for collaborative research, yet securing shared access to such resources remains a significant challenge. Consequently, there has been limited investigation into how collaboration context is conceptualized and leveraged during access provisioning, especially within RCIs. In this paper, we investigate existing resource sharing practices within RCIs, revealing key gaps in their ability to support secure authorization within dynamic, project-driven collaborative workflows. Building on these insights, we introduce a framework called CLEARS (Collaboration-Aware Authorization for Resource Sharing) that formally represents collaboration contexts and operationalizes them to guide secure, context-aware, and dynamically evolving resource sharing authorization throughout the collaboration lifecycle. Through experiments, we demonstrate that CLEARS delivers precise access enforcement under evolving collaborative scenarios while maintaining minimal execution overhead.**

I. Introduction

The multifaceted nature of the term 'cyberinfrastructure' is better introduced by referring to the widely cited definition:

"Cyberinfrastructure consists of computing systems, data storage systems, advanced instruments and data repositories, visualization environments, and people, all linked together by software and high-performance networks to improve productivity and enable breakthroughs not otherwise possible." [1]

RCIs are critical cyberinfrastructures owned and managed by research institutions, providing accelerated high-performance computing resources and data storage solutions to support a diverse scientific community [2]–[5]. Modern science, ranging from training large language models to genome sequencing, relies on massive datasets and complex computation pipelines [6]–[9]. To meet these demands, RCIs consolidate and centralize resources that were once siloed across research groups, offering a cost-effective and unified platform for scientific progress. As scientific research becomes increasingly collaborative and interdisciplinary, RCIs play a vital role in enabling secure sharing of computational and data resources. This evolution has transformed RCIs into socio-technical infrastructures that support scientific collaboration across distributed research teams.

However, managing shared resources across multiple research groups and navigating complex collaborative workflows presents significant challenges. Resource sharing in such settings must account for both the technical intricacies of the infrastructure and the potential consequences if access around shared resources is not securely regulated [10], [11]. Poorly managed access can lead to serious consequences, including confidentiality breaches and resource misuse, as exemplified by the recent FHS data breach at Boston University [12].

However, effective access control is challenging due to the need to balance security with the dynamic and flexible nature of collaboration. Current practices often rely on fragmented, ad hoc resource sharing mechanisms that fail to reflect the evolving structure of collaborations, leading to coarse-grained and overprivileged access, and challenges in privilege revocation [13]. Despite the highly collaborative nature of these infrastructures, little is known about how collaboration contexts are conceptualized and leveraged to guide access decisions within RCIs.

Our research addresses this gap by investigating the following research questions:

- **RQ1:** What are the existing challenges around access control and resource sharing practices within RCIs?
- **RQ2:** From an access control perspective, what unique requirements must be addressed to support effective and secure resource sharing in RCIs?
- **RQ3:** How can collaboration contexts be conceptualized, designed, and utilized to enable secure and flexible resource sharing authorization within RCIs?

To answer these questions, we perform a technical exploration to identify key requirements to support collaboration-aware resource sharing within RCIs. Drawing on these findings, we introduce **CLEARS**, a framework that introduces novel constructs such as the *Collaboration Network*, along with the *Privilege Expansion* and *Privilege Contraction* algorithms to allow user-centric flexibility in resource sharing. Through experiments, we demonstrate that CLEARS effectively captures the collaborative structure underlying resource sharing and delivers precise access enforcement under evolving collaborative scenarios.

Thus, the main contributions of this paper are as follows:

- We investigate access control and resource sharing practices within a real-world RCI (**RQ1**) to derive requirements essential for secure and efficient resource sharing (**RQ2**) in RCIs (§ III).

- We propose CLEARS (§ IV), a systemic approach that formalizes and manages collaborative contexts, enabling efficient privilege management to support secure and flexible resource sharing (**RQ3**).
- We implement a prototype of CLEARS, and evaluate its behavior through scenario-based correctness analysis and performance measurements in a simulated RCI environment (§ V).

The complete codebase, along with detailed guides for reproducing the experimental environment and evaluation results, is available in our project's GitHub repository [14].

II. Background and Related Work

In this section, we explore the RCI foundational concepts and recent advances in access control, underscoring the necessity for specialized approaches to regulate scientific resource sharing effectively.

Research Computing Infrastructure (RCI). At the core of RCIs are sophisticated supercomputing clusters, comprising thousands of high-performance CPU cores and hundreds of GPU accelerators, which are organized into partitions, with multiple partitions forming the cluster [3]–[5]. This massively parallel processing power allows allows executing and scaling computationally intensive tasks, such as complex simulations, data analytics, and scientific modeling. Complementing the HPC capabilities, RCIs also provide robust data storage solutions for managing and sharing large research datasets, often in terabytes. These include shared network file systems, project-specific repositories, and archival storage. High-speed, high-performance networks, typically powered by InfiniBand (IB), ensure fast and reliable connectivity between facilities.

Acknowledging the vital role of RCIs in advancing scientific innovation, recent efforts have significantly focused on enhancing the security posture of such environments [10]. Research has increasingly focused on secure distributed sharing of research data [15]–[17], infrastructure-level security [18]–[20], and human-centered cybersecurity [13], with the overarching goal of protecting scientific data, computation, collaboration workflows, and the infrastructure itself.

Traditional Access Control and Beyond. Classic access control models like Discretionary Access Control (DAC) [21], Mandatory Access Control (MAC) [22], and Role-Based Access Control (RBAC) [23] are widely used in large enterprise applications where resources and access needs are relatively static. However, these models face challenges in dynamic and distributed environments typical of modern collaborative systems [24]. Despite their limitations, these models inspire the development of paradigms, such as the Role-Based Delegation Model [25], which allows entities in distributed environments to delegate privileges based on rule-based policies, and the Team-Based Access Control model [26], which introduced "teams" to represent users in specific roles accessing shared resources for tasks.

Similarly, the Collaborative Access Control (CollAC) model [27] expands user-object relationships beyond ownership to increase transparency in access decisions. Relationship-based access control (ReBAC) model [28] considers interpersonal relationships between resource owners and accessors. To achieve more fine-grained control, Attribute Based Access Control (ABAC) was introduced that evaluates user, resource, and environmental attributes dynamically [29]. More recently, Activity-Centric Access Control [30] was proposed, which considers the relationship between activities in smart collaborative systems for access mediation. In summary, while traditional and recently-introduced paradigms offer a solid foundation, they are mainly geared towards enterprise applications and other collaborative contexts. There is limited research addressing the complex nature of scientific collaborations and resource sharing in RCIs. While these existing models can be adapted to RCI settings, we believe that explicitly conceptualizing collaboration contexts and leveraging them to guide secure, flexible resource sharing is essential for addressing the unique demands of scientific collaborations (§ IV-E).

III. Requirement Elicitation

In this section, we outline our approach to investigate the existing access control practices and elicit requirements to support secure and efficient resource sharing within RCIs.

A. System Exploration

An in-depth system-level exploration of an RCI was conducted to gain a clear understanding of its architectural components, user and resource management processes, and existing access control mechanisms. We proceeded as follows:

(1) Selection of the RCI Environment: We chose a representative RCI [4] that we had access to, allowing for direct hands-on system navigation. This RCI included two clusters containing thousands of computing nodes, hundreds of GPUs, and petabyte-scale storage, supporting nearly 1,000 active users, reflecting the typical scale of real-life RCIs. [1]

(2) System Navigation: We utilized *non-administrative* user accounts to reproduce tasks that users typically perform for collaborative resource sharing. This included creating files and directories, observing and updating default privileges, submitting jobs while modifying various parameters to assess control levels, and interacting with the admins to set up shared access to project resources. This informed our understanding of the collaborative resource sharing use cases (§ III-B).

(3) Exploration of Users and Resources: Through hands-on system navigation, we explored existing user and group management practices, along with the available resources, thereby identifying groups, directory structures, ownerships, configurations, and default permissions.

(4) Investigation of Access Control Methods: We investigated existing access control mechanisms supported within the constraints of *non-administrative* privileges, including DAC mechanisms to restrict access to files and directories [21], and identified *administrative* privileges necessary for advanced

[1] To ensure that our observations extend beyond a single platform, we further validated our findings by analyzing publicly available user documentation from other RCIs [2], [3], [5], [31], [32]

configurations such as job scheduling, group management, and project-wide privilege updates.

Observations. Our exploration revealed insights into the existing landscape of *user*, *resource*, and *privilege management* within RCIs, aligning with findings from a recent multi-stakeholder qualitative study on scientific collaboration and privilege management [13].

Users can belong to multiple groups, such as *faculty groups*, for users collaborating under the same faculty, and *project groups* for extended cross-faculty collaborations. RCI resources can broadly be classified into two categories: *computing partitions* and *data directories*. Partitions could be categorized based on job priorities, as some of them are uniformly prioritized across all users (*public partitions*), while others are prioritized for some user or group of users (*private partitions*). Resource allocation and access for these resources are typically managed by job schedulers, such as Slurm [33], in conjunction with systems like Pluggable Authentication Module (PAM) [34]. Data directories can be categorized based on their purpose, ownership, and shareability. For instance, they can range from shareable temporary storage (*scratch* space) to non-shareable long-term repositories (*home* directories), and customized project-specific storage accessible by a group of users, (*data* directories), managed by file management such as POSIX permissions [21].

From an access control perspective, resource sharing, as a whole, is managed through a combination of *user* and *group-level permissions*. Projects typically use group-level permissions for data sharing, with all collaborators granted *uniform* access, while further user-to-user sharing relies on basic DAC mechanisms where the resource owner decides access. Access to computing resources is controlled by the job scheduler, which enforces permissions based on configuration specifications like Slurm's settings for $AllowGroups$, $AllowAccounts$, and $AllowQos$ for each computational partition. However, these can only be updated by administrators.

B. An Illustrative Collaboration Workflow

With an overall understanding of the RCI environment, we now present an illustrative scenario to demonstrate the practical application of our exploration.

As shown in Figure 1, the workflow involves four researchers: $Alex(A)$, $Bailey(B)$, $Cathy(C)$, and $Drew(D)$ *collaborating* within the timeline of a project Pr_1.

$Alex$, a theoretical biologist, realizing that the project Pr_1 requires extensive computational tasks, such as data analysis and simulations, collaborates with two other researchers: $Bailey$ for computational simulations, and $Cathy$, for analyzing simulation results. Together, they use shared resources within an RCI. $Erin(E)$, an RCI administrator, assists them with privilege management.

Within RCI, each user owns specific objects such as files, directories, computational partitions, etc. We denote such an object as o_i^{owner} $(i \in \mathbb{N})$. For example, objects owned by $Alex(A)$ are denoted as $\{o_1^{\text{A}}, o_2^{\text{A}}, o_3^{\text{A}}, ...\}$. We also assume that by default only $Alex$ has access to these objects, and only he

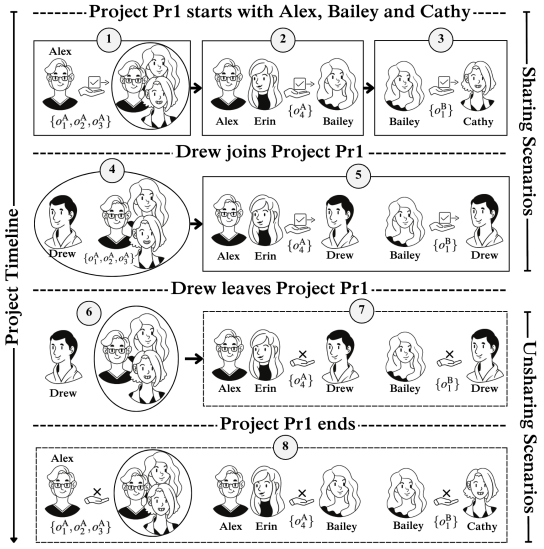

Fig. 1: An Illustrative Resource Sharing Scenario within Project Pr_1 Involving Different Users and Resources.

can further share this access with others. Here, *share* means providing access to the objects for performing valid operations (such as *read, write, submit jobs*, etc.).

The following are the activities throughout the collaborative workflow within the timeline of Pr_1:

(1) As Pr_1 starts, all collaborators engaged in the project require access to specific common files and datasets currently owned by $Alex$. Therefore, $Erin$ creates a group named grp_1 and adds $Alex$, $Bailey$, and $Cathy$. $Alex$ shares required files and directories, abstracted as $\{o_1^{\text{A}}, o_2^{\text{A}}, o_3^{\text{A}}\}$, with all other collaborators using the group-level permissions.

(2) $Bailey$ begins working on project Pr_1, requiring access to the private computational partition, $\{o_4^{\text{A}}\}$ owned by $Alex$. As such an operation requires updating the scheduler configuration, $Alex$ requests $Erin$ to provide access to $Bailey$.

(3) Subsequently, $Cathy$ requires access to the simulation results produced by $Bailey$, abstracted as $\{o_1^{\text{B}}\}$, to conduct analysis. As $Alex$ does not need access to $\{o_1^{\text{B}}\}$, $Bailey$ selectively shares $\{o_1^{\text{B}}\}$ with $Cathy$ using user-level permissions.

(4) $Alex$ decides to expand the collaboration by inviting a new researcher, $Drew(D)$, to join project Pr_1. $Drew$ is an existing collaborator of $Alex$ from another project Pr_2. $Erin$ adds $Drew$ to the project group grp_1. Since $Drew$ is a temporary collaborator on Pr_1, he does not need access to all the shared resources $\{o_1^{\text{A}}, o_2^{\text{A}}, o_3^{\text{A}}\}$. However, as $Drew$ is added to grp_1, he automatically gains access to all resources through group-level permissions.

(5) Subsequently, $Drew$ also starts helping $Bailey$ and $Cathy$ on simulation and analysis and requires access to relevant resources — the private partition $\{o_4^{\text{A}}\}$ and simulation results $\{o_1^{\text{B}}\}$; Alex shares $\{o_4^{\text{A}}\}$ with Drew with the help of Erin, $Bailey$ shares $\{o_1^{\text{B}}\}$ with $Drew$ using user-level permissions.

(6) $Drew$ leaves project Pr_1, thereby no longer needing access to the shared resources. $Drew$ is removed from grp_1,

which revokes his access to resources shared through group-level permissions.

(7) However, resources $\{o_4^{\mathrm{A}}, o_1^{\mathrm{B}}\}$ were not shared using group-level permissions. Therefore, $Alex$ (with the help of $Erin$) and $Bailey$ must manually revoke $Drew$'s access, one by one. They also need to consider whether $Drew$ still requires access to $Alex$'s resources, as he continues to collaborate with $Alex$ on Pr_2.

(8) As Pr_1 concludes, access to all shared resources must be revoked. However, the default access privileges obtained through ownership must be retained. $Erin$ removes them from grp_1, revoking all group-level privileges. However, the user-level privileges still need to be manually revoked.

Overall, this scenario reveals two fundamental limitations of current practices. First, group memberships and user-level permissions are managed in isolation, with no unified view that ties them together under a project-specific scope, i.e., collaboration context. Second, there is no explicit construct to represent and maintain the evolving collaboration structure over the project's lifetime.

C. Requirements for Collaboration-Aware Resource Sharing

From our system exploration and illustrative workflow analysis (§ III-A), we elicit the following requirements for enabling secure, efficient, and collaboration-aware resource sharing in RCIs. These requirements reflect the importance of explicitly representing and managing collaboration contexts throughout the lifecycle of a project.

(R_1) **Selective Privilege Sharing:** Users should have the flexibility to selectively determine the extent of access privileges they wish to share, both concerning resources (how much to share) and users (whom to share with). For example, instead of uniformly sharing all resources with $Drew$, $Alex$ should have the option to selectively designate a subset $\{o_3^{\mathrm{A}}\}$ from $\{o_1^{\mathrm{A}}, o_2^{\mathrm{A}}, o_3^{\mathrm{A}}\}$, for sharing with $Drew$, while still maintaining the sharing of the entire set of resources $\{o_1^{\mathrm{A}}, o_2^{\mathrm{A}}, o_3^{\mathrm{A}}\}$ with $Bailey$ and $Cathy$.

(R_2) **Selective Revocation of Shared Privileges:** Users should also have the ability to retract previously shared privileges at any time, including the extent of resources shared and/or the involved users. This action should not affect the privileges shared with other collaborators, and should also dynamically adjust such privileges when needed. For example, if $Alex$ chooses to revoke $Drew$'s access to $\{o_3^{\mathrm{A}}\}$, which is also shared with $Bailey$ and $Cathy$, he must ensure that only $Drew$ loses access while the rest retain it.

(R_3) **Project-specific Sharing and Revocation:** All shared privileges should be managed within the context of specific projects, ensuring that resource sharing and revocation are tied to a particular collaborative context. This approach maintains clarity and independence between the same collaborations under different projects. For instance, if $Drew$ leaves project Pr_1 but continues collaborating with $Alex$ on Pr_2, $Drew$ should no longer have access to resources associated with Pr_1. Similarly, revocations within Pr_1 should not affect privileges shared within Pr_2.

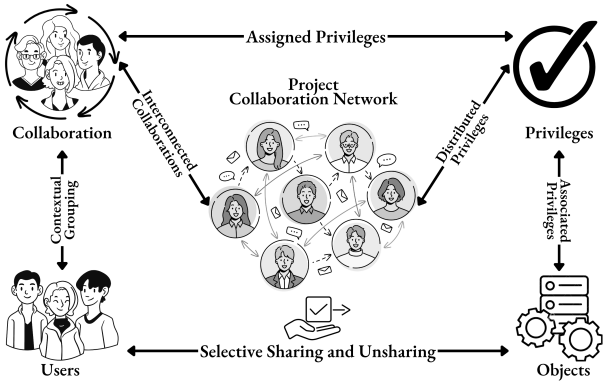

Fig. 2: Components of the CLEARS Framework.

(R_4) **Automatic Revocation of Shared Privileges:** Upon a user's departure from a project, all shared resources with that user should be automatically revoked as manual revocation is both error-prone and lacks scalability. For instance, following $Drew$'s exit from Pr_1, $Alex$, $Bailey$, or $Cathy$ should no longer have the privileges to share resources with $Drew$ within the context of Pr_1. Additionally, once the project ends all privileges should be automatically revoked.

(R_5) **A Uniform Interface for Resource Sharing:** A unified interface should streamline privilege management across various types of resources, by offering a consistent method for assigning and revoking access. This approach is essential not only to reduce administrative burden but also to provide users with a consistent experience.

These requirements are grounded in a set of critical guiding principles that emphasize the importance of formulating collaboration context in RCIs. For instance, requirements (R_1), (R_2), and (R_4) collectively reflect the *principle of least privilege* [35], enabling precise control over resource sharing and timely revocation of access. Requirement (R_3) enforces the management of privileges within explicit project contexts, ensuring clear association between collaborative relationships and granted privileges, promoting *context-aware resource sharing*. Finally, requirement (R_5) underscores the need for *uniform privilege management* across diverse resources, improving both usability and auditability.

IV. Collaboration-aware Resource Sharing

In this section, we propose and formalize the core constructs for representing collaboration context, through the CLEARS framework for enabling secure, collaboration-aware resource sharing within RCIs.

A. Projects, Collaborations, and Privileges

As illustrated in Figure 2, CLEARS is based on the following fundamental entities: Users (U), Objects (O), Privileges (P), Projects (Pr), and Collaborations (C).

Users in this framework refer to any non-administrative entity interacting with the system, that require access to protected resources. **Objects** represent an abstraction of the underlying protected resources. For resource sharing, objects

can either be *shareable* (e.g., a computational partition) or *non-shareable* (e.g., $home$ directories). **Privileges** represent an approval to execute specific operations (e.g., read, write, execute) on the objects within the system. Each privilege, thus, can be represented as a pair (*operation, object*). Traditionally operations mean the access mode that can be exercised on the objects. For instance, ($submit\_jobs$, $cpu\_partition_1$) indicates the privilege to perform the operation $submit\_jobs$ to the object named $cpu\_partition_1$.

Projects are conceptualized as organized efforts involving a *set of users* represented by a dynamic timeline with specific start and end points. Within a project, **Collaborations** are the flexible grouping of the involved users, forming a foundation for resource sharing. The framework allows collaborations to be formed only within the contextual scope of a project, ensuring users are never grouped without a context. It supports collaborations through various combinations of users, with two key bounds: the lower bound is a collaboration involving two users, referred to as *user-to-user (U2U) collaboration*, and the upper bound is a collaboration involving all users within the project, termed as *N-collaboration*. Self-collaboration, i.e., users forming collaborations with themselves is deliberately prevented, as it offers no practical benefit as resources can be accessed through direct ownership (§ IV-C). This flexible definition of collaboration allows users within the same project to belong to multiple collaborations, rather than being limited to a single, all-encompassing *per-project* or *per-faculty* group, as currently practiced (§ III-A).

Shared privileges representing user-to-user resource sharing are also contextualized within the scope of a project. The framework assigns shared privileges to collaborations rather than directly to individual users. This approach allows all users grouped within a collaboration to obtain the assigned privileges, effectively broadening the context from an individual user level to a collaboration level. Simultaneously, by contextually bounding the privileges within a project, the scope of each shared privilege is defined. This enhances auditability and provides a clearer foundation for making access decisions. In this way, all other foundational entities (U, C, and P) are contextualized within projects (Pr).

Definition 1. *The foundational elements of CLEARS are defined as follows:*

- $U = \{u_1, u_2, ..., u_n\}$*, a set of all users in the system;*
- $O = \{o_1, o_2, ..., o_m\}$*, a set of all objects in the system;*
- $OP = \{op_1, op_2, ..., op_k\}$*, a set of operations allowed in the system;*
- $P = \{(op_i, o_j) \mid op_i \in OP \text{ and } o_j \in O\}$*, a set of privileges to perform any valid operation on an object in the system;*
- $C = \{U_k \mid U_k \subseteq U, 2 \leq |U_k| \leq |U|\}$*, a set of all permissible collaborations in the system. Each collaboration* ($c_k \in C$) *is a subset of* U *containing 2 or more users in* U*; and,*
- $Pr = \{(pr_i, U_i, C_i, AP_i) \mid i \in \mathbb{N}\}$*, a set of all projects in the system, where each project is defined by a four-element tuple:*

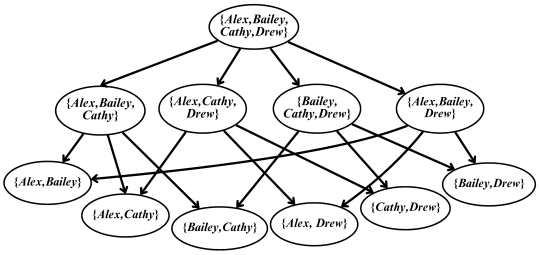

Fig. 3: A sample Collaboration Network (CN) depicting all possible combinations for users Alex, Bailey, Cathy, and Drew, all connected through a Collaboration Hierarchy (CH).

- pr_i*, where* $i \in \mathbb{N}$*, a unique identifier for each project,*
- $U_i \subseteq U$*,* $|U_i| \geq 2$*, a set of users involved in project* pr_i*,*
- $C_i = \{U_j \mid U_j \subseteq U_i, 2 \leq |U_j| \leq |U_i|\}$*, a set of collaborations under a project* pr_i*,*
- $AP_i = \{(c_{ij}, p_k) \mid j \in \mathbb{N}, c_{ij} \in C_i \text{ and } p_k \in P\}$*, a set of assigned privileges shared within project* pr_i

Expanding upon Definition 1, we introduce two additional concepts: **Collaboration Hierarchy** (CH) and **Collaboration Network** (CN) to support administrators in effectively managing shared privileges within collaborations.

CH serves as a means to depict the interconnected relationships among collaborations. Since collaboration represents a *set of users*, CH utilizes the *containment* relationship ($\supseteq$) based on the participating users to relate two collaborations. For instance, if we have a collaboration $\{Alex, Bailey, Cathy\}$, it inherently contains three other collaborations: $\{Alex, Bailey\}$, $\{Bailey, Cathy\}$, and $\{Alex, Cathy\}$. Given that all users involved in these latter collaborations are also part of $\{Alex, Bailey, Cathy\}$, the privileges associated with $\{Alex, Bailey, Cathy\}$ naturally extend to the smaller collaborations.

It is important to note that CH is distinct from constructs such as *Role Hierarchy* in RBAC [23] or the *lattice structure* in MAC [22]. Unlike these models, which encode organizational semantics of authority and responsibility, CH is based solely on a simple containment relationship between user sets.

Following the idea of collaboration hierarchies, for each project, a directed acyclic graph can be constructed by considering all possible collaborations connected through the CH relation. In Figure 3, we illustrate one such graph with four users $Alex$, $Bailey$, $Cathy$, and $Drew$. The directed edges signify the containment relationship between two collaborations, as defined by CH. For example, the directed edges from $\{Alex, Bailey, Cathy\}$ to $\{Alex, Bailey\}$, $\{Bailey, Cathy\}$, and $\{Alex, Cathy\}$ represent that the former contains the latter collaborations and therefore there is a natural flow of privileges from $\{Alex, Bailey, Cathy\}$ to the rest through privilege inheritance. We name such graphs a per-project Collaboration Network (CN). This hierarchical structure allows for a complete representation of collaboration dynamics within the project.

Definition 2. ***Collaboration Contexts.*** *The relevant collabo-*

ration contexts are defined as follows:

- $CH = \{(c_i, c_j) \mid c_i, c_j \in C, c_i \supseteq c_j\}$, *a partial order on* C *called the Collaboration Hierarchy based on the containment relationship,*
- $CN = \{(V, E) \mid V \subseteq C \text{ and } E \subseteq CH\}$, *a set of directed acyclic graphs, known as Collaboration Networks (CNs), where each node is a collaboration, and each edge is an element of the Collaboration Hierarchy,*
- $pcn : Pr \rightarrow CN$, *a function mapping each Project* pr_i *to a Collaboration Network* cn_i, *i.e.,* $pcn(pr_i) = cn_i$, *where,*

$$cn_i = \{C_i, E_i\};$$
$$E_i = \{(c_x, c_y) \mid c_x, c_y \in C_i, (c_x, c_y) \in CH\}$$

From the perspective of shared privilege management, such per-project networks aided with assigned privileges (AP_i) provide a complete snapshot of all shared privileges within the context of a project. For instance, referring back to the example in Figure 1, consider when $Alex$ shares resources $\{o_1^{\text{A}}, o_2^{\text{A}}, o_3^{\text{A}}\}$ with $Bailey$ and $Cathy$ while sharing resources $\{o_4^{\text{A}}\}$ with $Bailey$ alone. Assigning privileges to access $\{o_1^{\text{A}}, o_2^{\text{A}}, o_3^{\text{A}}\}$ at the collaboration $\{Alex, Bailey, Cathy\}$, and $\{o_4^{\text{A}}\}$ at $\{Alex, Bailey\}$ can accurately represent the shared privileges and effectively grant $Bailey$ and $Cathy$ to access $\{o_1^{\text{A}}, o_2^{\text{A}}, o_3^{\text{A}}\}$, while selectively allowing $Bailey$ to access $\{o_4^{\text{A}}\}$ restricting $Cathy$'s access. This way, the shared privileges within a project can be distributed across multiple collaborations instead of being exclusively shared within the N-collaboration context, as currently practiced.

B. CLEARS Functions

CLEARS is designed to help administrators efficiently manage shared privileges in collaborative projects, while also supporting resource owners in selectively sharing and revoking privileges. It provides both administrators and users with functions tailored to their respective needs.

Based on this understanding, the supported functions are categorized into two intuitive groups: *admin-only* and *owner-centric* functions. *Admin-only* functions require elevated privileges and are only executed by admins. In contrast, *owner-centric* actions directly modify privileges and, therefore, require further authorization.

'Admin-only' functions. Here, we discuss the administrative functions supported by CLEARS: **(1) create_project**: Creating a new project initializes an empty collaboration network associated with the project; **(2) end_project**: Terminating an existing project, removes all relevant user-project associations and the associated network and all shared privileges; **(3) add_collaborator**: This function associates users with an existing project, updating the project definition; **(4) remove_collaborator**: Users can also be removed from a project, which involves updating the project definition. This action also revokes any privileges, leading to a dynamic shrinkage in the collaboration network, as all collaborations involving the removed collaborator are eliminated.

For employing these functions, CLEARS requires only elevated privileges, leaving all authorization decisions to administrators. They may, for example, pre-authorize users to manage projects or collaborators, or restrict ownership and resource creation. Importantly, CLEARS itself remains agnostic to such administrative policies.

'Owner-centric' functions. As outlined in § III-C, a key objective of CLEARS is to support resource owners to selectively determine the scope of access privileges they intend to share, both in terms of resources (how much to share) and users (whom to share with). Equally important is the ability to retract previously shared privileges. Therefore, the following are the two owner-centric actions supported, in addition to traditional actions such as *create* and *delete* resources: **(1) share_privilege**: Resource owners can share privileges to access certain resources with others by meeting certain preconditions defined in *can_share* (§ IV-C); **(2) unshare_privilege**: Resource owners can revoke previously shared privileges with others. These actions might trigger privilege expansion and contraction, respectively, further discussed in § IV-D.

C. Authorization of Resource Sharing

It is crucial to recognize that executing the owner-centric functions (§ IV-B) directly alter the assigned privileges within project collaborations, dynamically influencing what objects a user can access. Therefore, imposing restrictions on these actions becomes imperative. By enforcing preconditions for each sharing and unsharing action, concerning both resources (which objects are involved) and users (who are the participants), CLEARS ensures that such actions do not result in unauthorized access.

Accordingly, CLEARS imposes constraints on these actions based on the association between the user performing the action, the involved objects, and the recipients of the shared privileges. For instance, the following constraints are enforced for sharing privileges: **(1)** Regarding the resources involved, the pre-condition is that the user wanting to share this resource must be the resource owner. This ownership constraint prevents unauthorized sharing of privileges acquired from other users (i.e., *multi-step sharing*), thereby mitigating the risk of resource misuse across multiple project contexts and ensuring better accountability of shared privileges; **(2)** Concerning the recipients of the shared privileges, users are authorized to share resources only with collaborators within a project. Any attempt to share resources with non-collaborators is restricted.

Before formally defining the constraints above, we augment the framework with the newly addressed concept of ownership.

Definition 3. ***Ownership.*** *Ownership is supported as follows:*

- $owner : O \rightarrow U$, *a function mapping each object (or Resource) to a single user, i.e.,* $owner(o_i) = u_j$

Definition 4. ***Constraints in Sharing.*** *The following relation authorizes the action share_privilege for user-to-user resource sharing, capturing the above-mentioned constraints:*

- *A user* u_i *can share the privilege* $p_k = (op, o_k)$ *that authorizes access to the object* o_k *to another user* u_j, *if* $(u_i, p_k, u_j) \in can\_share$, *where,*

$$owner(o_k) = u_i, \text{ and}$$
$$\exists_{pr_m \in Pr} : u_i, u_j \in U_m \text{ and } \{u_i, u_j\} \in C_m$$

The notation $(u_i, p_k, u_j) \in can\_share$ signifies that user u_i can perform $share\_privilege$ p_k associated with the object o_k with another user u_j, if (1) u_i is the owner of the concerned object o_k, and (2) there exists at least one project pr_m, of which both u_i and u_j are part of, and the collaboration between these two designated as $\{u_i, u_j\}$ is related with pr_m. The same constraints hold for *unshare_privilege* as well.

D. Dynamic Management of Privileges

As discussed in § IV-A, since every collaboration is contained within its parent collaborations, it can naturally inherit shared privileges from the parent collaborations. Therefore, an effective approach to managing shared privileges is to allocate privileges shared with more users closer to the *N-collaboration*, while reserving more specialized access privileges for *U2U* nodes. This allows each shared privilege to be assigned to a single context while meeting the sharing requirements. For example, in Figure 1, when $Alex$ initially shares $\{o_4^{\text{A}}\}$ with $Bailey$ and later extends it to $Drew$, duplicating $\{o_4^{\text{A}}\}$ within both $\{Alex, Bailey\}$ and $\{Alex, Drew\}$ is unnecessary. Instead, we can consolidate this common privilege at a broader collaboration, $\{Alex, Bailey, Drew\}$. This eliminates the need to maintain identical privileges across multiple collaborations.

This requires the CN to dynamically capture each modification of privilege assignment resulting from the *share_privilege* and *unshare_privilege* actions. Moreover, as the network grows (or shrinks) with the addition (or removal) of collaborators, privileges must be dynamically redistributed to ensure that assigned privileges accurately reflect the actions taken so far. This is facilitated by two phenomena: *Privilege Expansion* and *Privilege Contraction*.

Privilege Expansion. It occurs when the scope of a privilege is expanded to encompass a broader collaboration context in the CN, involving more users, closer to the *N-collaboration*. This becomes evident when a privilege is shared within a new collaboration, whereas it was already shared within an existing collaboration. The expanded collaboration now includes both the existing collaborators and the new collaborators with whom the privilege was shared. This serves two primary purposes: (1) to prevent duplication of the same privilege across multiple collaborations within the same network, and (2) to ensure that the privilege always resides in the broadest context possible including those who are authorized to have access, maximizing its accessibility within the network.

Mathematically, if a privilege p_i is shared within a new collaboration context c_j within the project pr_m, the elevated context c' can be obtained as follows:

$$c' = \begin{cases} c_j \cup c_k, & \text{if } \exists_{c_k} : c_k \in C_m, (c_k, p_i) \in AP_m \\ c_j, & \text{otherwise, given } c_j \in C_m \end{cases}$$

Therefore, the privilege p_i will be assigned to the expanded collaboration c' and removed from the past collaboration c_k, if it exists within pr_m. For instance, referring to Figure 4, if $Alex$ wanted to share only $\{o_3^A\}$ with $Drew$, initially shared with both $Bailey$ and $Cathy$ within the collaboration $\{Alex, Bailey, Cathy\}$, through privilege expansion, scope of $\{o_3^A\}$ is expanded to the N-collaboration, denoted as, $\{Alex, Bailey, Cathy, Drew\}$.

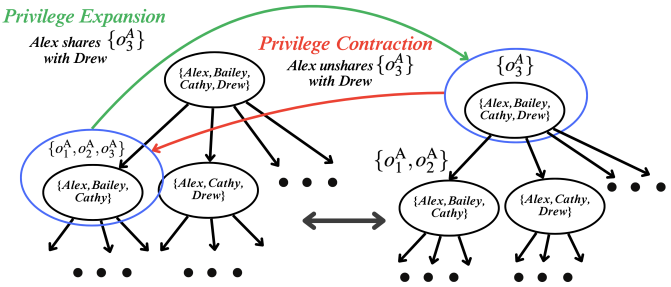

Fig. 4: Dynamic Management of Privileges within a CN through Privilege Expansion and Privilege Contraction.

Privilege Contraction. It occurs when the scope of a privilege is moved from a broader collaboration context to a more specialized context, closer to the *U2U collaboration* level. This happens when a shared privilege is partially revoked from a collaboration within a project, while still retaining it with some users in that collaboration. The contracted collaboration now comprises only the remaining collaborators who still need the privilege, along with the owner. This serves two primary purposes: (1) to ensure that users from whom the privilege has been revoked no longer possess it, and (2) to maintain the sharing status with other collaborators who still have access to that privilege.

Mathematically, if a privilege $p_i = (op, o_i)$ is to be revoked from a collaboration context c_j within the project pr_m, the contracted context c' can be obtained as follows:

$$c' = \begin{cases} (c_k - c_j) \cup owner(o_i), \\ \quad \text{if } \exists_{c_k} : c_k \in C_m, (c_k, p_i) \in AP_m, c_k \neq c_j \\ \phi, \quad \text{if } c_k = c_j \end{cases}$$

Therefore, if the privilege p_i needs to be revoked from a collaboration c_j, and there exists another collaboration c_k within pr_m, where p_i is already shared, the privilege will be removed from c_k and assigned to the contracted context c'. This c' is formed by taking a *set difference* of c_k and c_j and explicitly including the $owner(o_i)$, as the owner is a member of both c_j and c_k collaborations. Consequently, if c_k is the same as c_j, the privilege p_i is simply removed since the contracted context is empty (represented by ϕ).

For instance, consider the scenario (Figure 4) where $Alex$ decides to unshare $\{o_3^A\}$ from $\{Alex, Drew\}$ to selectively revoke $Drew$'s access. Since $\{o_3^A\}$ is currently shared within the *N-collaboration*, in this case, $\{o_3^A\}$ moves from the *N-collaboration* to the collaboration $\{Alex, Bailey, Cathy\}$, as $Bailey$ and $Cathy$ still retain access to $\{o_3^A\}$, but $Drew$ no longer has access to it. This ensures that the sharing status with other collaborators, if any, remains unchanged while preventing $Drew$ from retaining access to the privilege.

E. Integration with Existing Models

It is important to note that, the core contribution of CLEARS lies in formulating the notion of collaboration context, which can be consumed by existing access control models (§ II) to make more precise and context-aware authorization decisions.

For instance, CLEARS can complement traditional ACL-based DAC mechanisms by representing each collaboration context as a system group (or ACL principal). In this setup, privilege expansion and contraction are realized through corresponding group membership and ACL updates. By integrating CLEARS's constructs into this process, privileges can be managed with greater precision and contextual awareness (§ III-C). This integration is further illustrated in our prototype implementation [14]. Similarly, CLEARS can potentially extend RBAC by enabling collaboration-scoped roles, that are instantiated and bounded to a specific collaborative context. This approach ties role assignments and their permissions directly to the dynamic nature and structure of an active collaboration context, avoiding global, coarse-grained roles and promoting least-privilege enforcement at the project level. Moreover, the lifecycle events emitted by privilege expansions and contractions can further support Administrative RBAC [36], governing the assignment and revocation of role memberships as collaboration structures evolve, reducing the risk of lingering privileges when a collaboration or a project ends. In addition, CLEARS can be utilized to lay out and enforce context-related constraints within RBAC settings.

While this work introduces and formalizes the core constructs of CLEARS, exploring its systematic integration with established access control frameworks remains a promising direction for future research (§ VI).

V. Evaluation

In this section, we evaluate CLEARS against the following two key questions:

(Correctness) *How effectively does CLEARS's collaboration-context representation support correct access decisions in dynamic resource sharing scenarios?*

(***Performance***) *How much execution overhead CLEARS incurs when handling resource sharing under realistic workload?*

A. Correctness Evaluation

To evaluate the *correctness* of CLEARS, we conduct a scenario-based small-scale case study that simulates a realistic resource-sharing scenario, allowing us to analyze access decisions under dynamic privilege assignments and revocations throughout the project lifecycle.

Case Study Design. The scenario is an extended version of the one introduced in § III-C. It involves six users ($Alex$, $Alice$, Bob, $Connor$, $Drew$, and $Dave$) participating in two overlapping research projects: *ProjectX* (involving $Alice$, Bob, $Connor$, and $Dave$) and *ProjectY* (involving $Alex$, $Alice$, Bob, and $Drew$). Each user has a personal directory in $/scratch/$, while $Alice$, $Alex$, and Bob also own directories in $/data/$. Additionally, Alice owns a private partition named $alice\_partition1$, resulting in a total of ten resources.

t	No. of 'Permit' Decisions (out of 60 requests)				
	G	**U**	**R**	**CLEARS**	**GT**
$t=0$	10	10	10	10	10
$t=1$	$15_{(+5)}{}^{\checkmark}$	$15_{(+5)}{}^{\checkmark}$	$15_{(+5)}{}^{\checkmark}$	$15_{(+5)}{}^{\checkmark}$	$15_{(+5)}$
$t=2$	$18_{(+3)}{}^{\dagger}$	$17_{(+2)}{}^{\checkmark}$	$17_{(+2)}{}^{\checkmark}$	$17_{(+2)}{}^{\checkmark}$	$17_{(+2)}$
$t=3$	$21_{(+3)}{}^{\dagger}$	$19_{(+2)}{}^{\checkmark}$	$19_{(+2)}{}^{\checkmark}$	$19_{(+2)}{}^{\checkmark}$	$19_{(+2)}$
$t=4$	$24_{(+3)}{}^{\dagger}$	$20_{(+1)}{}^{\checkmark}$	$20_{(+1)}{}^{\checkmark}$	$20_{(+1)}{}^{\checkmark}$	$20_{(+1)}$
$t=5$	$27_{(+3)}{}^{\dagger}$	$21_{(+1)}{}^{\checkmark}$	$21_{(+1)}{}^{\checkmark}$	$21_{(+1)}{}^{\checkmark}$	$21_{(+1)}$
$t=6$	$25_{(-2)}{}^{\checkmark}$	$19_{(-2)}{}^{\checkmark}$	$19_{(-2)}{}^{\checkmark}$	$19_{(-2)}{}^{\checkmark}$	$19_{(-2)}$
$t=7$	$22_{(-3)}{}^{\dagger}$	$19_{(-0)}{}^{\dagger}$	$17_{(-2)}{}^{\checkmark}$	$17_{(-2)}{}^{\checkmark}$	$17_{(-2)}$
$t=8$	$19_{(-3)}{}^{\dagger}$	$18_{(-1)}{}^{\dagger}$	$16_{(-1)}{}^{\checkmark}$	$16_{(-1)}{}^{\checkmark}$	$16_{(-1)}$
$t=9$	$18_{(-1)}{}^{\dagger\#}$	$18_{(-0)}{}^{\dagger\#}$	$16_{(-0)}{}^{\dagger\#}$	$14_{(-2)}{}^{\checkmark\#}$	$14_{(-2)}$
$t=10$	$10_{(-8)}{}^{\checkmark\#}$	$15_{(-3)}{}^{\dagger\#}$	$16_{(-0)}{}^{\dagger\#}$	$10_{(-4)}{}^{\checkmark\#}$	$10_{(-4)}$

$t=0$: ProjectX, ProjectY start, and Users are added to projects
$t=1$: Alice shares $/scratch/alice$ with ALL in both projects
$t=2$: Alice shares $/data/alice$ with Bob, Connor in ProjectX
$t=3$: Dave shares $/scratch/dave$ with Alice, Connor in ProjectX
$t=4$: Alice shares $alice\_partition1$ with Alex in ProjectY
$t=5$: Bob shares $/data/bob$ with only Alex in ProjectY
$t=6$: Alice unshares $/scratch/alice$ with ALL in ProjectX
$t=7$: Alice unshares $/scratch/alice$ with Alex, Drew in ProjectY
$t=8$: Alice unshares $alice\_partition1$ with Alex in ProjectY
$t=9$: Dave leaves ProjectX
$t=10$: ProjectX and ProjectY end and Users are removed

$^{\checkmark}$ Matches with the ground truths at each consequent step (green).
† Mismatches with the ground truths at each consequent step (red).
$^{\#}$ Manual revocation of privileges is not assumed.

TABLE I: Correctness Comparison based on the number of 'Permit' decisions across scenario timestamps.

Throughout the scenario, we simulate different project events such as resource sharing and revocation, collaborators joining and leaving the project, and project completion, over 11 timestamps as illustrated in Table I. At each timestamp, six users make access requests to all ten resources, yielding 60 total access requests.

Understanding Existing Approaches in RCI. During our exploration (§ III-A), we observed hybrid DAC mechanisms involving a combination of *user* and *group-level permissions*. Privileges for project-specific resources are managed uniformly at the *group level*, while in others, resource owners grant access on an as-needed basis to *individual users* without any direct association with a project. These approaches are managed in isolation, with no representation of collaboration contexts.

To understand the role of collaboration context in resource sharing, we explore three observed practices in RCIs, none of which explicitly represent collaboration context. They differ primarily in how privileges are assigned by resource owners—whether at the *group* level, *user* level, or *role* level.

- **Group-only (G):** All shared privileges are assigned at the project group level with no additional support for user-to-user resource sharing [37].
- **User-centric (U):** This is a hybrid approach where only the privileges that need to be shared with all collaborators are assigned at the group level. All other privileges are shared directly on the user-to-user level [38].
- **Role-based (R):** A role-based discretionary approach [39]

where each object is associated with an *owner* role and one or more *access* roles. Users assigned to the *owner* role can grant or revoke access for others by managing their membership in *access* roles, controlling access to the corresponding object.

Comparative Analysis and Insights. As shown in Table I, the *correctness* of each approach is measured by counting the number of 'permit' decisions out of 60 possible access requests at each timestamp. The subscripted numbers indicate changes in permits compared to the previous step, illustrating the effect of each event/action on access decisions.

The number of 'permit' decisions serves as an abstract yet effective metric for measuring the correctness of the approaches by capturing the deviation from the theoretical ground truth in making access decisions. A positive or negative deviation means the model is too permissive or too restrictive, respectively, with an exact match being ideal. The ground truth for each timestamp is derived by accounting for the intended access privileges throughout the project's lifecycle.

Initially, at timestamp $t = 0$, only the owners have access to their resources, resulting in 10 permit decisions (one per resource). As the project advances, these initial privileges are modified based on specific sharing and un-sharing actions. This process continues dynamically until the project's completion, at which point all shared privileges are revoked, ensuring that only the original 10 privileges remain at timestamp $t = 10$. Table I indicates the correct access decisions (✓) matching the ground truth, and any unmatched decisions (†). As observed, the access decisions evaluated by CLEARS match exactly with the ground truth, supporting its correctness. Upon further analysis, we observe the following:

The **Group-only approach (G)** configures privileges in a manner that is either too permissive or too restrictive, resulting in unintended 'permit' or 'deny' decisions. For instance, at $t = 2$, *G* assigns the shared privilege directly to the ProjectX group, thereby granting Dave excessive access as the privilege cannot be selectively assigned only to Bob and Connor. Similarly, at $t = 7$, access is revoked for all involved, including Bob, which was not intended.

CLEARS addresses this by representing all possible collaboration contexts within a project-specific CN, enabling more flexible and selective sharing and revocation of privileges.

The **User-centric approach (U)**, while effective in flexible resource sharing by allowing a user-to-user approach, fails to maintain project or collaboration context. This leads to difficulties in revoking access ($t = 7; 8$).

CLEARS addresses this by associating each privilege with a unique project-collaboration context within a per-project CN, improving visibility and management.

The **Role-based approach (R)**, while solving the issue of flexible and selective revocation, the lack of context-awareness makes it difficult to support the automatic revocation of shared privileges. For example, at $t = 9$, when Dave leaves the project, it is essential to ensure that all privileges shared with Dave are revoked and that any privileges Dave shared with others are also revoked. However, in the current setup, such revocations must be performed manually, which can result in unretracted privileges.

CLEARS supports automated privilege revocation by removing the contexts involving the removed collaborator from the CN, ensuring that no shared privileges remain unrevoked after such events.

Requirements	Sub-reqs.	G	U	R	CLEARS
Selective Sharing	with Individual	○	●	●	●
	with All	●	●	●	●
	with Subset	○	●	●	●
Selective Revocation	with Individual	○	◐	●	●
	with All	●	●	●	●
	with Subset	○	◐	●	●
Project-specific Sharing	-	●	○	○	●
Automatic Revocation	with Departure	◐	○	○	●
	with Completion	●	○	○	●

● Fully Supported, ◐ Partially Supported, ○ Not Supported.

TABLE II: Requirement-level Comparison (§ III-C) based on findings from the scenario-based case study.

We summarize the outcome of this comparative analysis in Table II, depicting the level of support each approach provides across critical access control requirements identified in § III-C. Each requirement is marked as fully (●), partially (◐), or not (○) supported based on the model's behavior observed in the case study. As illustrated, CLEARS is the only model that fully supports all key requirements, including selective, project-specific privilege sharing and revocation, and automatic revocation upon departure or project completion.

B. Performance Evaluation

While the correctness analysis highlights the benefits of embedding privileges in a collaboration context, it is also important to assess the overhead of maintaining and updating this context under realistic workloads. To this end, we conduct a detailed experimental evaluation focused on measuring the *execution latency* during *share* and *unshare* operations in dynamic collaboration settings. For this evaluation, we developed a command-line interface (CLI) tool, `clears`. Implementation details can be found in our code repository [14].

Evaluation Procedure. To evaluate the execution latency overhead introduced by CLEARS, we simulate a dynamic project environment with a randomized workload that evolves over 100 discrete timestamps. The project begins with 20 collaborators and gradually scales up to a maximum of 100 users with 100 resources. At each timestamp, a randomized set of operations is performed to emulate realistic project dynamics. These operations include user additions and removals, as well as privilege *share* and *unshare* actions.

For example, at any given timestamp, a subset of users can initiate *share* (or *unshare*) operations, with recipients randomly selected to reflect varying scopes of collaboration, ranging from a single individual to all active collaborators. This randomized selection models diverse real-world sharing patterns as outlined in Table II. By incorporating such vari-

Metric	Action	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	Mean
Minimum Latency	Share	197	191	190	174	185	181	137	174	49	178	**166**
	Unshare	150	149	157	139	143	144	372	142	50	142	**159**
Maximum Latency	Share	1157	977	738	997	802	274	678	532	346	610	**711**
	Unshare	949	1265	723	934	630	835	542	467	522	1055	**792**
Mean Latency	Share	343	362	345	402	312	322	306	300	312	304	**331**
	Unshare	332	343	324	332	281	287	272	268	285	270	**299**

TABLE III: Execution Latency of CLEARS for Each Share and Unshare Operation under a Randomized Workload in a Project with up to 100 Users, and 100 Resources Measured Across 10 Experimental Runs (in milliseconds).

ability, the workload closely approximates realistic resource-sharing behaviors observed within RCIs.

Furthermore, to better reflect the temporal evolution of collaborative projects, the simulation is divided into three distinct phases:

- *Ramp-Up* ($t = 0$–39): The project experiences growth, with more collaborators joining and a high volume of privilege-sharing operations.
- *Steady-State* ($t = 40 - 59$): The project stabilizes at its peak size, maintaining approximately 100 active users. The rate of user additions and removals remains low, and while sharing continues, its frequency begins to decline.
- *Wind-Down* ($t = 60$–99): The project gradually declines in activity. Collaborators begin to leave, and privilege revocations (*unshare*) become more frequent than sharing.

To model the behavioral shifts across the different project phases, the probabilities of operation types are dynamically adjusted over time. For example, during the *ramp-up* phase, the probability ratio of *share* to *unshare* operations is initialized at $70:30$, emphasizing the predominance of privilege assignment over revocation. Moreover, to simulate a gradual transition from sharing-dominant to revocation-dominant behavior, this ratio is progressively adjusted over time. Specifically, the probabilities are linearly interpolated from $70:30$ to $30:70$ across successive 10-timestamp intervals, ensuring a smooth behavioral shift that mirrors a natural and intuitive evolution of collaborative workflows.

Each *share* and *unshare* operation executed at every timestamp is timed to capture the execution latency. We repeated the entire experiment 10 times using different random seeds to ensure statistical robustness and minimize the influence of workload-specific variations. The experiments were conducted on a VirtualBox VM running Ubuntu 22.04.3 LTS Server, with 8 CPU cores and 8GB RAM allocated. Python 3.10 was used for code execution and timing measurements.

Evaluation Results. Table III presents the minimum, maximum, and average execution latencies for *share* and *unshare* operations, aggregated over 10 experimental runs. On average, both operations exhibit a latency of approximately 300 milliseconds. Latency peaks under high-user-count conditions. This increase is expected, as the underlying collaboration network is more densely connected at this stage, requiring additional traversal to accurately identify and update the relevant collaboration context. Nonetheless, the overall low average latency (consistently below 400 milliseconds) indicates that these peak values are infrequent.

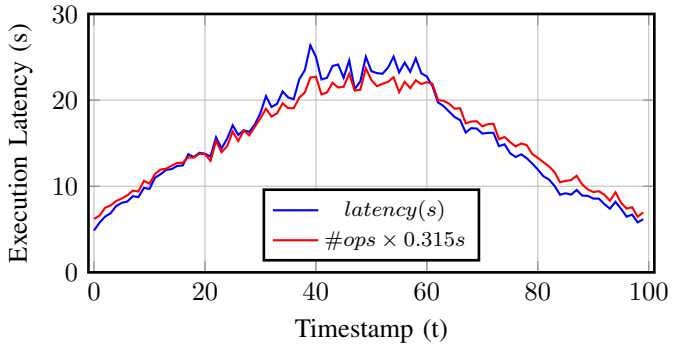

Fig. 5: Total Execution Latency of CLEARS (Blue) and Number of Operations (Red, Scaled) Across Project Timeline (Timestamps 1–200), Averaged Over 10 Experimental Runs.

Furthermore, to evaluate the scalability of CLEARS, we plotted the total execution latency (in seconds) in each timestamp alongside the number of operations (scaled by the average latency of 0.315 seconds), as shown in Figure 5. The close alignment between the two curves confirms a linear scalability pattern—execution latency increases proportionally with the number of operations, rather than exponentially.

Overall, these results suggest that maintaining an explicit collaboration context, can be achieved with low and predictable overhead, well within acceptable limits for real-world usage. This makes it feasible to integrate CLEARS's constructs into existing access control frameworks such as RBAC or ABAC, enhancing them with collaboration-aware privilege management without replacing their core mechanisms.

VI. Conclusion and Future Work

In this work, we systematically investigated current access control and resource sharing practices within RCIs, identified key requirements, and introduced CLEARS as a systemic framework for formalizing and managing collaboration contexts to support secure and flexible resource sharing. By assigning privileges to well-defined collaboration contexts, CLEARS provides a structured basis for consistent, context-aware authorization decisions.

Future work includes extending CLEARS beyond a single RCI environment by incorporating a multi-institutional perspective, as highlighted in prior work [13], to refine requirements under diverse regulatory and collaborative settings. We

also plan to investigate CLEARS's operational concerns under real-world deployment, such as race conditions, atomicity of share/unshare operations, conflict resolution when multiple actors intervene concurrently, and the security of system-level mechanisms (e.g., JSON storage, setuid root helper). Additionally, we aim to conduct quantitative analyses of CLEARS' worst-case complexity, including dense collaboration networks and overlapping projects, and to design efficient data structures and algorithms for scalable deployment in real RCI environments. Finally, incorporating feedback from potential stakeholders through a user-validation study will be critical to assessing the framework's usability and acceptance.

Acknowledgment

We acknowledge the support from the Research Computing Core Facilities at Arizona State University for providing access to the computing and storage resources that have contributed to the research results reported in this paper. This work was partly supported by the National Science Foundation (NSF) under grant NSF-CICI-2232911 and the Institute of Information & Communications Technology Planning & Evaluation (IITP) through the following grants: RS-2024-004398199 and RS-2024-00442085.

References

[1] C. A. Stewart, S. Simms, B. Plale, M. Link, D. Y. Hancock, and G. C. Fox, "What is cyberinfrastructure," in *Proc. of the 38th annual ACM SIGUCCS fall conference: navigation and discovery*, pp. 37–44, 2010.

[2] "Research Computing University of Colorado." https://www.colorado.edu/research/report/2021-22/research-computing-top-tier-research-universitys-unsung-zeros-and-ones.

[3] "Research Computing Infrastructure (RCI)—Ohio State College of Medicine." https://medicine.osu.edu/research/research-information-technology/services-and-products/research-computing-infrastructure.

[4] "About RC—Arizona State University." https://cores.research.asu.edu/research-computing/about.

[5] "Research Computing Infrastructure—University at Buffalo." https://www.buffalo.edu/ccr/services/high-performance-computing.html.

[6] L. D. Stein, "Towards a cyberinfrastructure for the biological sciences: progress, visions and challenges," *Nature Reviews Genetics*, vol. 9, no. 9, pp. 678–688, 2008.

[7] E. Strubell, A. Ganesh, and A. McCallum, "Energy and policy considerations for modern deep learning research," in *Proceedings of the AAAI conference on artificial intelligence*, pp. 13693–13696, 2020.

[8] P. Muir, S. Li, S. Lou, D. Wang, D. J. Spakowicz, L. Salichos, J. Zhang, G. M. Weinstock, F. Isaacs, J. Rozowsky, *et al.*, "The real cost of sequencing: scaling computation to keep pace with data generation," *Genome biology*, vol. 17, pp. 1–9, 2016.

[9] Z. D. Stephens, S. Y. Lee, F. Faghri, R. H. Campbell, C. Zhai, M. J. Efron, R. Iyer, M. C. Schatz, S. Sinha, and G. E. Robinson, "Big data: astronomical or genomical?," *PLoS biology*, vol. 13, no. 7, 2015.

[10] National Science Foundation (NSF), "Cybersecurity innovation for cyberinfrastructure (cici)." https://new.nsf.gov/funding/opportunities/cybersecurity-innovation-cyberinfrastructure-cici, 2024.

[11] J. Reed, "White house mandates stricter cybersecurity for R&D institutions." https://securityintelligence.com/news/white-house-mandates-stricter-cybersecurity-for-research-and-development/, 2024.

[12] M. Callahan, "BU and Federal Investigation Underway into Hacking of Framingham Heart Study Data." https://www.bu.edu/articles/2024/investigation-into-hacking-of-fhs-data/, 2024.

[13] S. Nath, A. Soneji, J. Baek, T. Bao, A. Doupé, C. Rubio-Medrano, and G.-J. Ahn, ""It's almost like Frankenstein": Investigating the Complexities of Scientific Collaboration and Privilege Management within Research Computing Infrastructures," in *IEEE Symposium on Security and Privacy (SP)*, pp. 2995–3013, IEEE Computer Society, 2025.

[14] S. Nath, A. Soneji, J. Baek, C. Rubio-Medrano, and G.-J. Ahn, "CLEARS Code Repository." https://github.com/confusedDip/CLEARS.

[15] S. Sivagnanam, S. Yeu, K. Lin, S. Sakai, F. Garzon, K. Yoshimoto, K. Prantzalos, D. Upadhyaya, A. Majumdar, and S. S. Sahoo, "Towards building a trustworthy pipeline integrating neuroscience gateway and open science chain," *Database*, vol. 2024, p. baae023, 2024.

[16] M. J. H. Faruk, J. Basney, and J. Q. Cheng, "Leveraging smart contracts for privacy-preserving authentication mechanisms to enhance data security in scientific data access," in *2023 IEEE Int. Conf. on Big Data (BigData)*, pp. 5493–5502, 2023.

[17] M. J. Hossain Faruk, B. Saha, and J. Basney, "A comparative analysis between scitokens, verifiable credentials, and smart contracts: Novel approaches for authentication and secure access to scientific data," in *Practice and Experience in Advanced Research Computing*, 2023.

[18] Z. Anwar, R. Shankesi, and R. H. Campbell, "Automatic security assessment of critical cyber-infrastructures," in *2008 IEEE Int. Conf. on Dependable Systems and Networks With FTCS and DCC (DSN)*, pp. 366–375, IEEE, 2008.

[19] F. Enayaty-Ahangar, L. A. Albert, and E. DuBois, "A survey of optimization models and methods for cyberinfrastructure security," *IISE Transactions*, vol. 53, no. 2, pp. 182–198, 2020.

[20] A. R. Mathew, "Cyber-infrastructure connections and smart grid security," *Int. Journal of Engineering and Advanced Technology*, vol. 8, no. 6, pp. 2285–2287, 2019.

[21] "Why is linux filesystem considered dac and not mac." https://security.stackexchange.com/questions/199153/why-is-linux-filesystem-considered-dac-and-not-mac, 2020.

[22] R. Sandhu, "Lattice-based access control models," *Computer*, vol. 26, no. 11, pp. 9–19, 1993.

[23] R. S. Sandhu, "Role-based access control," in *Advances in computers*, vol. 46, pp. 237–286, Elsevier, 1998.

[24] T. Mawla, M. Gupta, and R. Sandhu, "Bluesky: Activity control: A vision for active security models for smart collaborative systems," in *Proc. of the 27th ACM symp. on access control models and technologies*, pp. 207–216, 2022.

[25] L. Zhang, G.-J. Ahn, and B.-T. Chu, "A rule-based framework for role-based delegation and revocation," *ACM Transactions on Information and System Security (TISSEC)*, vol. 6, no. 3, pp. 404–441, 2003.

[26] R. K. Thomas, "Team-based access control (tmac) a primitive for applying role-based access controls in collaborative environments," in *Proceedings of the second ACM workshop on Role-based access control*, pp. 13–19, 1997.

[27] S. Damen, J. den Hartog, and N. Zannone, "Collac: Collaborative access control," in *2014 International Conference on Collaboration Technologies and Systems (CTS)*, pp. 142–149, IEEE, 2014.

[28] P. W. Fong, "Relationship-based access control: protection model and policy language," in *Proceedings of the first ACM conference on Data and application security and privacy*, pp. 191–202, 2011.

[29] X. Jin, R. Krishnan, and R. Sandhu, "A unified attribute-based access control model covering dac, mac and rbac," in *Proc. of the Conf. on Data and Applications Security and Privacy*, pp. 41–55, Springer, 2012.

[30] M. Gupta and R. Sandhu, "Towards activity-centric access control for smart collaborative ecosystems," in *Proceedings of the ACM symposium on access control models and technologies*, pp. 155–164, 2021.

[31] "Research Computing—University of Florida." https://www.rc.ufl.edu/documentation/procedures/groups-and-allocations/.

[32] "Savio system overview." https://docs-research-it.berkeley.edu/services/high-performance-computing/overview/system-overview/.

[33] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Workshop on job scheduling strategies for parallel processing*, pp. 44–60, Springer, 2003.

[34] V. Samar, "Unified login with pluggable authentication modules (pam)," in *Proc. of the 3rd ACM Conf. on Comp. and Comm. Security*, 1996.

[35] F. B. Schneider, "Least privilege," *IEEE Security & Privacy*, vol. 1, no. 5, pp. 55–59, 2003.

[36] R. Sandhu, V. Bhamidipati, and Q. Munawer, "The arbac97 model for role-based administration of roles," *ACM Transactions on Information and System Security (TISSEC)*, vol. 2, no. 1, pp. 105–135, 1999.

[37] Department of Computer Science Computing Guide, "User groups/unix groups." https://csguide.cs.princeton.edu/account/groups, 2024.

[38] M. Hausenblas, *Learning Modern Linux*. " O'Reilly Media, Inc.", 2022.

[39] R. Sandhu and Q. Munawer, "How to do discretionary access control using roles," in *Proc. of the third ACM workshop on Role-based access control*, pp. 47–54, 1998.