Łukasz Opioła
Kamil Jarosz
Łukasz Dutka
Renata G. Słota
Jacek Kitowski

GROUP MEMBERSHIP MANAGEMENT FRAMEWORK FOR DECENTRALIZED COLLABORATIVE SYSTEMS

Abstract *Scientific and commercial endeavors could benefit from cross-organizational, decentralized collaboration, which becomes the key to innovation. This work addresses one of its challenges, namely efficient access control to assets for distributed data processing among autonomous data centers. We propose a group membership management framework dedicated for realizing access control in decentralized environments. Its novelty lies in a synergy of two concepts: a decentralized knowledge base and an incremental indexing scheme, both assuming a P2P architecture, where each peer retains autonomy and has full control over the choice of peers it cooperates with. The extent of exchanged information is reduced to the minimum required for user collaboration and assumes limited trust between peers. The indexing scheme is optimized for read-intensive scenarios by offering fast queries – look-ups in precomputed indices. The index precomputation increases the complexity of update operations, but their performance is arguably sufficient for large organizations, as shown by conducted tests. We believe that our framework is a major contribution towards decentralized, cross-organizational collaboration.*

Keywords group membership management, decentralized system, decentralized collaboration, coordination middleware

Citation Computer Science 23(4) 2022: 521–544

1. Introduction

Computer-supported collaboration is constantly evolving, trying to utilize the powerful computing infrastructures that are becoming more and more accessible. One of the major ongoing challenges is the ability to combine the capabilities of multiple data centers and the expertise of cooperating specialists in order to boost the scientific process. Different institutions and federations constantly strive to make advancements in this area and eventually they face the problem of group membership management – arranging users into cooperating groups that reflect the organizational structure and its granularity (e.g. teams, units, departments). Apart from bringing work colleagues together, group structures are often used to control access to organizations' data assets, which are fundamental to collaborative data processing. Data assets (called *assets* through the rest of the paper), are understood as arbitrary, persistent digital information accessible for users to process and collaborate on with others. There is a wide range of solutions that allow organizations to effectively manage user memberships in groups and their access to assets, for example Microsoft Azure Active Directory [20], Internet2's Grouper [11] or various LDAP [16] protocol implementations.

Despite the major potential of modern computing infrastructures, conducting research in the scope of a single organization becomes insufficient. In times of ever growing globalization, innovations require engaging in spontaneous, cross-organizational and interdisciplinary collaboration. To some extent, it can be facilitated using the ideas of Virtual Organizations (VOs) [30] and Ad-hoc user collaboration [18]. The former proposes that geographically dispersed users, groups and organizations create virtual working groups to tackle common challenges, using dedicated software. The latter defines a dynamic working environment where users can engage in short-term and task-specific collaboration that is arranged on demand. These concepts greatly improve the collaborative potential within complex organizations, but they require agreements between participating parties and joint administration. For that reason, they are mostly viable within federated environments, such as Grids.

At the same time, the need for global collaboration spanning over autonomous institutions is growing [12], along with international research programs and grant funding frameworks. To satisfy this demand, we envision of a truly borderless collaborative environment that would embrace the ideas of VOs and Ad-Hoc user collaboration. To make them applicable on a global scale, we propose to back them up with a platform that enables decentralized management of user identities, group memberships and access to assets among autonomous organizations. Such platform could serve as a technological backbone for setting up spontaneous collaborative processes and controlled resource sharing without the prerequisite of establishing a federation.

One of the challenges of the postulated platform is decentralized group membership management, which is addressed in this paper. The essence is to allow management of groups, memberships and access to assets among independent organizations, given large and dynamic structures of groups and assets. In typical setups, groups are arranged into nested tree structures. Assets may constitute a flat collection of independent data sets, but they may also have some interconnecting relations, relevant for determining access to them. These possibly deep structures must be analyzed continuously to enforce access control, which in turn should not hinder the performance of data access. Moreover, relations between users, groups and assets cross the boundaries of organizations, but there is no centralized knowledge and no trust between peers. Nevertheless, it is required that they securely cooperate with each other and jointly realize group membership management in order to enable global, decentralized collaboration. To the best of our knowledge, there are yet no solutions offering such possibilities.

The inspiration for this research was taken from the Onedata system, which strives to provide a decentralized data management platform for global collaboration [33].

Our contribution

This paper proposes a novel framework – a basic structure for decentralized collaborative systems that securely handles managing group membership data. It is based on two main concepts; a) decentralized knowledge base for storing and exchanging information about entities and relations; b) incremental indexing scheme optimized for read-intensive scenarios. The framework allows determining memberships and privileges in nested hierarchies, where evolving relations may cross organizational boundaries. It is targeted at systems using IBAC (Identity Based Access Control), based on client's identity and privileges. This model aligns with scenarios where group structures and user memberships play an important role, e.g. distributed storage in organizations. The main purpose of the framework is to manage access control to assets that are distributed among storage systems of autonomous organizations and thus facilitate collaborative data processing.

This paper is structured as follows. In Section 2, we propose a research problem representation that uses a graph of memberships between entities (users, groups and assets) and break it down into incremental scopes to pinpoint the main difficulties. In Section 3, we present our novel framework for decentralized group membership management. In Section 4, we evaluate the performance and scalability of a prototype implementation of the presented concept. To that end, we simulate large membership graphs and run test procedures that generate changes in the graph, while measuring the significant parameters. We discuss the results and the viability of the solution in decentralized collaborative scenarios.

2. Group membership model

In this section, we show our interpretation of the decentralized group membership management problem using a graph model. It is divided into three incremental steps for better depiction of its intertwined aspects.

2.1. Graph representation of memberships

Organizational structures are essentially a composition of users, groups and assets with interconnecting relations. They can be modeled using directed graphs. Nodes represent entities: users, groups and assets. To achieve unified modeling, all relations are reduced to *memberships*, represented by edges directed from a child (member) entity to a parent (containing) entity. A child entity can be a user or a group (groups can be nested). A parent entity can be a group or an asset. Membership relation between a child entity and a group indicates that the child is a member of the group. Membership relation between a child entity and an asset indicates that the child has access to the data stored within the asset. Entities and membership relations constitute an *entity graph*. An example that reflects above definitions is shown in Figure 1.

The membership relation is transitive. In nested structures, groups and assets can have indirect members, e.g. a user *directly* belonging to a group that *directly* belongs to another group. Such memberships are called *effective*. With this representation, access control to assets can be realized by determining if the client is an *effective member* of the asset.

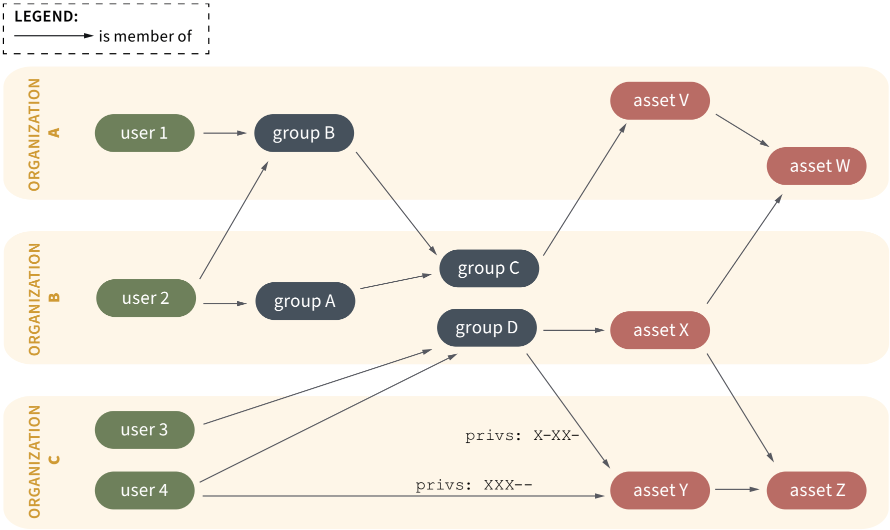


Figure 1. Exemplary entity graph with edges representing memberships and corresponding privileges (selectively annotated)

Determining an *effective membership* is equivalent to a query that tries to find a path between two nodes, which is essentially a reachability problem. If there exists at least one such path, then the starting node (effective child) is an effective member of the target node (effective parent). There can be more than one effective membership path – e.g. in Figure 1, *user 4* is an effective member of *asset Z* and there are three different membership paths between them.

2.2. Membership privileges

Further, the graph representation from the previous section is extended to reflect the needs of collaborative systems, which require modeling privileges along with memberships for finer access control. They are represented using edge attributes.

Consequently, apart from determining an effective membership, it is required to find the privileges of the effective child towards the effective parent. Such query answers the access control question: "does entity X have privileges to do action Y in the scope of asset Z?" The proposed representation dictates that effective privileges be a union of privileges of all direct members of the parent to which the child effectively belongs. Figure 1 presents an example where characters "`X`" and "`-`" represent the possession or lack of an arbitrary privilege, respectively. Here, *user 4* belongs to *asset Y* directly (with privileges `XXX--`) and through *group D*, inheriting its privileges in the asset (`XX-X-`). Hence, the effective privileges of *user 4* in *asset Y* are `XXXX-`.

With such problem definition, to find the effective privileges one must find all effective paths connecting the query nodes, as well as gather and process the privileges during edge traversal, keeping in mind that nesting of groups and assets can be arbitrary.

2.3. Entity graph decentralization

The above-mentioned problems are addressed in the context of decentralized environments, where users, groups and assets originate from different organizations (cf. Figure 1). The organizations remain in a peer-to-peer relation with each other, without a central point or superior coordination.

Membership relations may or may not cross the boundaries of an organization. The information about entities, their relations and privileges is sensitive and cannot be made public – it should be available only to the authorized actors (e.g. entity owners, privileged members, organizations of origin). This effectively limits the knowledge of an organization to a certain subset; each peer knows a different subgraph and remains unaware of the extent of the global graph. Hence, the decentralization significantly complicates matters of both determining effective memberships and privileges as well as storing the graph structure itself. Assuming there might be thousands of peers in the system, the cost of the graph traversal rises even higher.

3. Group membership management framework

The proposed framework is essentially a synergy of our two novel contributions: decentralized knowledge base spanning over the databases of peers and incremental membership indexing scheme. These concepts are discussed through the rest of this section.

This work focuses on group membership management, which should be complemented by other mechanisms for a comprehensive access control implementation. They fall outside of the scope of this paper, while it is assumed that they are already in place:

- The peers (organizations) can globally discover and identify each other, as well as all entities – knowing a global identifier of an entity, each peer must be able to find out its origin. This can be achieved using a decentralized ledger or a DNS + HTTPS based approach, where peer domains are transmitted along with client authorization, as shown in [21].
- There is an infrastructure that verifies user identities (authentication), as the aim of this framework is to manage memberships and streamline access control to assets for authenticated clients (authorization). Typically, peers may use OpenID Connect or SAML for that purpose.
- The framework manages solely the high-level information about assets and memberships. Peers maintain their own storage for the physical assets and no requirements are imposed in that matter. In fact, peers are expected to have existing storage systems and asset collections when joining the peer network. The information about users and memberships managed by the framework is then used to control access to the physical assets (cf. Figure 2).

3.1. Decentralized knowledge base

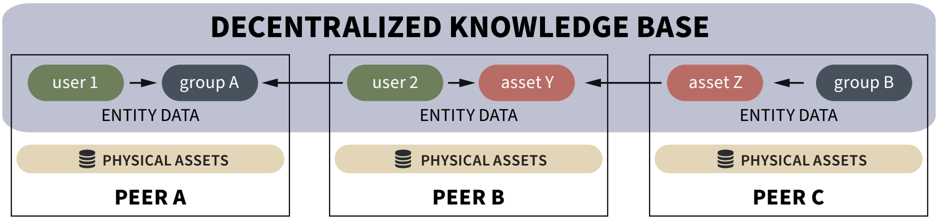


Figure 2. Knowledge base spanning over separate peer databases stores high-level information about entities. Physical assets are stored independently

The purpose of the knowledge base is storing and exchanging information about entities in a decentralized scope, where the global information is effectively split into separate fragments. We find it crucial that each peer (organization) retains full control and possession of their database, thus we eliminated decentralized data stores, such as blockchain ledgers or DHTs. Our approach assumes that each entity has its peer of origin, which locally stores its information and membership data. The peer has authoritative control over the entity and handles all modifications of the entity or its relations. Peers share only the information required for collaboration with other peers, using secure end-to-end channels. Based on the entity graph and information about origin of different entities, a peer can decide to disclose certain information to another peer. For example, *Peer B* can request information about *group A* from *Peer A*, because one of its users belongs to the group – cf. Figure 2. In this approach, the database storing the entity data itself is not decentralized, but a decentralized knowledge base is introduced – a data exchange layer on top of the peer databases that uses secure end-to-end channels for accessing remote entity data. This information is used by the peers to make access control decisions regarding the assets located on their storage systems.

```
struct DirectIndex :
    neighbors: map from NodeId to Privileges
end
struct EffectiveIndex :
    effectiveNeighbors: map from NodeId to EffectiveIndexValue
end
struct EffectiveIndexValue :
    effectivePrivileges: Privileges
    intermediaries: set of NodeId
end
```

Figure 3. Structures of direct and effective indices

Entity data includes several basic attributes, such as name, creation time and configuration, along with entity's relations, which are essential for membership management. It can be stored using arbitrary database technology, as long as it complies with the following guidelines. Node relations (connected edges) are persisted in an index – a key-value store that supports queries in constant or logarithmic time (it may be based on a hash table or a B+ tree, for example).

There are two types of indices, corresponding to relation types – direct and effective. Their structures are shown in Figure 3. In a direct index, the key denotes the node's direct neighbor ID and the value encodes the privileges assigned to the relation. In an effective index, the key denotes the node's effective neighbor ID and the value encodes precalculated effective privileges and intermediaries. *Intermediary* for an effective relation is a direct neighbor of a node through which an effective relation path goes, and each effective neighbor can have any number of intermediaries. If there is a direct relation with the effective neighbor, the neighbor itself is added as an intermediary. A representation of node indices with exemplary content is shown in the following subsection. Each node (entity) has at most four indices: direct children, direct parents, effective children, effective parents. Privileges are stored only in child indices; in parent indices this value is ignored. Hence, to find direct/effective privileges of a user in a group, one must consult the group's direct/effective children index and look up the entry for the user.

3.2. Incremental indexing scheme

The indexing scheme defines how to process the information stored in the knowledge base to answer queries about effective memberships and privileges in the entity graph. It is optimized for fast queries at the cost of graph update performance, due to read-intensive characteristics of the target systems. We argue that in typical collaborative scenarios, relations change occasionally (e.g. a user is moved to another group), but the membership information is queried constantly – during every access to an asset. Minimizing the query times is especially important for this solution to be applicable in systems dedicated for data processing, where the overheads of access control must be as low as possible.

Hence, we propose full indexing of effective memberships and privileges in a continuous process of incremental index adjustments. There are three types of operations that can modify the graph: addition of a relation, update of privileges assigned to a relation, and deletion of a relation. Each such modification triggers reconciliation of the indices, but only over the subgraph affected by the change. The reconciliation process reuses the existing indices, making minimal adjustments stemming from the changed relation. Thus, complexity of an update operation depends on the current graph structure and the number of nodes affected by the update. The main advantage is that queries are fast, being a simple look-up in a precomputed index.

In the following subsections, we firstly present the general principles of the indexing scheme, then we show how it is jointly realized by independent peers.

3.2.1. Event-driven reindexing

The incremental indexing scheme utilizes the aforementioned persistent indices. Upon a relation change request, the direct relation indices are modified in a transactional (atomic) way. For instance, when a user is added to a group, two entries are added: 1) the group is added to the user's direct parents index and 2) the user is added to the group's direct children index along with the assigned privileges. The transaction also emits asynchronous events, triggering recomputation of effective indices in the graph. Consequently, direct indices are immediately consistent, while effective indices are eventually consistent – they converge when all related events are processed.

Events are the basic unit of information and drive index updates by propagating through the graph. Each event is assigned to a specific node and falls into the event queue designated for the node. Events are handled in parallel by a pool of reindexing processes that continuously poll the queues, but with the restriction that all events concerning the same node must be processed sequentially to avoid race conditions when updating indices. Event structure is presented in Figure 4 – it carries information about a change in effective relations of an entity: either their addition, update, or removal (i.e. "*user 1* and *user 2* are added as effective children via intermediary *group C*").

```
struct Event :
            operationType: enum { ADD, UPDATE, DELETE }
            direction: enum { TOP_DOWN, BOTTOM_UP }
            intermediary: NodeId
            effectiveNeighbors: set of NodeId
end
```

Figure 4. Structure of an event

Event propagation is understood as creating descendant events for each direct neighbor of the subject node. There are two possible directions of propagation: bottom-up (from children towards parents) and top-down (from parents towards children). Propagation is always done in the same direction as of the original event, e.g. a bottom-up event causes descendant bottom-up events to be emitted for each direct parent of the node (cf. Figure 6). The nodes receiving descendant events are called *successors*.

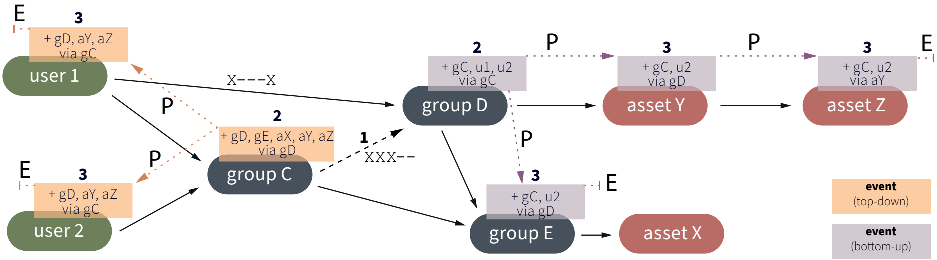
Events are persisted since the moment of creation until processed, to ensure durability of changes to the graph in case of any sudden system failures. For the same reason, there is one persistent queue of events that are yet to be processed, internally divided into sub-queues for specific nodes. Events may be prioritized arbitrarily, but the framework assumes that events for the same node must be enqueued strictly in the order of creation – which is straightforward, as updates of each node (which cause event emission) are done in non-overlapping transactions.

When event e concerning node v is pulled from the event queue, it is processed according to the Process-Event(v, e) procedure, shown in Figure 5. The procedure operates on one of the node's effective indices depending on the event's direction (cf. lines 9–15). The index is updated depending on the operation type and its previous state for each effective neighbor referenced by the event (cf. lines 19–39). Recalculation of privileges is triggered only in case of bottom-up events that concerned an update operation or caused any change in the set of intermediaries. New effective privileges are a union of all direct privileges of the intermediaries (cf. lines 40–42). After the index is reconciled, a further propagation of the event is considered (cf. lines 44–53). Propagation is required only if the reconciliation caused any effective neighbor to be added or deleted. In such case, descendant events are emitted for each successor of the node. If there are no successors or no changes in effective neighbors, the propagation ends. Finally, the event is considered processed.

Figure 6 illustrates an exemplary course of the reindexing algorithm, triggered by adding *group C* as a child of *group D*. The transaction adds new entries to the direct indices of the nodes **(1)** and emits two events propagating in opposite directions **(2)**. In the course of the propagation **(3)**, *group C* and its children are notified about their new effective parents, while *group D* and its parents are notified about their new effective children, with the privileges inherited along the propagation path. Successive propagation is marked with **(P)**, while **(E)** denotes where it ends. The *user 1* node is an example where there are no successor nodes.

```
procedure PROCESS-EVENT($v, e$)
    ▷ $v$: node
    ▷ $e$: event to process
    ▷ $\mathcal{D}_c(v)$: index of direct children of $v$
    ▷ $\mathcal{D}_p(v)$: index of direct parents of $v$
    ▷ $\mathcal{E}_c(v)$: index of effective children of $v$
    ▷ $\mathcal{E}_p(v)$: index of effective parents of $v$
    toPropagate ← {}
    if $e$.direction = TOP_DOWN then
        $\Omega \leftarrow \mathcal{E}_p(v)$
        successors ← $\{u : \mathcal{D}_c(v)$.neighbors$[u] \neq nil\}$
    else
        $\Omega \leftarrow \mathcal{E}_c(v)$
        successors ← $\{u : \mathcal{D}_p(v)$.neighbors$[u] \neq nil\}$
    end if
    for $v_n \in e$.effectiveNeighbors do
        $\omega \leftarrow \Omega[v_n]$                              ▷ index value for the effective neighbor
        recalculatePrivileges ← false
        if $e$.operationType = ADD then
            if $\omega = nil$ then
                $\Omega[v_n] \leftarrow \omega \leftarrow$ new EffectiveIndexValue
                toPropagate ← toPropagate $\cup \{v_n\}$
            end if
            if $e$.intermediary $\notin \omega$.intermediaries then
                $\omega$.intermediaries ← $\omega$.intermediaries $\cup$ {$e$.intermediary}
                recalculatePrivileges ← true
            end if
        else if $e$.operationType = DELETE then
            if $\omega = nil \lor e$.intermediary $\notin \omega$.intermediaries then continue
            $\omega$.intermediaries ← $\omega$.intermediaries $\setminus$ {$e$.intermediary}
            if $\omega$.intermediaries = {} then
                $\Omega[v_n] \leftarrow nil$
                toPropagate ← toPropagate $\cup \{v_n\}$
            else
                recalculatePrivileges ← true
            end if
        else if $e$.operationType = UPDATE then
            recalculatePrivileges ← true
        end if
        if $e$.direction = BOTTOM_UP $\land$ recalculatePrivileges then
            $\omega$.effectivePrivileges ← $\bigcup_{i \in \omega.\mathrm{intermediaries}} \mathcal{D}_c(v)$.neighbors$[i]$
        end if
    end for
    if |toPropagate| > 0 then
        $e_p$ ← new Event
        $e_p$.operationType ← $e$.operationType
        $e_p$.direction ← $e$.direction
        $e_p$.intermediary ← $v$
        $e_p$.effectiveNeighbors ← toPropagate
        for $v_s \in$ successors do
            send $e_p$ to $v_s$
        end for
    end if
end procedure
```

Figure 5. Procedure PROCESS-EVENT(v, e)



LEGEND: **1** Direct relation is added.
2 Initial events are emitted during the update transaction.
3 Events are asynchronously processed and propagated until all the affected indices converge.
P Further event propagation.
E End of event propagation.

Figure 6. Exemplary course of reindexing after a relation is added

The case of no changes in the effective neighbors can be observed at *group E*, which already had all the effective children in its index due to the preexisting memberships of *group C* and *group D*. As operations on a node are done in non-overlapping transactions and events assigned to a single node are processed sequentially in the order of creation, it can be assumed that *asset X* must have already been notified about these effective children as a result of previous graph updates. Hence, *asset X* is not affected by the operation and requires no recomputation. Finally, Figure 7 depicts the state of indices of *group D* after all events are processed.

group D – direct children	
	privileges
group C	XXX--
user 1	X---X

group D – effective children		
	eff. privileges	intermediaries
group C	XXX--	group C
user 1	XXX-X	user 1, group C
user 2	XXX--	group C

group D – direct parents	
	privileges
group E	N/A
asset Y	N/A

group D – effective parents		
	eff. privileges	intermediaries
group E	N/A	group E
asset Y	N/A	asset Y
asset X	N/A	group E
asset Z	N/A	asset Y

Figure 7. Indices of group D after recomputation

An additional benefit of storing intermediaries in effective indices is the possibility to rebuild all effective paths connecting nodes A and B, by starting from node A and recursively visiting all intermediaries of its effective neighbor B.

3.2.2. Decentralization of the Indexing Scheme

The scheme dictates that the entity's peer of origin stores all its indices and is responsible for recomputing its effective indices. Peers cooperate only in matters regarding cross-organizational relations – cf. Figure 8. Firstly, when such relation is added, modified or deleted, the peer responsible for the parent node coordinates a distributed transaction that succeeds when the two peers update their direct indices and generate corresponding events. Secondly, when local index recomputation emits an event concerning a node in another peer, it is placed in a persistent outbox and waits as long as required for the other peer to consume it. Thirdly, a peer releases local entity details only if it has a relation with an entity from the requesting peer. It is used by the peer to display information about a remote entity to its local users. The details are fetched by its global entity ID, held in membership indices.

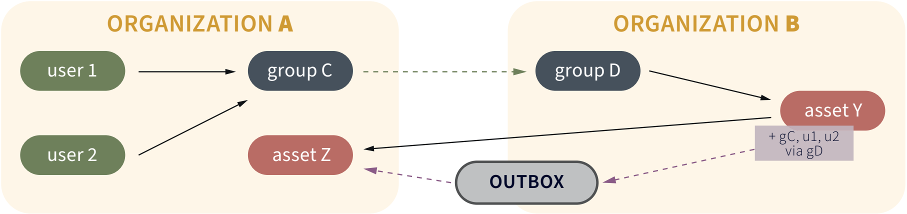


Figure 8. Decentralization of the indexing scheme – peers cooperate in distributed transactions and exchange events using persistent outboxes

Peers in the environment are expected to be in different operational states; some may be offline, some may be busy and falling behind with processing of events, while others may be idle. If peers are operational, the convergence times depend on the size of the subgraph affected by the change and the current workload of the participating parties. On the other hand, any long-lasting downtime or network outage respectively delays the event propagation, which is clearly beyond the influence of the scheme. Nevertheless, thanks to the persistent outboxes, it is guaranteed that all involved peers will eventually acknowledge a change to the graph that has been made elsewhere. With this approach, from the global point of view, the system is eventually consistent, but allows peers to remain independent and thus to provide uninterrupted services for their local communities, while realizing cross-organizational access in a best-effort manner.

3.3. Security, privacy and trust considerations

This framework assumes no inherent trust between individual organizations in the environment, which are determined to ensure security and privacy for their users and assets. Despite that, it proposes a way to arrange decentralized collaboration based on willful exchange of information in a limited and controlled manner. In regard to security, privacy and trust, this work builds on [21], where the authors presented the concept of trust-driven access control between autonomous peers and shown how trust statements can be used to facilitate secure information flow between independent parties.

The proposed approach is applicable only if the peers recognize the need of their users to reach out to people from other organizations and agree to share some information with the outside world, in return receiving the same from other peers. Access to information depends on relations between users, groups and assets, and in fact the creation of these relations is the impulse for a pair of peers to start cooperating. For example, when a user from a remote peer is invited to a local group, the local peer starts to recognize the user and grants him access to the parent assets of the group – but nothing else. At every level, the peers follow the rule of limited trust, but accept foreign users that were entrusted by local users and invited for collaboration.

The process of inviting people to collaborate differs depending on policies in each organization; for instance it may be overseen by administrators, or available to some privileged users (e.g. asset owners). The extent of information exchanged by the peers is limited to minimum required for collaboration, and only to entities connected by a cross-organizational relation. It is communicated using secure end-to-end channels, which means that both sides are responsible for keeping the information private and secure. However, especially as the system is decentralized, it cannot be assumed that all peers have good intentions. In fact, malicious parties may join the P2P network and try to extract sensitive information. For that reason, each organization should implement some security measures to have control over what external parties are allowed to participate in collaboration. As organizations have very different policies and administrative processes, the framework does not impose any concrete requirements in this matter. To name a few possible approaches, they might be based on trust models, use mechanisms such as auditing, whitelisting, blacklisting or depend on some real-life agreements between organizations. In specific cases, a peer might even choose to remain completely isolated and realize the group membership management only for internal purposes, or cooperate with a strictly limited list of trusted peers. In case a peer does not implement any security measures to control the pool of participating peers, the data security lies entirely in the hands of its users. Unauthorized data extraction may appear if the users entrust some malicious party and invite it for collaboration, but only in the scope of groups and assets where it has been granted access. Nevertheless, such peer with impaired security does not impact the security of others – each peer oversees the security of entities originating from it and applies its own security measures when determining which peers are allowed to access the entity data.

4. Evaluation

In order to verify the viability of our framework, we implemented a prototype in Java, called *gmm-indexer*, with Redis database as underlying persistence. Its native key-value store based on a hash table was used for persisting direct/effective membership indices. We conducted performance tests measuring query times and the throughput of update operations at different workload levels. All tests were also run for a naive implementation, which simply stored the graph without indexing and traversed it at every query. The aim was to estimate the gain in terms of query performance offered by the precomputed indices and the overheads introduced by our indexing mechanism. The prototype implementation and scripts used for tests are publicly available in a Github repository [10].

The testing environment consisted of 4 machines, each with 4 core CPU (2.66GHz) and 16GB of RAM. One machine was used solely to make requests to the other three, each of which hosted one peer – simulating three independent organizations. For each test, we simulated a preexisting entity graph, generating a hierarchy of assets, groups and users. Even though the tests were intended to serve as a proof of concept rather than an accurate simulation, we strove to generate graphs corresponding to actual structures of real organizations. Based on the numbers for one of the biggest scientific research centres – CERN – that had 3459 employees and 12731 community members at the end of 2021 [2], we deemed the number of 5000 entities a good representation of a large organization. The nesting level and member count of each asset or group was randomized using a normal distribution. Generated graphs had about 15k nodes and 19k edges spanning over three peers, which gives about 5k entities per organization. We identified one parameter that is especially significant, namely *cross-organizational factor*, indicating how many relations (edges) cross the boundaries of a peer. At 0%, there is no cooperation between peers in recomputation of the indices. The higher this parameter, the bigger the overheads caused by P2P communication, required to propagate the events and process all changes made to the graph. From a single peer's point of view, this parameter is arguably more important than the total number of peers in the environment, as it regulates how often the peer has to communicate with any of the external parties.

Table 1

Performance test results – query times for *gmm-indexer* and naive implementation

Query times [ms]		Query type / Cross-organization factor							
		is member		is mem. non-ex.		eff. privileges		eff. members	
impl.	metric	0%	10%	0%	10%	0%	10%	0%	10%
naive	avg	4.7	5.5	11.1	12.7	7.7	7.2	34.0	47.3
naive	max	91.2	82.2	152.7	110.4	75.4	98.2	370.2	340.2
gmm-indexer	avg	1.3	1.3	0.9	0.9	1.4	1.3	2.2	2.1
gmm-indexer	max	5.4	48.9	38.6	35.4	40.9	14.1	52.7	50.2

After preparing the initial graph, we generated a set of queries based on the actual graph structure and started the client that continuously sent requests to all peers and measured relevant parameters over longer time periods, to collect representative averaged measurements. The test results are shown in Table 1, which presents a comparison of query times depending on the query type and cross-organizational factor (0% or 10%). The following types of queries were performed:

- Query *is member* – a query checking if an entity is a member of another entity. These queries were generated for existing memberships, i.e. the query result was always positive. In this case, the naive algorithm could yield an answer after traversing only some of the membership paths.
- Query *is mem. non-ex.* – like above, but queries were generated for non-existing memberships, i.e. the query result was always negative. This case is harder for the naive implementation as it requires traversal of all membership paths before reaching the final decision.
- Query *eff. privileges* – a query that returns computed effective privileges of a child entity towards a parent entity for an existing membership. Causes the naive algorithm to traverse all membership paths.
- Query *eff. members* – a query that returns computed effective members of an entity. Causes the naive algorithm to traverse all membership paths. The traversed subgraph is typically larger that in case of previous query types, because the traversal is done towards children rather than parents, which causes more branching.

Thanks to the precomputed indices, we achieved about 3–25× faster average query times, which were very similar for all test cases and oscillated around 1–2 milliseconds. The table also includes the maximum query time that was observed during each test case. For *gmm-indexer*, they depict maximum spikes in latency during communication with the underlying database, but for the naive implementation, they can be attributed to the cases when a long graph traversal was required.

For update throughput tests, the client maintained a constant workload expressed in operations per second. We measured how many operations were finished by each peer in a time period – for the naive implementation, an operation was considered finished when the graph structure was updated, and for *gmm-indexer*, when all related events have been processed. The results for different workload levels are presented in Figure 9.

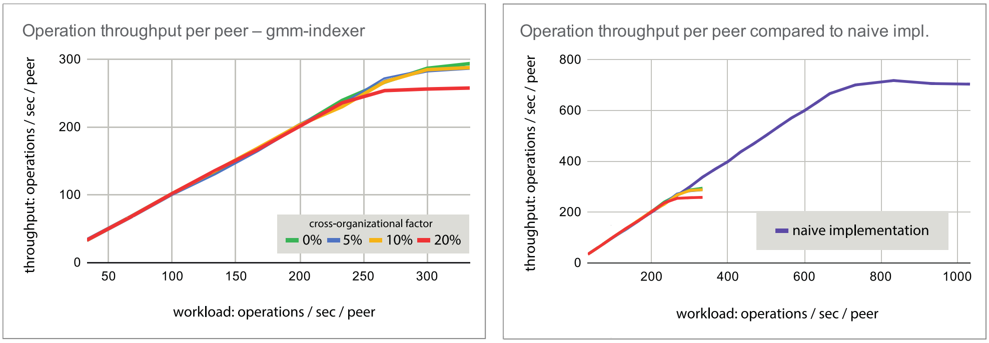


Figure 9. Performance test results – operation throughput per peer

For each cross-organizational factor, *gmm-indexer* performed about 3× worse than the naive implementation, which depicts the cost of reindexing that was continuously performed in the background. Starting at about 230 requests/sec/peer, the *gmm-indexer* implementation was no more able to handle all incoming requests on the fly and they started to pile up. On the other hand, the naive implementation was able to process about 650 requests/sec/peer until the test machine resources began to deplete. Looking at different data series for *gmm-indexer*, it can be inferred that the cost of graph updates generally grows with the cross-organizational factor. Nevertheless, the graph had to be regenerated for each test case due to the changing cross-organizational factor, so the results give only a rough comparison. Table 2 shows average and maximum measured operation times in different test cases. The average times for *gmm-indexer* were usually lower than 15 milliseconds under medium load, but rose to about 20 seconds under heavy load. Maximum propagation times depict the extreme cases when an operation caused a recomputation of a large chunk of the graph. This means that, in rare cases and under heavy load, a user might have to wait several minutes until their operation (e.g. joining a group) is processed.

Table 2

Performance test results – update times for *gmm-indexer* and naive implementation

Update times [s]		Workload: operations / sec / peer									
impl. / cross-org. f.	metric	33	67	100	133	167	200	233	267	300	333
naive / N/A	avg	0.008	0.007	0.007	0.007	0.007	0.006	0.006	0.006	0.007	0.008
naive / N/A	max	0.133	0.213	0.102	0.111	0.217	0.114	0.148	0.103	0.081	0.219
gmm-indexer / 0%	avg	0.016	0.014	0.013	0.014	0.015	0.029	2.445	12.06	16.47	19.67
gmm-indexer / 0%	max	0.494	0.458	0.349	0.306	0.311	1.711	67.77	182.5	175.8	179.2
gmm-indexer / 5%	avg	0.016	0.014	0.013	0.014	0.016	0.052	3.479	15.40	21.13	21.18
gmm-indexer / 5%	max	0.421	0.374	0.254	0.339	0.488	2.724	71.21	179.1	193.9	186.5
gmm-indexer / 10%	avg	0.015	0.014	0.013	0.013	0.015	0.031	2.565	12.22	19.37	20.71
gmm-indexer / 10%	max	0.427	0.327	0.284	0.221	0.304	1.338	58.50	144.2	175.8	187.1
gmm-indexer / 20%	avg	0.016	0.014	0.013	0.014	0.016	1.341	9.394	15.34	15.63	18.08
gmm-indexer / 20%	max	0.496	0.390	0.305	0.433	0.327	50.75	178.4	205.3	204.1	208.0

4.1. Discussion

In our update performance tests, we simulated exaggerated conditions and large organizations. If we assume that an entity generates 1 operation per day (i.e. once a day user's membership in groups or assets change), the expected total workload at 5k entities is about 0.06 operations per second. This means that our framework, despite the reindexing overheads, requires only a fraction of the available throughput to handle organizations with thousands of users, where relations are changing with intensity adequate to long-lasting scientific projects. What is more, our tests used a single commodity machine per organization, which leaves room for horizontal scaling.

In our simulations, each organization offered update throughput well beyond required, even at 20% of cross-organizational factor. Consequently, from the global point of view, we argue that the framework is able to handle millions of users working in thousands of organizations. Thanks to the precomputed indices, we were able to achieve low query times, making our solution a good candidate for realizing access control in distributed data processing environments. It is worth noting that in our prototype, the indices were stored in an third-party database. Introducing an in-memory caching layer for the indices could further reduce the query times.

The current trends in decentralized applications are dominated by the blockchain technology, as it naturally solves the problem of achieving agreement amongst independent parties. However, we deemed that its shortcomings such as inherent lack of data privacy, limited transaction throughput and dependence on well-established peer network make it less suitable for this research problem. Hence, we decided to make a case for a custom tailored solution, which has considerable merits that are not straightforward to achieve using a blockchain ledger:

- Scalability – the global knowledge is split into subsets, each maintained by an autonomous peer, where only the information about relations crossing organizational boundaries are shared between peers. This means that the total size of the peer network has low impact on each separate peer. We believe it is a natural implication of our sparse network architecture, nevertheless we are working towards simulating large networks and performing scalability tests with millions of entities to back up this claim.
- High isolation and autonomy of peers – there is no interconnected, global network or ledger in which all peers have to participate, rather than that they create sparse connections wherever cooperation is instantiated. A typical peer will discover only a small portion of all peers and entities in the global scope during its lifecycle.
- Limited, privacy retaining data flow – exchange of information is possible only concerning cooperating entities in the context of a cross-organizational relation and is sanctioned by mutual trust of corresponding users, groups and organizations. The data is transmitted over secure end-to-end channels between the peers, rather than published in the scope of the global network.
- Manageability – our framework builds a layer upon existing infrastructures, making integration and maintenance more manageable.

5. Related work

The framework presented in this work is a high-level concept that touches the topics of group membership management, decentralized collaboration and access control, as well as determining graph reachability.

5.1. Group membership management

Group membership management is an inherent part of many identity management systems and an indispensable tool for any organization. There are numerous open source projects (e.g. Internet2's Grouper [11]) and commercial products (e.g. Microsoft Azure AD [20]) that support group membership management in centrally managed environments. They often support nested group structures and provide tools for determining effective membership (also called transitive membership in Azure AD). However, there is not much research done in the area of decentralized systems, and it is mostly related to Web 2.0 concepts and groups in decentralized social networking [35]. For instance, in [25] the authors propose a distributed group management framework using Friend-of-a-Friend vocabulary. It is independent of existing platforms, without a central point and can be used for restricting access to web resources.

5.2. Decentralized collaboration and data access control

The idea of decentralized collaboration and decentralized data access has been present for years in the scientific community and can be perceived in several different contexts, but regardless of its area of application and goals, it usually must be accompanied by some kind of access control.

Firstly, there are general purpose access control frameworks that can be used to control access to resources in a decentralized environment. This area is undergoing intensive research lately, especially after emergence of the blockchain technology and its refinements such as smart contracts. They can be used to codify access control policies, as shown in [5] that embraces Attribute Based Access Control (ABAC) model. Another work [19] proposes using a private ledger to store file locations (IPFS is used for data storage) and digital fingerprints, while using a public ledger to store ABAC attributes. Among non-blockchain propositions, there is for example a concept of multi-tier user authentication for cloud storage that employs multiple Key Distribution Centers and cryptography backed up by steganography for data security – only legitimate clients can decrypt the stored data [27].

Secondly, there are attempts at creating frameworks that facilitate scientific collaboration between independent institutions. Again, some researchers took the approach of using a blockchain ledger as the backbone for decentralized networking. For example, in [17], the authors propose a new approach to arranging collaboration in virtual organizations by using an enriched version of Git VCS that stores commit hashes using smart contracts and IPFS data storage, in order to securely manage authorship, system access privileges and licensing policies in a decentralized environment. Other ideas include for instance the Teamwork system [7] that stores the content at users' devices and provides a P2P overlay network for content distribution with enhanced privacy. The MIDEP [15] protocol offers public identity management for decentralized collaborative architectures that puts bigger focus on security and resistance to malicious attacks, without assuming any central point. Another related research field is Virtual Organizations where access control must be suited to dynamic coalitions. This problem is addressed for example in [13], where the authors propose a fully decentralized framework embracing role-based access control with custom policies verifiable using threshold BLS signature schema. Another contribution worth noting is [4], proposing a data-driven coordination middleware for dynamic collaboration of autonomous peers. This work leverages ABAC and similarly to ours, bases security (to some extent) upon trust assumptions. It supports fine-grained policies that are used to authorize data access, service invocations, dynamic behavior changes and policy updates.

Thirdly, collaboration and access control in ubiquitous and ad-hoc environments is another closely related field of research. With the omnipresence of mobile and IoT devices, there is a need of secure coordination between peers in quickly changing environments without a central point in order to utilize their collective computational potential. Some of the works include for instance a framework for decentralization and management of collaborative applications based on trust models [22], a decentralized access control model for IoT with Decentralized ID (DID) [14] and an access control system implementing sociological trust concepts to evaluate interaction partners [1].

Finally, there are works that address the general idea of collaboration between independent peers and have a lot in common with our research problem, despite having different aim than collaborative data processing. In [3], the authors present an architecture called `SOMEWHERE2` for decentralized, collaborative consequence finding, where each peer in the network is not aware of the global theory used for reasoning, but only knows its local theory and the variables shared with its neighbors. Another interesting effort was made in [29], proposing a blockchain based approach for cheat detection in multiplayer games on the edge. This solution defines a completely decentralized architecture for edge devices, entailing an innovative consensus mechanism based on verifiable delay functions.

The discussed solutions address significant challenges of decentralized collaboration and data access control, but not the problem of cross-organizational group membership management.

5.3. Graph reachability

Determining reachability is a well studied topic with many insights into query optimizations, possible heuristics for approximation, as well as partial or full reachability precomputation. Some representative examples include ideas to maintain a transitive closure while the graph is constantly changing. For example, in [9] the authors proposed two dynamic algorithms that constantly recompute the transitive closure and can be applied for real-world graphs. Roditty and Zwick applied further refinements and added their original ideas to obtain a decremental algorithm for maintaining the transitive closure with a total expected time of O(mn), among several other algorithms for optimized reachability queries [23]. The same authors later published another fully dynamic reachability algorithm with an almost linear update time [24]. Another example is FERRARI [26] – a scalable index structure for the reachability problem that adaptively compresses the transitive closure during construction of the graph, yielding fast query times with reduced index size. These solutions could be used to address the group membership management problem, but they all assume centralized knowledge about the graph. There have been attempts at privacy-preserving reachability algorithms coping with distributed graphs, but they have major limitations. For instance, in [6] the authors show how to perform reachability queries on a distributed graph and private sets of edges. However, the research assumes a two-party system and guarantees security in a semi-honest model, ensuring that only required information is shared between parties.

5.4. Summary

There has been a lot of research done concerning the abovementioned areas, but seemingly there is none that directly addresses our research problem, i.e. collaboration-oriented decentralized group membership management based on membership graphs.

Table 3
Summary of related works in relation to paper's research problems

	Group memb. management	Decentralized collaboration	Decentralized access control	Graph reachability
Internet2 Grouper [11]	x	–	–	x
Microsoft Azure AD [20]	x	–	–	x
dg-FOAF [25]	x	–	x	–
Blockchain based access control [5, 19]	–	–	x	–
Dec. access control with multi-tier authentication [27]	–	–	x	–
Dec. Blockchain-based platform for collaboration [17]	–	x	x	–
Teamwork [7]	–	x	–	–
MIDEP [15]	–	x	–	–
VO-Sec [13]	–	x	x	–
Dec. access control model for dynamic collaboration [4]	–	x	x	–
Decentralized access control for IoT [1, 14]	–	x	x	–
Framework for mngmt. of dec. collaborative applications [22]	–	x	–	–
SOMEWHERE2 [3]	–	x	–	–
Dec. authoritative multiplayer architecture [29]	–	x	–	–
Maintaining transitive closure incrementally [9, 23, 24, 26]	–	–	–	x
Secure reachability query on private shared graphs [6]	–	–	–	x

A summary of the discussed related works in relation to the research topics addressed in this paper is presented in Table–3. We take inspiration from different approaches to realizing decentralized collaboration, group management and solving the graph reachability problem using fast, precomputed indices in order to create a–novel solution.

6. Conclusions and future work

In this paper, we present a novel framework for decentralized group membership management. With its ability to manage information about dynamically changing memberships without a central point, as well as determine effective memberships and privileges using fast look-ups, we believe that our framework is an important step towards global, decentralized collaboration. It can be used to realize access control to assets in environments consisting of multiple data centers and facilitate distributed data processing among unfederated organizations, introducing low overheads and thus low impact on data access performance.

The proposed framework is built upon two concepts: a decentralized knowledge base and an incremental indexing scheme. The knowledge base allows storing and exchanging entity data and relations without a central point, assuming a sparse peer network that splits the global knowledge into subsets with limited overlap, which is advantageous in the context of scalability. It uses secure channels between peers that retain full autonomy, where the flow of information is subject to relations in the graph and must be sanctioned by both peers. The proposed incremental indexing scheme optimizes queries for effective memberships and privileges as much as possible, at the cost of graph update performance, based on an assumption that queries are orders of magnitude more frequent. As a proof of concept, we present a synthetic benchmark to compare a prototype implementation of our proposition with a naive implementation, simulating an environment with 3 peers and graphs with 15k entities. The performance test results show that our prototype yields 3-25x faster query times, while the update throughput is about 3x lower. Nevertheless, we argue that in typical scenarios, only a fraction of the framework's update throughput will be used.

For future work, we plan to perform tests on a larger scale to empirically explore the capacity and scalability of our framework. We are working on some refinements and optimizations, among which is introducing an in-memory caching layer for the precomputed effective membership indices, to further reduce the query times – which is not straightforward as the knowledge base convergence during updates should not be impaired. Finally, we intend to implement the proposed framework in the Onedata system, which we hope to be a major milestone in truly decentralized scientific collaboration between independent organizations.

Acknowledgments

This scientific work was published in part by an international project co-financed by The National Centre for Research and Development under the program entitled "ERA-NET CO-FUND ICT-AGRI-FOOD", contract No. ICTAGRI-FOOD/I/FINDR/02/2022. KJ, RGS and JK are grateful for support from the subvention of Polish Ministry of Education and Science assigned to AGH University of Science and Technology.

References

[1] Adams W., Davis N.: Toward a decentralized trust-based access control system for dynamic collaboration. In: *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*, pp. 317–324, 2005. doi: 10.1109/IAW.2005.1495969.

[2] CERN Personnel Statistics 2021. https://cds.cern.ch/record/2809746/files/CERN-HR-STAFF-STAT-2021-RESTR.pdf.

[3] Chatalic P., de Amorim Fonseca A.: A Multi-Layered Architecture for Collaborative and Decentralized Consequence Finding, *Computing and Informatics*, vol. 34(1), pp. 210–232, 2015.

[4] Craß S., Joskowicz G., Kühn E.: A Decentralized Access Control Model for Dynamic Collaboration of Autonomous Peers. In: B. Thuraisingham, X. Wang, V. Yegneswaran (eds.), *Security and Privacy in Communication Networks. SecureComm 2015*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 164, pp. 519–537, Springer, Cham, 2015. doi: 10.1007/978-3-319-28865-9_28.

[5] Di Francesco Maesa D., Mori P., Ricci L.: Blockchain Based Access Control Services. In: *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pp. 1379–1386, 2018. doi: 10.1109/Cybermatics_2018.2018.00237.

[6] Do H., Ng W.K.: Secure reachability query on private shared graphs. In: *IEEE Ninth International Conference on Intelligent Sensors, Sensor Networks and Information Processing (ISSNIP)*, pp. 1–6, 2014.

[7] Draghici A., Burloiu C.A., Deaconescu R., Karlsson M., Müller D.: Teamwork: A Decentralized, Secure and Portable Team Management System. In: *12th International Symposium on Parallel and Distributed Computing*, pp. 182–189, 2013.

[8] Foster I.: The Anatomy of the Grid: Enabling Scalable Virtual Organizations. In: R. Sakellariou, J. Gurd, L. Freeman, J. Keane (eds.), *Euro-Par 2001 Parallel Processing*, Lecture Notes in Computer Science, vol. 2150, pp. 1–4, Springer, Berlin–Heidelberg, 2001. doi: 10.1007/3-540-44681-8_1.

[9] Frigioni D., Miller T., Nanni U., Zaroliagis C.: An Experimental Study of Dynamic Algorithms for Transitive Closure, *ACM Journal of Experimental Algorithmics*, vol. 6, p. 9–50, 2001. doi: 10.1145/945394.945403.

[10] Public Github repository with the gmm-indexer prototype and scripts. https://github.com/kjarosh/agh-gmmf-prototype.

[11] Internet2's Grouper. https://incommon.org/software/grouper.

[12] Jemielniak D., Przegalinska A.: *Collaborative society*, MIT Press, 2020.

[13] Jin H., Qiang W., Shi X., Zou D.: VO-Sec: An Access Control Framework for Dynamic Virtual Organization. In: C. Boyd, J.M. González Nieto (eds.), *Information Security and Privacy*, Lecture Notes in Computer Science, vol. 3574, pp. 370–381, Springer, Berlin–Heidelberg, 2005. doi: 10.1007/11506157_31.

[14] Jung E.: A Decentralized Access Control Model for IoT with DID. In: H. Kim, K.J. Kim (eds.), *IT Convergence and Security*, Lecture Notes in Electrical Engineering, vol. 712, pp. 141–148, Springer, Singapore, 2021. doi: 10.1007/978-981-15-9354-3_14.

[15] Khan R., Hasan R.: MIDEP: Multiparty Identity Establishment Protocol for Decentralized Collaborative Services. In: *IEEE International Conference on Services Computing*, pp. 546–553, 2015.

[16] Lightweight Directory Access Protocol (LDAP). https://ldap.com.

[17] Lenko V., Kunanets N., Pasichnyk V., Shcherbyna Y.M.: Decentralized Blockchain-based platform for collaboration in virtual scientific communities, *ECONTECHMOD*, vol. 8(1), pp. 21–26, 2019.

[18] Lorch M., Kafura D.: Supporting Secure Ad-hoc User Collaboration in Grid Environments. In: *Grid Computing – GRID 2002*, Lecture Notes in Computer Science, vol. 2536, pp. 181–193, Springer, Berlin–Heidelberg, 2002. doi: 10.1007/3-540-36133-2_16.

[19] Men R.: Research on access control method of Digital Archives based on blockchain, *Journal of Physics: Conference Series*, vol. 1550, 2020. doi: 10.1088/1742-6596/1550/6/062021.

[20] Microsoft Azure Active Directory. https://azure.microsoft.com/en-us/services/active-directory.

[21] Opioła Ł., Dutka Ł., Słota R.G., Kitowski J.: Trust-driven, Decentralized Data Access Control for Open Network of Autonomous Data Providers. In: *2018 16th Annual Conference on Privacy, Security and Trust (PST)*, pp. 1–10, 2018.

[22] Quinn K., Kenny A., Feeney K., Lewis D., O'Sullivan D., Wade V.: A Framework for the Decentralisation and Management of Collaborative Applications in Ubiquitous Computing Environments. In: *2006 IEEE/IFIP Network Operations and Management Symposium NOMS 2006*, pp. 1–4, 2006. doi: 10.1109/NOMS.2006.1687677.

[23] Roditty L., Zwick U.: Improved dynamic reachability algorithms for directed graphs. In: *The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proceedings*, pp. 679–688, 2002. doi: 10.1109/SFCS.2002.1181993.

[24] Roditty L., Zwick U.: A Fully Dynamic Reachability Algorithm for Directed Graphs with an Almost Linear Update Time, *SIAM Journal on Computing*, vol. 45(3), pp. 712–733, 2016. doi: 10.1137/13093618X.

[25] Schwagereit F., Scherp A., Staab S.: Representing Distributed Groups with dgFOAF. In: *The Semantic Web: Research and Applications*, vol. 6089, pp. 181–195, 2010.

[26] Seufert S., Anand A., Bedathur S., Weikum G.: FERRARI: Flexible and efficient reachability range assignment for graph indexing. In: *2013 IEEE 29th International Conference on Data Engineering (ICDE)*, pp. 1009–1020, 2013.

[27] Shiny S., Jasper J.: Decentralized access control technique with multi-tier authentication of user for cloud storage, *Peer-to-Peer Networking and Applications*, pp. 1–15, 2021.

[28] Svirskas A., Ignatiadis I., Roberts B., Wilson M.: Virtual Organization Management Using Web Service Choreography And Software Agents. In: *Network-Centric Collaboration and Supporting Frameworks*, pp. 535–542, Springer US, Boston, MA, 2006.

[29] Tošić A., Vičič J.: A Decentralized Authoritative Multiplayer Architecture for Games on the Edge, *Computing and Informatics*, vol. 40(3), pp. 522–542, 2021.

[30] Travica B.: The Design of the Virtual Organization: A Research Model. In: *AMCIS 1997 Proceedings*, 1997.

[31] Viet Dung D.: *Coalition Formation and Operation in Virtual Organisations*, Ph.D. thesis, University of Southampton, School of Electronics and Computer Science, 2004. https://eprints.soton.ac.uk/260239/.

[32] Wognum N., Faber E.C.C.: Infrastructures for collaboration in virtual organisations, *International Journal of Networking and Virtual Organisations*, vol. 1(1), pp. 32–54, 2002. doi: 10.1504/IJNVO.2002.001462.

[33] Wrzeszcz M., Dutka Ł., Słota R.G., Kitowski J.: New approach to global data access in computational infrastructures, *Future Generation Computer Systems*, vol. 125, pp. 575–589, 2021. doi: https://doi.org/10.1016/j.future.2021.06.054.

[34] Xhafa F., Poulovassilis A.: Requirements for Distributed Event-Based Awareness in P2P Groupware Systems. In: *24th IEEE International Conference on Advanced Information Networking and Applications Workshops*, pp. 220–225, 2010.

[35] Yeung C.A., Liccardi I., Lu K., Seneviratne O., Berners-Lee T.: Decentralization: The future of online social networking. In: *W3C Workshop on the Future of Social Networking Position Papers*, pp. 2–7, 2009.

Affiliations

Łukasz Opioła
AGH University of Science and Technology, Academic Computer Centre CYFRONET AGH, Krakow, Poland, lopiola@agh.edu.pl

Kamil Jarosz
AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications, Institute of Computer Science, Krakow, Poland, kjarosz@agh.edu.pl

Łukasz Dutka
AGH University of Science and Technology, Academic Computer Centre CYFRONET AGH, Krakow, Poland, lukasz.dutka@cyfronet.pl

Renata G. Słota
AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications, Institute of Computer Science, Krakow, Poland, renata.slota@agh.edu.pl

Jacek Kitowski
AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications, Institute of Computer Science, Krakow, Poland,
AGH University of Science and Technology, Academic Computer Centre CYFRONET AGH, Krakow, Poland, jacek.kitowski@agh.edu.pl

Received: 28.12.2021
Revised: 30.05.2022
Accepted: 15.06.2022