Data Science and Engineering (2026) 11:343–359
https://doi.org/10.1007/s41019-025-00320-y

RESEARCH PAPERS

Relationship-based Access Control for Data Spaces

Nikos Fotiou[1] · Chalima Dimitra Nassar Kyriakidou[1] · Athanasia Maria Papathanasiou[1] · Vasilios Siris[1,2] · George Polyzos[1,3]

Received: 3 January 2025 / Revised: 2 September 2025 / Accepted: 13 September 2025 / Published online: 29 January 2026

Abstract
Data spaces are an emerging concept with significant potential to enable a data-centric economy by fostering seamless and secure data sharing across diverse stakeholders. These environments are designed to unlock the value of data by ensuring interoperability and collaboration, which are essential for innovation and informed decision-making. However, managing access control in data spaces poses unique challenges, as it must account for complex relationships not only among stakeholders but also among data items themselves, requiring a flexible and context-aware approach. To this end, in this paper we present the design, implementation, and evaluation of an access control solution tailored for data spaces. Our solution leverages the paradigm of Relationship-Based Access Control (ReBAC), enabling the definition and enforcement of access control policies that consider the relationships between entities within the data space, as well as data consumer organisational structures. Furthermore, we propose a distributed version of our solution to facilitate the segregation of access control management across different administrative domains. Our approach supports fine-grained, continuous access control by dynamically evaluating the context of both the protected data items and the consumers of the data space. To ensure compatibility with existing data-sharing standards, we have integrated our solution with ETSI NGSI-LD API, a standardised interface for interacting with data spaces.

Keywords ReBAC · FIWARE · OpenFGA · Usage control

1 Introduction

Data spaces represent a transformative approach to digital ecosystems, providing a controlled environment for secure and efficient data sharing. These platforms aim to facilitate data-driven innovations and drive digital transformation across industries and domains [1]. Various initiatives around the world, including the International Data Spaces Association (IDSA), GAIA-X, and FIWARE, have been established to promote the development and adoption of data spaces. These initiatives focus on creating standardized frameworks and architectures to enable seamless data exchange while preserving data sovereignty and fostering trust among stakeholders.

A data space comprises essential building blocks that ensure semantic interoperability of data, uniform access methods, and robust mechanisms for maintaining trust. Data sovereignty–the ability of data owners to retain control over their data–is a cornerstone of these platforms, making them particularly appealing in a world increasingly concerned about privacy, security, and regulatory compliance. At the

Nikos Fotiou, Chalima Dimitra Nassar Kyriakidou, Athanasia Maria Papathanasiou, Vasilios Siris and George Polyzos contributed equally to this work.

✉ Nikos Fotiou
fotiou@excid.io

Chalima Dimitra Nassar Kyriakidou
dnassar@aueb.gr

Athanasia Maria Papathanasiou
sissypapathanasiou@aueb.gr

Vasilios Siris
siris@excid.io

George Polyzos
polyzos@excid.io

[1] ExciD, 11362 Athens, Greece

[2] Department of Informatics, Athens University of Economics and Business, 10434 Athens, Greece

[3] School of Data Science, The Chinese University of Hong Kong, 518172 Shenzhen, Guangdong, China

same time, data users require assurance about the quality and authenticity of the data they access to protect against potential risks associated with low-quality or malicious data [2].

Despite these advancements, one of the critical challenges in the realization of data spaces is the enforcement of effective data usage policies and access control [3, 4]. Advanced access control mechanisms are required to accommodate complex relationships among stakeholders and to ensure that both data owners and users can engage with confidence. Addressing these challenges is crucial for unlocking the full potential of data spaces as enablers of a secure and data-centric economy. To better illustrate the access control requirements of data spaces, we present the following data sharing use case that involves multiple stakeholders.

1.1 A Data Sharing Use Case

A **Smart City** has deployed a wide range of IoT devices, including security cameras, smart plugs, environmental sensors, and waste monitoring equipment. Data generated by these devices can be accessed through a data space, which facilitates advanced data queries via an interoperable and standardized data access API. This infrastructure enables trusted data sharing across organizational boundaries.

To support collaborative research, Smart City signs a memorandum of understanding with the **Technical University**, granting members of its **Computer Science Department read** access to **energy consumption data** generated by the **smart plugs** installed in the **City Hall**. Access is mediated through the data space's infrastructure, which offers an API for both data discovery and policy-based access enforcement.

In such environments, access control policy systems must address a number of challenges arising from complex and dynamic relationships between entities. Policies must take into account contextual factors such as the affiliation of a user with a particular organization, the spatial and functional characteristics of the devices that produce the data, and the structure or grouping of resources. For example, a policy might need to permit researchers from institutions that have formal collaborations with Smart City to access data generated by smart plugs located in the City Hall, or to grant access to all data produced by environmental sensors managed by a specific department. These requirements highlight the need to express access decisions not only based on organizational ties but also based on location-based conditions, and hierarchically grouped resources.

Moreover, several operational requirements must be addressed by an effective access control policy system. First, there must be **automatic access right assignment**: e.g., when a new smart plug is installed in the City Hall, researchers should immediately be granted access to its energy measurements without requiring manual policy updates. Second, **automatic access right revocation** must be supported: e.g., if a smart plug is removed or relocated outside the City Hall, access to its data should be revoked without any additional configuration. Finally, there must be **segregation of policy administration systems**: e.g., the Technical University should be able to independently manage its identity and access management (IAM) system, defining its users and their roles, while the Smart City should maintain its own IAM system responsible for defining devices and their associated metadata and relationships. These requirements reflect a need for decentralized, relationship-aware policy systems that can adapt dynamically to the evolving structure of the data space.

Traditional access control models such as role-based access control (RBAC) and attribute-based access control (ABAC) face limitations when it comes to supporting automatic assignment and revocation of access rights in dynamic environments like data spaces. Both RBAC and ABAC are typically built around static policy models that do not naturally incorporate the evolving, relationship-driven nature of access rights in data spaces. This makes it challenging to design policies that can respond dynamically to real-world events without significant manual intervention. In contrast, a relationship-based approach enables access control decisions to reflect the current graph of associations among users, devices, and organizations, allowing access rights to be granted or revoked automatically as relationships change—without requiring explicit policy rewrites or attribute reconfigurations.

1.2 Contributions and Paper Structure

To address the access control requirements of our defined data-sharing use case, we employ Relationship-Based Access Control (ReBAC), a novel and versatile access control paradigm. ReBAC enables the explicit definition of relationships among protected items and facilitates the creation of access control policies that leverage these relationships. This paradigm offers significant flexibility and supports fine-grained control, making it well-suited for the complex and dynamic nature of data spaces. With this paper, we make the following contributions:

- We design and integrate our ReBAC-based solution into a data space implementation that adheres to ETSI's NGSI-LD API, a standardized interface for interacting with data spaces, ensuring compatibility and ease of adoption.
- Additionally, we provide a distributed version of our solution, which supports *segregation of policy*

Table 1 Relation tuples examples

Relation tuples	Semantics
user:Alice, member, group:family	User *Alice* is a member of the user group *family*
folder:shared, parent, doc:calendar	Doc file *calendar* is in folder *shared*
group:family#member, can_read, folder:shared	Members the user group *family* have read access to folder *shared*

administration systems, enabling decentralized management of access control policies across multiple administrative domains.
- Finally, we perform and report on a security evaluation and an extensive experimental performance evaluation of the approach, demonstrating its practical significance.

In the remainder of this paper, we introduce ReBAC, Data Spaces, and the ETSI NGSI-LD API in Sect. 2, detail our solution in Sect. 3, and evaluate it in Sect. 4. We discuss related work in this area in Sect. 5 and we provide a summary and our conclusions in Sect. 6.

2 Background

2.1 Relationship-based Access Control

Relationship-Based Access Control (ReBAC) is an access control paradigm that allows access control decisions based on relationships. Our solution relies on an implementation of the Zanzibar [5] ReBAC system.

The main building block of Zanzibar is an *authorization model* that specifies the *types* of entities and the corresponding *relations* per type, supported by an authorization system. Listing 1 is an example of an authorization model created using OpenFGA's[1] Domain Specific Language (DSL). In this example four types of entities have been defined ("user","group","folder", and "file"). Furthermore, for some types, relations have been defined (for example, the "file" type includes the relations "parent" and "can_read").

An entity is represented using the notation `type:identifier` (e.g., "user:Alice", "device:plug1"). A memberset of an entity's relation is represented using the notation `type:identifier#relation` (e.g., "device:plug1#owner", which represents the set of entities that have "owner" relation with "device:plug1")

Authorization system *administrators* can then provide *relation tuples*. Relation *tuples* are of the form [*subject*, *relation*, *object*] where *subject* may be either an entity or the memberset of an entity's relation, *relation* is an arbitrary string, and *object* is another entity. Table 1 includes examples of relation tuples and their semantics.

For each relation, *rules* are defined in an *authorization model* that specify how the relation's *memberset* is created. Specifically, a memberset can be created by the union or intersection of the following membersets:

- The memberset is created by combining the subjects of all tuples provided by the system administrator. This is indicated using the [] notation. For example, line 5 of Listing 1 defines that the memberset of the relation "member" is created by combining the subjects of all tuples provided by the system administrator, which should be of type "user".
- A memberset *inherited* from a relation of the same type. For example, line 10 of Listing 1 defines that the memberset of the "can_read" relation automatically includes the memberset of the "can_write" relation. This rule implies that entities that are authorized to write to a folder are also authorized to read from this folder.
- A memberset can be defined to *inherit* from another type that has a relationship with the current type. For example, line 16 of Listing 1 defines that the memberset of the "can_read" relation automatically includes the memberset of the "can_read" relation of entities that have "parent" relation with the current entity. This rule implies that entities that are authorized to read a folder are also authorized to read the files included in this folder.

Listing 1 An example of authorization model defined using OpenFGA DSL.

```
1  type user
2
3  type group
4    relations
5      define member:[user]
6
7  type folder
8    relations
9      define can_write: [user, group#member]
10     define can_read: [user, group#member] or can_write
11     define parent: [folder]
12
13 type file
14   relations
15     define parent: [folder]
16     define can_write: [user, group#member] or can_write from parent
17     define can_read: [user, group#member] or can_read from parent
           or can_write
```

Authorization checks in Zanzibar take the form of queries, such as "does subject U have relation R to object O?". In our example, the query "Does `user:Alice` has `can_read` relation with `doc:calendar`" will evaluate to *True*, since `user:Alice` has relation `member` with `group:family`, `group:family` has relation `can_read` with `folder:shared`, and `folder:shared` has relation `parent` with `doc:calendar`.

[1] https://openfga.dev.

Table 2 Notation used in this section

Notation	Semantics
$\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$	The set of types of a data space
$\mathcal{T}' = \{T'_1, T'_2, \ldots, T'_n\}$	The set of types of an authorization model
$r_j^{(i)}$	A relationship attribute of data space type T_i
$p_j^{(i)}$	A property attribute of data space type T_i
$r_j'^{(i)}$	A relation of authorization model type T'_i
$\vec{P}_{T_i} = \{p_1^{(i)}, p_2^{(i)}, \ldots, p_{m_i}^{(i)}\}$	The set of property attributes of data space type T_i
$\vec{R}_{T_i} = \{r_1^{(i)}, r_2^{(i)}, \ldots, r_{k_i}^{(i)}\}$	The set of interrelationship attributes of data space type T_i

2.2 Data Spaces and the NGSI-LD API

A data space includes *entities*. An entity is in many cases a digital representation of a real-world asset. Each entity is uniquely identified by a URI, belongs to a well-known *type*–also uniquely identified by a URI–and has several *attributes*. Each such attribute may correspond to a *property* of the entity or to a *relationship* with another entity. Accordingly, a type can be associated with many objects and an object may have multiple attributes.

A data space is composed of several key components that enable the secure and efficient sharing of data among multiple stakeholders. Central to a data space is the *context broker*, which acts as the repository where entities are stored and serves as the core platform for managing data exchange. The *context broker* is administrated by a data *intermediary*. Each entity within the data space is associated with a specific *owner*, who holds the rights to the entity and controls its usage. Data associated with an entity is provided by a data *provider*, which is typically part of the same administrative domain as the *owner*. Data *consumers*, on the other hand, are stakeholders who access the data associated with entities in the data space. They may retrieve attributes of specific entities or access data across all entities of a particular type, often using criteria to filter and tailor the data they obtain. Moreover, consumers can subscribe to updates, enabling them to receive notifications when attributes of specific entities are modified.

The ETSI standardized *Next Generation Service Interfaces Linked Data* (NGSI-LD) API [6] provides a comprehensive specification for executing data space operations using HTTP. This API defines a set of endpoints and operations that enable management and interaction with entities within a data space. To create new entities, an HTTP POST request is made to the */entities* endpoint of the context broker that includes the new entity. Entities are serialized in JSON-LD, a format designed for representing linked data, which ensures that the data is both human-readable and machine-interpretable. Once an entity is created, a dedicated endpoint is automatically established under */entities*, scoped specifically for that entity. At this endpoint, HTTP PUT operations can be performed to modify the attributes of the entity, enabling updates to its state or metadata. Similarly, HTTP GET operations allow users to read the current attributes of the entity, providing an efficient way to retrieve information.

3 Design

3.1 Preliminaries

In the rest of this paper, in order to distinguish between a type defined by a ReBAC authorization model and a data space type, we refer to the former as *authorization type* and to the latter as *data space type*.

Our solution considers that data providers and consumers belong to an *organization* (e.g., *Smart City*, the *Technical University* from our use case). Organizations may use structures to group their users (e.g., *researchers*). A provider may perform any of the *read*, *write*, or *subscribe* operations defined in the previous section, whereas a consumer may perform only the *read* and *subscribe* operations.

Authorization models are created by an *administrator*. Initially, we consider that an authorization model is stored in a centralized location where organizations and data owners can provide relation tuples. Then, we present a version of our solution where the authorization model and the corresponding tuples are managed in a distributed manner.

In the rest of this section, we use the following notation. A data space is defined as a finite set of types:

$$\mathcal{T} = \{T_1, T_2, \ldots, T_n\}$$

Each type $T_i \in \mathcal{T}$ is associated with a finite set of **property attributes**:

$$\vec{P}_{T_i} = \{p_1^{(i)}, p_2^{(i)}, \ldots, p_{m_i}^{(i)}\}$$

where each $p_j^{(i)}$ is a property attribute of type T_i. Similarly, each type $T_i \in \mathcal{T}$ is associated with a finite set of **relationship attributes**:

$$\vec{R}_{T_i} = \{r_1^{(i)}, r_2^{(i)}, \ldots, r_{k_i}^{(i)}\}$$

where each $r_j^{(i)}$ is a relationship attribute of type T_i. Finally, we define function

$$\text{obj} : \bigcup_{T_i \in \mathcal{T}} \vec{R}_{T_i} \rightarrow \mathcal{T}$$

that maps a relationship property to the type of its object. For example, consider the relationship "container" defined in line 12 of Listing 2; let the type T of "urn:building:building1" be "building", then $\text{obj}(\text{“container”}) = \text{“building”}$.

3.2 Reference Data Space

In the rest of this section we consider the data space described in Sect. 1.1. In the considered data space, entities of the same type have the same attributes. Additionally, entities may have relations with each other. In order to be compatible with the ReBAC terminology, we refer to the two parts of a relationship as the *subject* and the *object*.

Entity identifiers in our data space follow the format urn: < type >:< id >, for example, `urn:plug:plug1`. In accordance with the ETSI NGSI-LD specifications, entities are serialized using JSON-LD, where each attribute is represented as a JSON object. Attributes that describe properties are encoded with two fields: *type*, set to "Property", and *value*, which holds the actual value of the property. Similarly, attributes that define relationships are represented by JSON objects with the field *type* set to "Relationship", and the field *object* specifying the identifier of the related entity.

Listing 2 is an example of an entity of type `plug`. This entity includes two attributes representing properties (i.e., `consumption` and `output`) and an attribute representing a relationship (i.e., `container`).

Listing 2 An example of an object of type *plug* defined using JSON-LD following ETSI NGSI-LD specification.

```
1  {
2      "id": "urn:plug:plug1",
3      "type": "plug",
4      "consumption": {
5          "type": "Property",
6          "value": "0"
7      },
8      "output": {
9          "type": "Property",
10         "value": "220V"
11     },
12     "container": {
13         "type": "Relationship",
14         "object": "urn:building:building1"
15     },
16     "@context": "https://example.com/dataspace.jsonld"
17 }
```

3.3 Authorization Model

Our authorization model considers two classes of authorization types: authorization types related to users and authorization types related to data space entities.

3.3.1 Authorization Types Related to Users

With respect to users, our authorization model includes two authorization types: `user` and `group`. The latter type is used for organizing users into user groups and for this reason, it includes a relation named `member`; the memberset of this relation is created using relation tuples provided by the administrator and is of type "user".

3.3.2 Authorization Types Related to Data Space Entities

For each data space type T_i that can exist in our data space, we define a new authorization type T_i' in our authorization model. For example, in our use case, for the data space type `plug` (defined in Listing 2) the corresponding authorization type has been defined in line 23 of Listing 3.

For each relationship property $r_j^{(i)} \in \vec{R}_{T_i}$ we define a relation $r_j'^{(i)}$ in the corresponding authorization model type T_i'. The memberset of each such relation is created using relation tuples provided by the administrator. The subject of these tuples should include an entity of type T' that corresponds to the data space type $obj(r_j^{(i)})$. For example, as seen in line 12 of Listing 2, the data space type `plug` has a relationship attribute named `container` with an object of type `building`, hence in the authorization model a similar relation is defined (line 26 of Listing 3).

For each attribute property $p_j^{(i)} \in \vec{P}_{T_i}$ we define three relations in the corresponding authorization model type T_i': one that represents the *read* right, another that represents

the *write* right, and a third one that represents the *subscribe* right. These relations are named using the following naming convention: $< accessright\_T'_i\_p_j^{(i)}$ (e.g., "read_plug_consumption"). The memberset of each such relation is provided by the administrator and is of type "user" or "group#member". Furthermore, we define an implicit "special purpose" attribute property named "all", which represents all attributes of an entity, e.g., a user that has *read_plug_all* relationship with entity (of type plug) can read all its attributes.

In the next step we define rules for memberset inheritance. Let $r_j^{(i)}$ be a relationship property of T_i, $T_o = obj(r_j^{(i)})$, and T'_i, T'_o, and $r_j^{\prime(i)}$ is the corresponding authorization types and relation in the authorization model. We define in T'_o the same read, write, and subscribe relations as in T'_i. Then, we define rules in the read, write, and subscribe relations of T'_i for inheriting the corresponding membersets of $r_j^{\prime(i)}$. For example, in the authorization model of Listing 3, we have defined the relation "read_plug_consumption" in both `plug` and `building` types since they are related with relation `container`. Furthermore, we have configured "read_plug_consumption" memberset of `plug` to inherit the corresponding memberset of the `container` relation. The semantics of this inheritance is that a user (or user group) that has "read_plug_consumption" relationship with a `building` has automatically "read_plug_consumption" with all `plugs` located in this building.

Finally, in order to cater for initial object creation in a data space, as well as type-wide access rights, we define an authorization model type named *data_space*. For each data space type *T* that may exist in the data space we define a relation using the naming convention $< d\_write$ (e.g., $plug\_write$. The memberset of each such relation is created using relation tuples provided by the administrator and is of type "user" or "group#member". A user that has such a relation with the data space is allowed to create new objects of type *T*. Similarly, for all attribute properties $p_j^{(i)} \in \vec{P}_{T_i}$ we define a similar property in the *data_space* type. Similarly, the memberset of each such relation is created using relation tuples provided by the administrator and is of type "user" or "group#member": a user that has such a relation with the data space has the same relationship with all corresponding objects (e.g., a user that has `read_plug_consumption` with the data space can read the consumption of objects in data space of type plug).

Listing 3 An instance of our authorization model defined uinsg OpenFGA DSL.

```
type user

type group
  relations
    define member: [user]

type data_space
  relations
    define building_write: [user, group#member]
    define plug_write: [user, group#member]
    define read_plug_all:[user, group#member]
    define write_plug_all:[user, group#member]
    define read_plug_consumption:[user, group#member]
    define write_plug_consumption:[user, group#member]

type building
  relations
    define read_plug_all:[user, group#member]
    define write_plug_all:[user, group#member]
    define read_plug_consumption:[user, group#member]
    define write_plug_consumption:[user, group#member]

type plug
  relations
    define data_space:[data_space]
    define container:[building]
    define read_plug_all:[user, group#member] or read_plug_all from
        container or read_plug_all from data_space
    define write_plug_all:[user, group#member] or write_plug_all
        from container or write_plug_all from data_space
    define read_plug_consumption:[user, group#member] or
        read_plug_all or read_plug_consumption from container or
        read_plug_consumption from data_space
    define write_plug_consumption:[user, group#member] or
        write_plug_all or write_plug_consumption from container or
        write_plug_consumption from data_space
```

In Sect. 3.7 we detail how our authorization model is managed.

3.4 Access Control Components

Our access control solution consists of the following components:

- *Identity Provider (IdP)* A registry of user identifiers. This registry is maintained by the corresponding organization, or by a 3^{rd} party trusted by the organization.
- *Policy Administration Point (PAP)* A component responsible for managing the authorization model and the corresponding relationship tuples that define access policies. In our architecture, the PAP is maintained by a trusted party on behalf of the data owner. It serves as the authoritative registry for defining and updating access control rules, including the types, relationships, and permissions applicable to data space entities. Importantly, a data space may consist of multiple autonomous data owners, each retaining control over its own resources and associated access policies. Our design explicitly supports this heterogeneity: there is no requirement for global agreement on a single authorization model. Instead, each data owner can independently define and

manage its own model within its respective PAP instance. During policy evaluation, the correct authorization model is dynamically selected based on the ownership of the target resource. This approach enhances scalability and flexibility, while preserving each participant's control over its data governance logic.

- *Policy Enforcement Point (PEP)* A transparent HTTP proxy that intercepts API calls from consumers to the broker. If a request originates from a consumer with the appropriate access rights, it is forwarded to the broker, otherwise it is rejected. A PEP is administrated by an entity trusted by the data owner.
- *Policy Decision Point (PDP)* A component responsible for evaluating whether an incoming request is authorized based on the applicable access control policies. The PDP extracts the target resource and user-related attributes from the request and formulates an authorization query using the semantics defined in the PAP's authorization model. It is administered by an entity trusted by the data owner.

From a high level perspective, authorization using our solution is implemented as follows (see also Fig. 1). The IdP is configured with user identifiers, e.g., through a user registration process. Similarly, a data owner stores an authorization model in the PAP. Data owners and organizations provide the appropriate relation tuples. A consumer initially identifies itself to the IdP and receives an *identity token* (step 1), i.e., a token signed by the IdP that includes a proof of the consumer's identifier. Then it makes an NGSI-LD API call, including the received token in an HTTP header (step 2). The request is intercepted by the PEP, which forwards it to the PDP that makes an access control decision (step 3).

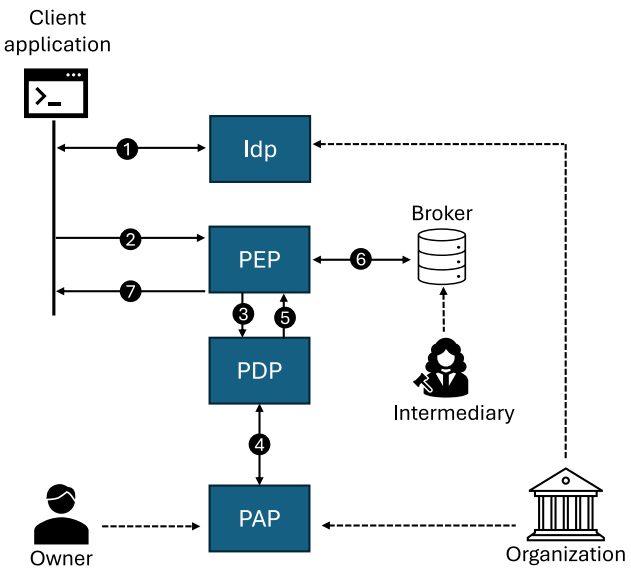

Fig. 1 High level overview of the authorization process

The PDP queries the PAP and learns if the consumer has an appropriate relation with the corresponding entity (step 4). For example, if consumer `Alice` requests to *read* the `consumption` attribute of `plug1`, the PDP will query the PAP if *Alice has read_pulg_consumption with plug1*. Using the retrieved response, the PDP makes an access control decision which it forwards to the PEP (step 5). Finally, if the consumer is authorized, the PEP forwards the request to the broker (step 6) and relays its response back to the consumer (step 7).

3.5 Distributed Access Control

We now present a distributed version of our solution that enables organizations and data owners to independently manage the relationship tuples relevant to its own users or data space entities. This separation of concerns improves scalability, enables local autonomy, and enhances data sovereignty and trust boundaries.

3.5.1 Decentralized Relationship Management

In this architecture, each organization is responsible for managing identity information and access rights for its users. This includes issuing identity tokens via a trusted Identity Provider (IdP) and optionally maintaining its own PAP that holds relationship tuples involving those users.

Data owners, on the other hand, are solely responsible for managing the relationship tuples that pertain to the data space entities they control. This clean separation allows each stakeholder to retain administrative control over its own resources, without requiring a globally agreed set of tuples or a centralized access management authority.

3.5.2 Tuple Creation: Implicit vs. Explicit

Relationship tuples about users can be provisioned in two ways:

- **Implicit Tuple Embedding.** In this approach, relationship information is embedded directly in the identity token (e.g., as custom claims in a JWT). When the token is presentedby the user, the PDP extracts and uses this information during access evaluation. While simple and self-contained, this method can lead to oversized tokens and raises security concerns. For instance, this approach may result in embedding sensitive access information directly in tokens, which may violate the principle of least disclosure or complicate token revocation.
- **Explicit Tuple Exchange.** A more modular and privacy-preserving approach involves using the OAuth 2.0 Token Exchange protocol [7]. Here, the identity token

serves as an input to an authenticated query to the organization's PAP. The PAP returns a signed security token containing only the relevant relationship tuples. This approach allows finer-grained control, easier rotation of access rights, and better compliance with privacy constraints.

3.5.3 OAuth 2.0 Token Exchange and Tuple Retrieval

As depicted in Fig. 2, the token exchange protocol proceeds as follows: the consumer first authenticates with its organization's IdP (e.g., via OpenID Connect) and obtains an *identity token* that includes the consumer's identifier $Consumer_{Id}$. The consumer then invokes a resource in the data space, presenting this token to the Policy Enforcement Point (PEP).

The PDP identifies the corresponding organization from the token's claims and performs a token exchange by sending the identity token to the organization's PAP. The PAP validates the token, extracts the consumer identity, and issues a *security token*–a signed JSON document that contains the user's relationship tuples (e.g., group memberships, assigned roles, etc.). These tuples are then used by the PDP to construct the access query to the owner's PAP.

3.5.4 Proof of Possession and Token Integrity

To prevent unauthorized use or sharing of identity tokens, we incorporate *proof-of-possession* mechanisms. Specifically, we adopt Demonstrating Proof of Possession (DPoP) [8], which cryptographically binds an identity token to a public/private key pair controlled by the consumer.

Each identity token includes a hash of the consumer's public key in its claims. When interacting with the data space, the consumer must present both the identity token and a DPoP proof: a digital signature computed over selected fields, including the HTTP method, request URI, a timestamp, and a random nonce. The PEP validates this

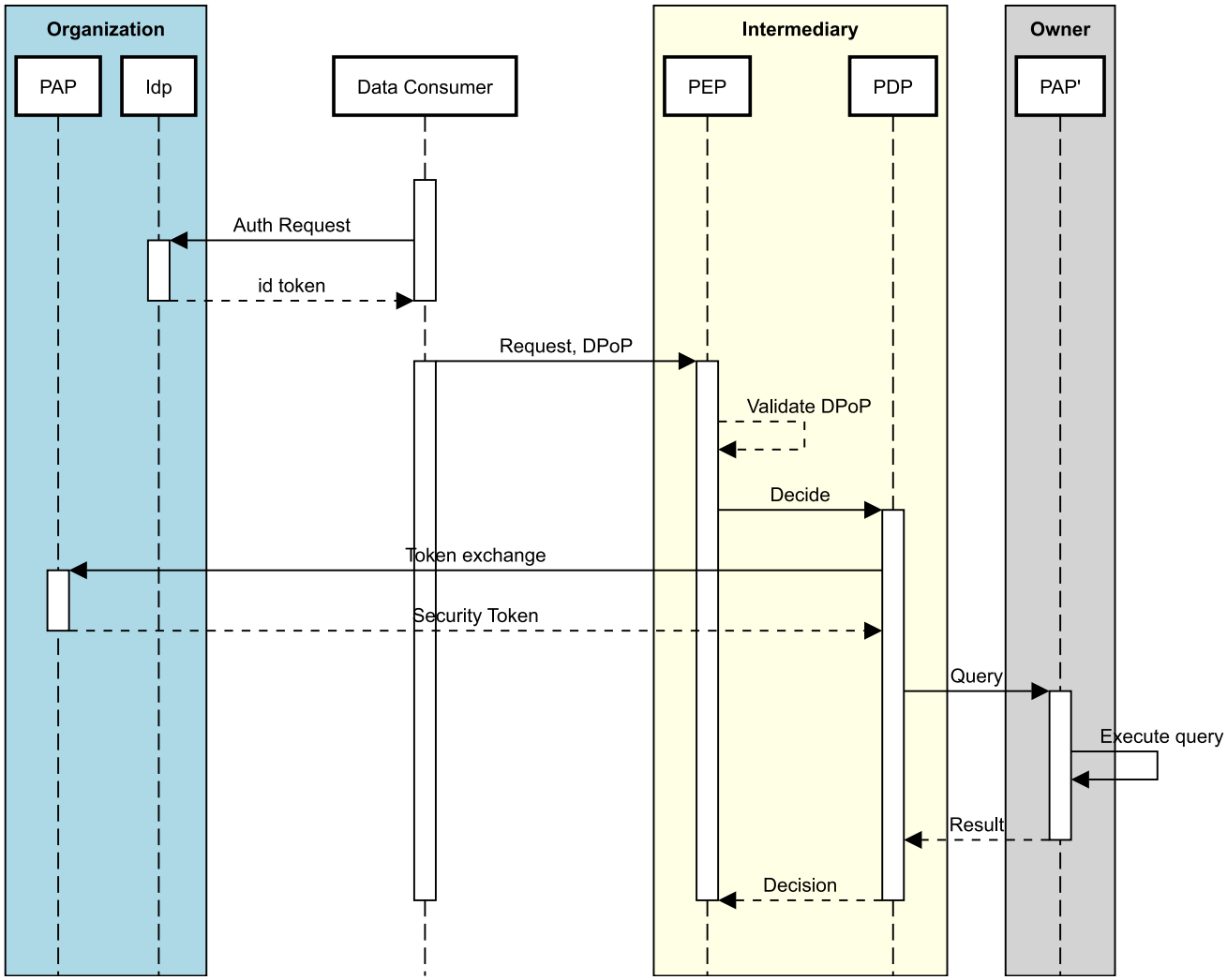

Fig. 2 Authorization using OAuth 2.0 Token exchange

proof using the public key hash in the token, thereby confirming that the requestor holds the corresponding private key. This prevents replay and misuse of identity tokens by unauthorized parties.

Alternative proof-of-possession schemes, such as OpenPubkey [9], can also be used in this architecture without affecting the overall protocol flow.

3.5.5 Policy Decision Process

Once a PDP has obtained the relevant relationship tuples about the user–either through implicit claims in the identity token or via explicit token exchange–it proceeds to evaluate the user's access rights over the requested resource.

Rather than permanently storing these user-specific relationships in the data space PAP, our system leverages a feature of OpenFGA called *contextual queries*. This mechanism allows the PDP to *temporarily augment* the authorization model with *ephemeral relationships*–relationship tuples that are not persisted but are valid only for the scope of the current access decision. These ephemeral tuples are injected into the query at runtime, enabling access control decisions without modifying the underlying persistent state.

To reduce latency and avoid repeated round-trips to the organization's PAP, the PDP may cache these ephemeral relationships locally. The duration for which such cached tuples remain valid is determined by the issuing organization through the use of token lifetimes. Specifically, the organization can set an expiration time on the identity or security token, which the PDP uses as the upper bound for caching and reuse of the associated ephemeral data.

3.6 Continuous Access Enforcement

In dynamic and decentralized environments such as data spaces, access control cannot be limited to the moment of the initial request. Instead, there are cases where a data consumer continues to receive access to a resource over an extended period of time. One such example is the use of the *Subscription* mechanism provided by the ETSI NGSI-LD API.

Using this mechanism, a consumer subscribes to updates about specific data space entities by issuing a `POST` request to the NGSI-LD `/subscriptions` endpoint. This request includes a callback URL (also referred to as a *notification endpoint*) and a filter specifying the entities and attributes of interest. Once the subscription is registered, the consumer receives asynchronous notifications whenever the specified entity is updated. Importantly, these notifications continue to be delivered as long as the subscription remains active.

This pattern introduces a challenge: once a data consumer is authorized and a subscription is registered, the consumer may continue to receive updates even if their access rights change in the meantime. In other words, authorization is only checked at the time of subscription creation, not during the lifecycle of the subscription itself. To address this, our architecture can be extended to support *continued access enforcement*. Specifically, we envision a function—deployed alongside the Policy Decision Point (PDP) that periodically re-evaluates the consumer's access rights, e.g., by refreshing the security token previously obtained from the organization's PAP. If the new evaluation indicates that the consumer is no longer authorized to access the subscribed entity, the function can revoke access by canceling the subscription. This can be achieved using the NGSI-LD `DELETE` operation on the corresponding subscription resource.

This mechanism would allow our architecture to enforce access revocation dynamically and securely, aligning with principles of *usage control* and continuous authorization. At the time of writing, the implementation and evaluation of this function is left as future work, but it represents an important step toward building secure and responsive access control mechanisms in real-time data-sharing ecosystems.

3.7 Relationships Management

In our system, we distinguish between two categories of relationships within the authorization model:

1. Relationships between users (or groups of users) and data space entities, that encode access control policies.
2. Relationships among data space entities, that reflect the structural organization of the data space itself.

Relationships between users and data space entities are managed through a centralized Policy Administration Point (PAP). These relationships directly encode access permissions and support both fine-grained and coarse-grained authorization policies. Fine-grained policies may grant access to specific attributes of a single entity (e.g., a plug), while coarse-grained policies can apply to broader scopes, such as entire buildings or the entire data space. Table 3 illustrates representative examples of access control policies

Table 3 Examples of access control policies and their corresponding relationship tuples

Policy	Tuple
User Alice can read the consumption of plug1	user:Alice, read_plug_consumption, plug1
User Alice can read the consumption of all plugs	user:Alice, read_plug_consumption, data_space
User Alice can read all attributes of all plugs located in building 1	user:Alice, read_plug_all, building1

and their corresponding relationship tuples stored in the authorization engine.

Relationships among data space entities—such as the association of a plug to a building—are managed automatically through integration with the ETSI NGSI-LD API. These relationships reflect the dynamic topology of the data space and are updated based on intercepted HTTP requests that modify the data space:

- **POST requests:** When a new entity is created in the data space (e.g., a new plug or building), and the entity includes a relationship attribute (e.g., a plug belonging to a building), a corresponding relationship tuple is added to the authorization model.
- **PATCH requests:** When an existing entity is updated and the update involves modifying a relationship attribute (e.g., transferring a plug from one building to another), the old relationship tuple is removed from the authorization model and a new one is inserted to reflect the updated linkage.
- **DELETE requests:** When an entity is deleted (e.g., an IoT device is removed), all associated relationship tuples are identified and purged from the authorization model to maintain consistency.

This seamless integration with the NGSI-LD API allows automatic and consistent management of structural relationships without human intervention. It is particularly beneficial in dynamic environments, where data space changes frequently. As a result, administrators are relieved from the burden of manually maintaining entity-to-entity relationships in the authorization model, ensuring both scalability and correctness in evolving deployments.

4 Implementation and Evaluation

We implemented a prototype of our data space, which is used for sharing data collected by IoT devices installed in real smart buildings. Data are serialized using JSON-LD, and stored in a context broker of a data intermediary implemented using FIWARE Orion [10].

As an IdP we used Keycloak.[2] A PAP has also been implemented using OpenFGA[3]. The corresponding PDP has been implemented as a custom application. Finally, for the PEP we rely on Ory Oathkeeper.[4]

[2] https://www.keycloak.org/

[3] https://openfga.dev.

[4] https://www.ory.sh/oathkeeper/.

4.1 Security Evaluation

We now evaluate the security of our distributed ReBAC-based access control solution using a structured approach. In particular, we first define the system's trust assumptions and threat model, and then we analyze its security properties with respect to authorization correctness, resistance to impersonation, privacy, availability, and revocation consistency.

4.1.1 Threat Model and Assumptions

In our model, we assume trusted IdPs that issue identity tokens correctly and trusted PAPs that provide correct relation tuples. Legitimate intermediaries are expected to correctly enforce access control decisions, in accordance with the authorization model. Moreover, all communications between components are assumed to be conducted over secure and authenticated channels, to ensure confidentiality and integrity. Consumers are also assumed to securely manage their own private keys, which are used to generate proofs of possession via DPoP or OpenPubkey. Finally, we rely on the correctness and security of standard cryptographic primitives, including the unforgeability of the underlying digital signatures and the resistance of DPoP proofs to forgery or replay attacks.

We consider an adversary whose primary objective is to gain unauthorized access to protected data items in the data space. The adversary may operate either as an external party attempting to impersonate a legitimate user (*impersonator*) or as a *malicious intermediary* trying to gain access to a data space provided by another (legitimate) intermediary. Therefore, we define the following authorization predicate:

$$\mathrm{Auth}(C, I, o) = \mathrm{true}$$

to state that consumer C is authorized by intermediary I to access object o.

For the impersonator, we consider an external adversary who acts as a legitimate consumer C, in order to gain access to an object o governed by an intermediary I. The adversary may obtain a valid identity token T_C issued to C through credential theft and has goal is to submit a request that passes the PEP and PDP checks of I and achieves

$$\mathrm{Auth}(\mathcal{M}, I, o) = \mathrm{true}, \quad \text{where } \mathcal{M} \neq C \text{ and } \neg\mathrm{Authorized}(\mathcal{M}, I, o).$$

This attack is related to the *Authorization Soundness* property, and the adversary tries to break it by impersonating a legitimate consumer. Below, we formally prove that authorization soundness holds in our system against impersonator adversaries.

In the event where the adversary operates as a malicious intermediary and her goal is to gain access to a data space provided by another (legitimate) intermediary, let $Auth(C, I, o) \rightarrow true$, if consumer C is authorized to access an object o in a data space provided by intermediary I. Additionally, let I_{leg}, I_{mal}, C_1, and o_1 be a legitimate intermediary, a malicious intermediary, a consumer, and an object, respectively. Supposedly,

$$Auth(C, I_{leg}, o) \rightarrow true$$

$$Auth(C, I_{mal}, o) \rightarrow true$$

The goal of I_{mal} is to achieve $Auth(I_{leg}, I_{leg}, o) \rightarrow true$.

This attack is related to the *Impersonation Resistance via DPoP* property and below we will formally show that our model is secure against adversaries who operate as malicious intermediaries.

4.1.2 Authorization Soundness

Statement. A consumer is granted access to a data object if and only if the following three conditions are satisfied: (i) the consumer presents a valid identity token, (ii) the request includes a valid proof-of-possession of the corresponding key, and (iii) the access request is authorized according to the intermediary's policy model and relation tuples.

Let C be a data consumer, I be an intermediary, o be a protected object in the data space, T_C be a token issued to C by a trusted IdP, K_C be the public key associated with C and cryptographically bound to T_C (e.g., via DPoP), req be the access request sent by C, $\mathrm{HasAccess}(C, I, o)$ denote that C is granted access to o by I, $\mathrm{ValidToken}(C, T_C)$ denote that T_C is a valid token for C, $\mathrm{ValidDPoP}(T_C, \mathrm{req}, K_C)$ denote that the DPoP proof in req verifies correctly against K_C, $\mathrm{Auth}(C, I, o)$ denote that C has an appropriate relationship to o according to the policy and relation tuples maintained by I.

We claim the following equivalence:

$$\forall C, I, o.\quad \mathrm{HasAccess}(C, I, o) \iff \big(\ \mathrm{ValidToken}(C, T_C) \wedge \mathrm{ValidDPoP}(T_C, \mathrm{req}, K_C)\ \wedge\ \mathrm{Authorized}(C, I, o)\big)$$

Proof. We prove both directions of the equivalence.

(Soundness: $\Rightarrow$) Assume that $\mathrm{HasAccess}(C, I, o)$ holds. That is, intermediary I has granted C access to object o. Access in our system is mediated through the PEP and the PDP. Therefore, for C to be granted access:

1. The PEP must have accepted the request, which implies that it validated T_C and verified the DPoP proof using K_C. Thus,

$$\mathrm{ValidToken}(C, T_C) \wedge \mathrm{ValidDPoP}(T_C, \mathrm{req}, K_C)$$

must both hold.
2. The PDP must have queried the PAP and determined that the policy model and relation tuples authorize C to access o. Hence,

$$\mathrm{Authorized}(C, I, o)$$

must also hold.

Therefore, all three conditions on the right-hand side hold.

(Completeness: $\Leftarrow$) Assume now that:

$$\mathrm{ValidToken}(C, T_C) \wedge \mathrm{ValidDPoP}(T_C, \mathrm{req}, K_C) \wedge \mathrm{Authorized}(C, I, o)$$

holds. Then:

1. The PEP verifies the identity token T_C and checks that the DPoP proof in req is valid with respect to the public key K_C that is bound to the token. Thus, the request is accepted at the transport and session layer.
2. The PDP evaluates $\mathrm{Authorized}(C, I, o)$ against the policy model and relation tuples in the PAP, and finds that C is authorized to access o.

Consequently, the request passes both authentication and authorization checks, and access is granted. Hence,

$$\mathrm{HasAccess}(C, I, o)$$

holds.

Proof of Authorization Soundness We now prove that a malicious user $\mathcal{M}$ who attempts to impersonate a legitimate consumer C cannot satisfy the right-hand side of the equivalence in the statement of authorization soundness. In particular, such an adversary cannot satisfy the condition $\mathrm{ValidDPoP}(T_C, \mathrm{req}, K_C)$.

Let us assume that the adversary $\mathcal{M} \neq C$ is in possession of a valid identity token T_C that was originally issued

for consumer C. This token is cryptographically bound to a public key K_C, such that T_C contains a public key fingerprint identifying K_C.

In order to pass the DPoP validation step, $\mathcal{M}$ must produce a digital signature σ over a set of request-specific data (including the HTTP method, URI, timestamp, and a nonce), using the private key k_C corresponding to K_C:

$$\sigma = \mathrm{Sign}_{k_C}(\mathrm{req\_data})$$

However, since $\mathcal{M}$ does not possess k_C, and the DPoP verification step requires:

$$\mathrm{Verify}(K_C, \sigma, \mathrm{req\_data}) = \mathrm{true}$$

$\mathcal{M}$ is unable to produce a valid DPoP proof. Therefore:

$$\neg\mathrm{ValidDPoP}(T_C, \mathrm{req}, K_C)$$

Hence, the right-hand side of the authorization soundness condition fails, and thus we conclude that:

$$\neg\mathrm{HasAccess}(\mathcal{M}, I, o)$$

4.1.3 Impersonation Resistance via DPoP

Recall that we consider an adversary that operates as a malicious intermediary and attempts to gain unauthorized access to a data space managed by another (legitimate) intermediary. Let $\mathrm{Auth}(C, I, o) \rightarrow \mathrm{true}$ denote that consumer C is authorized to access object o in a data space provided by intermediary I. Let I_{leg} and I_{mal} be a legitimate and a malicious intermediary, respectively. Let C_1 be a consumer and o_1 an object.

Suppose:

$$\mathrm{Auth}(C_1, I_{\mathrm{leg}}, o_1) = \mathrm{true}$$
$$\mathrm{Auth}(C_1, I_{\mathrm{mal}}, o_1) = \mathrm{true}$$

The goal of the adversary I_{mal} is to reuse valid access credentials (e.g., a DPoP proof created for its own domain) in order to trick I_{leg} into granting access to object o_1. That is, the attacker attempts to achieve:

$$\mathrm{Auth}(I_{\mathrm{leg}}, I_{\mathrm{leg}}, o_1) = \mathrm{true}$$

Impersonation Resistance via DPoP Proof Each DPoP proof σ is a digital signature over structured request data that includes the target URL of the request:

$$\sigma = \mathrm{Sign}_{k_C}(\mathrm{method}, \mathrm{url}, \mathrm{timestamp}, \mathrm{nonce})$$

In particular, the url field in the signed structure includes the fully qualified domain of the target, e.g.,:

$$\mathrm{url} = \mathrm{https://api.}I_{\mathrm{mal}}\mathrm{.com/resource}$$

Since the DPoP proof is verified by the PEP of I_{leg}, which expects the URL to match its own domain, the signature verification will fail. Specifically, the PEP at I_{leg} will extract the DPoP proof and attempt to verify:

$$\mathrm{Verify}(K_C, \sigma, \mathrm{https://api.}I_{\mathrm{leg}}\mathrm{.com/resource})$$

This fails because the signed structure bound to I_{mal} does not match the target URL expected by I_{leg}. Therefore:

$$\neg\mathrm{ValidDPoP}(T_C, \mathrm{req}_{I_{\mathrm{leg}}}, K_C)$$

4.1.4 Revocation Consistency

Revocation consistency ensures that once an authorization relationship is revoked, the corresponding access rights are also withdrawn in a timely and effective manner across the system. In particular, when a tuple is deleted or updated in the PAP, any future token issuance or access request that relies on this relationship should be denied. For this purpose, in order to ensure consistency, our system enables the following:

- Access tokens are short-lived and must be refreshed periodically using a secure token endpoint.
- Refresh operations re-evaluate the authorization logic at the PAP, ensuring that revocation is enforced.

4.1.5 Privacy

Using OAuth 2.0 token exchange, an organization learns when a data consumer interacts with an intermediary. This happens because the PDP must receive a security token from the PAP in order for a consumer to be authorized. Therefore, this facilitates tracking by the PAP. In particular, when a consumer C requests access to an object o via an intermediary I, the authorization check involves sending relation tuple queries to PAPs. However, these queries are scoped only to the tuples relevant to the authorization decision and do not reveal the specific resource being requested, nor the full access context. Additionally, identity tokens used in the protocol do not include plaintext consumer identifiers, but only a digest of the public key. As a result, while intermediaries may observe the frequency or volume of access requests, they are not exposed to the full trace of consumer behavior across the data space.

4.1.6 Availability Considerations

In distributed access control systems such as the one we propose, availability is a critical aspect that must be addressed to ensure uninterrupted authorization decisions. The use of OAuth 2.0 Token Exchange introduces potential availability risks, particularly related to the Policy Administration Points (PAPs) of organizations and data owners.

Specifically, the PAP of an organization may become a single point of failure during the initial authorization of a data consumer. At this stage, the PAP must be available to validate the identity token and to generate and return the corresponding security token that encodes the user's relationship tuples. This token is then consumed by the Policy Decision Point (PDP). The PAP must also remain available to refresh security tokens when they expire or are evicted from the PDP's cache.

Similarly, the PAP of the data owner–responsible for managing relationships over data space entities–must be reachable by the PDP when evaluating access decisions involving those resources. If the data owner's PAP is unavailable, access requests cannot be evaluated in real time, potentially disrupting service availability.

Mitigation Strategies. Several mechanisms can be employed to mitigate these availability concerns:

- **Caching of Security Tokens.** PDPs may cache the security tokens obtained from an organization's PAP for a limited time. The cache duration is determined by the token's validity period, which is set by the issuing organization. This approach reduces dependency on the organization's PAP for every request and allows access decisions to proceed even in case of temporary unavailability.
- **Caching of Data Owner PAP Responses.** Similar caching strategies can be applied to queries involving the data owner's PAP. For example, the result of evaluating a user's access to a specific resource could be cached and reused for the token's lifetime, assuming that authorization policies are relatively stable.
- **Redundant PAP Deployments.** A data space can support multiple independent PAPs for different data owners, distributing the availability risk across entities rather than centralizing it.
- **Graceful Degradation and Fallback.** In some cases, PDPs may implement fallback policies when upstream PAPs are temporarily unavailable. For example, access could be granted based on previously validated credentials or conservative defaults, depending on the security context and sensitivity of the data.

By combining token-based caching, distributed architecture, and robust deployment strategies, the risk of availability bottlenecks can be minimized.

4.2 Performance Evaluation

In this section, we evaluate the performance of our proposed solution.[5] All experiments were conducted using a Dockerized deployment of OpenFGA v1.8.6, running on macOS with an Apple M4 CPU and 16 GB of RAM, configured to use in-memory storage without any optimization. In every experiment, OpenFGA was initialized with the authorization model shown in Listing 3.

We investigated four scenarios, each representing a different depth of relationship-based access control:

- **Direct access rights:** A user is directly granted read access to a device.
- **Second-level access rights:** A user has read access to a group, and the device belongs to that group.
- **Third-level access rights:** A user belongs to a user group that has read access to a group of devices.
- **Fourth-level access rights:** A user belongs to a group that is nested within another group that holds read access to a device.

In all evaluation scenarios, we begin by pre-loading OpenFGA with the full set of relationship tuples relevant to the experiment. Once the data is loaded, we execute 100 access queries in a serialized manner, i.e., one query at a time, without parallelization.

Figure 3 illustrates the average, minimum, and maximum response times as a function of the number of stored relationships. This experiment focuses on users with direct access rights to devices. As shown, response times increase slightly with the number of relationships, but even with 120 K relationships, the average response time remains below 5 ms.

Figure 4 shows the impact of increasing relationship depth. Here, we fix the total number of relationships to 120 K and vary the depth of access. For example, in the two-level experiment, 60 K relationships connect devices to groups and another 60 K connect users to those groups. As the number of relationship levels increases, so does the query response time, indicating the computational cost of deeper traversals.

Figure 5 explores the use of contextual relationships to implement the distributed version of our authorization model. This setup evaluates the effect of injecting one or two contextual relationships per query while OpenFGA maintains only relationships among devices and groups.

[5] The source code for the performance measurements can be found at https://github.com/excid-io/rebac-data-spaces

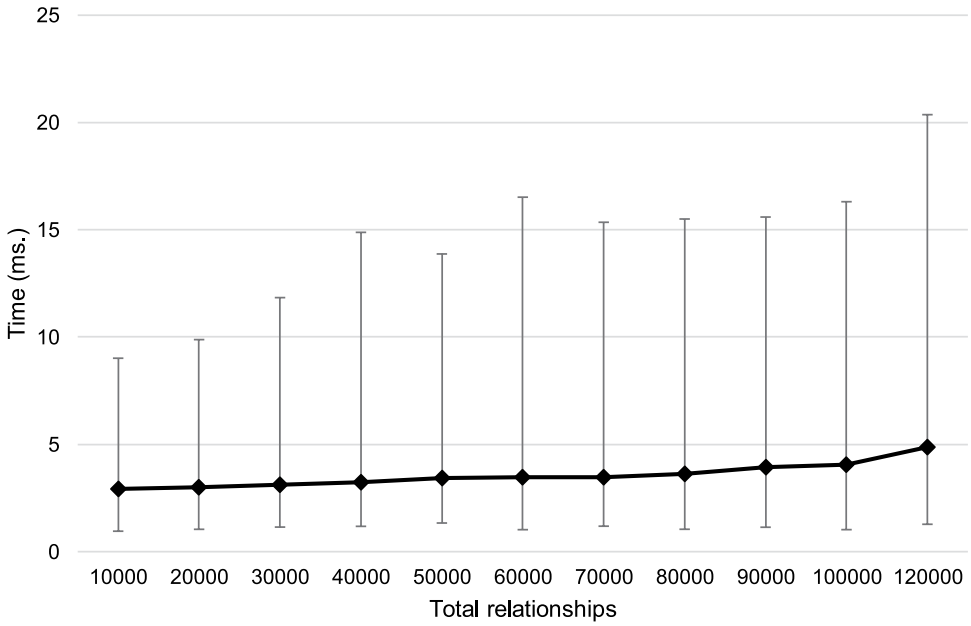

Fig. 3 Average, min, and max response time as function of the number of relationships

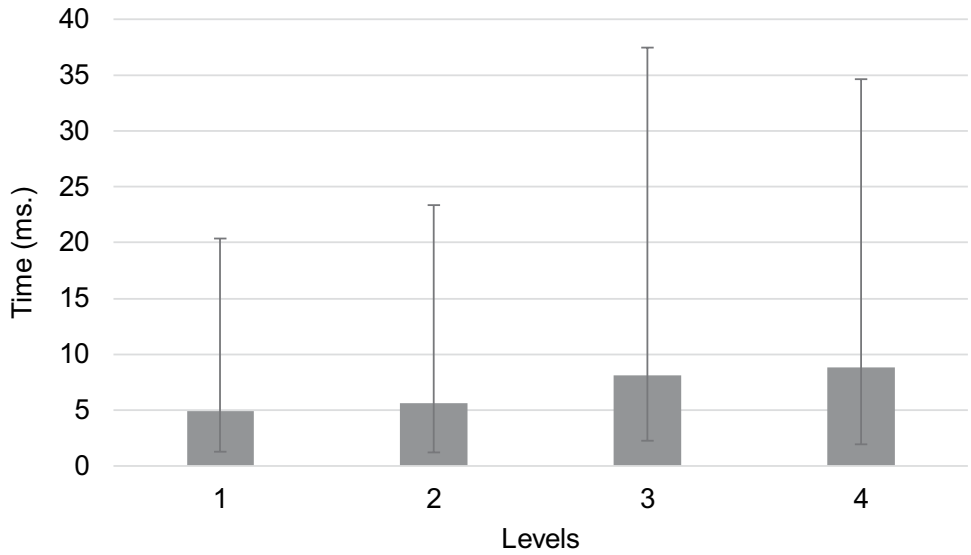

Fig. 4 Average, min, and max response time as function of the number of relationship levels

With one contextual relationship, the query includes the user-to-group link, simulating the two-level access pattern. With two contextual relationships, the query includes both the user-to-user-group and user-group-to-device-group links, effectively simulating a three-level access. The results show that, while response time increases with the number of relationships, contextual queries perform better than their non-contextual equivalents, demonstrating that offloading some relationships to the context can reduce server-side computation and improve performance.

5 Related Work

Munoz-Arcentales et al. [11] define a somewhat similar system for providing data usage and access control in data spaces, which is extended in [12]. However, the solution considers a single PAP managed by the corresponding intermediary, whereas our solution allows each owner to maintain its own PAP. Moreover, in contrast to our solution, the access control system presented in these papers is not data semantic-aware, which limits the functionality of the data space.

Our solution can be seen as a *usage control* solution, since it continuously monitors consumers' capabilities and acts accordingly when they are updated (e.g., it cancels a subscription). A notable improvement of our solution

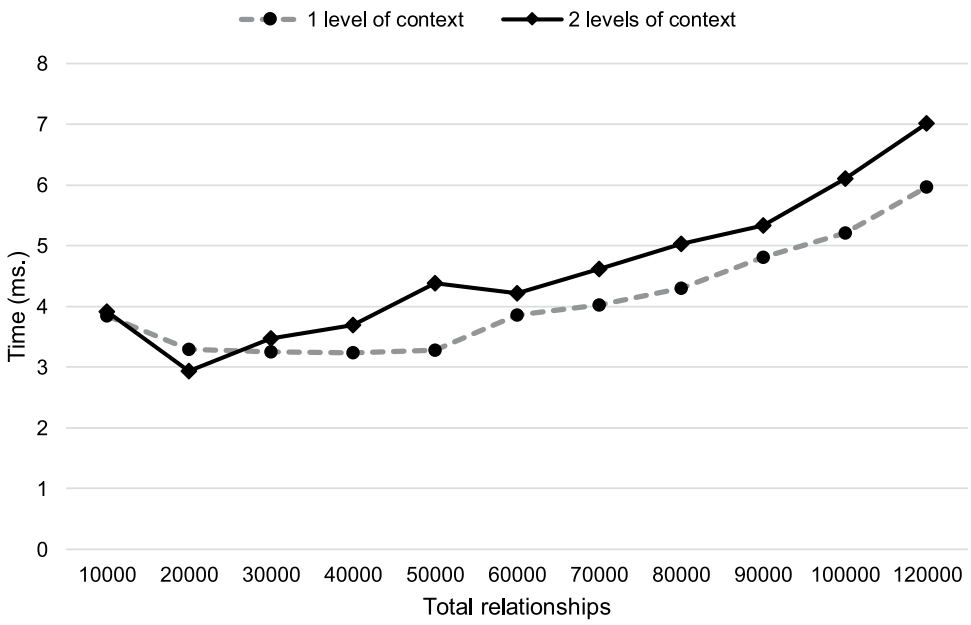

Fig. 5 Average response time of contextual queries as a function of the number of relationships

compared to existing systems (e.g., [13, 14]) is that in our solution the re-evaluation of the access rights also considers information provided by multiple, distributed PAPs, which allows the revocation of access rights even after the initial authorization has been completed.

Emerging data-sharing access control systems utilize cryptographic constructions to enforce access control policies. For example, the solution proposed by Ye et al. [15] employs Attribute-Based Encryption (ABE). These methods eliminate the need for a trusted PEP but introduce higher computational and key management overhead. Our solution can be easily adapted to use ABE by converting each capability into an attribute. Developing an ABE-based version of our solution remains a task for future work.

Bagheri [16] defines a governance framework for data platforms, focusing on the construction industry. However, unlike our paper, the work in [16] only identifies security and trust requirements without proposing a solution. Similarly, Hauwers et al. [17] only discuss data sharing business models, and Van Dijck [18] tries to visualize data sharing platforms and identify their governance structure.

Firdausy et al. [19] follow a similar structure for implementing data spaces, nevertheless their work focuses solely on the data intermediary role. Torres et al. [20] present the design of the security modules of a data space that can be used in the textile and clothing industry. They follow a typical approach where the corresponding PAP intervenes in the communication between a data consumer and a data intermediary. In our system, the communication between the intermediary and the PAP is minimized or even completely removed when the VC-based approach is used. This results in significant security and performance improvements.

Our system provides a capabilities-based access solution built upon the iSHARE trust framework for data governance. Nevertheless, alternative approaches can be considered. A growing body of research efforts investigates alternative data governance frameworks (e.g., [21–24]): these frameworks are complementary to our work and could be adopted in other use cases. Similarly, related efforts integrate external entities, e.g., the solution in [25] integrates eIDAS into their data space. By being standards' compliant, our solution can also interact with such external entities.

Various efforts explore similar concepts in other contexts. Seidel et al. [26] created a data space for managing the life cycle of data related to space applications. Their system is built on top of GAIA-X [27]. Similarly, Ionescu et al. [28] developed a data marketplace for geo-spatial data. Koren et al. [29] present the design of a similar architecture focusing on smart farming, whereas Scheider et al. [30] present the design of an architecture that can be used for sharing personal data. Pinto et al. [31] designed a B2B data exchange system for the footwear industry. Farahani and Monsefi [32] create a data space for exchanging data produced by Industrial IoT devices, which are used for training AI models. We believe that our access control solution is also applicable in these domains.

6 Conclusion

We designed a Relationship-Based Access Control (ReBAC) solution that specifically addresses the access control requirements of data spaces, which we presented in this paper. Our solution enables fine-grained access control by allowing the definition of access policies that capture

the intricate relationships among protected data objects as well as the organizational structures and roles of users. By aligning access control policies with these relationships, our approach not only enhances the expressiveness and precision of access control mechanisms but also strengthens data sovereignty, ensuring that data owners retain full control over how their data is accessed and utilized. This capability is critical for fostering trust and facilitating secure data sharing within the diverse and dynamic environments of data spaces.

Our security evaluation considers malicious entities (operating as malicious intermediaries) trying to gain unauthorized access to a data space (provided by another, legitimate, intermediary). The analysis demonstrates that the design is secure against such attacks. Our experimental performance evaluation (with the prototype we have developed) demonstrated that the approach is feasible for a large number of relationships (e.g., for 120.000 flat relationships, mean response time remains below 5 ms) and relationship depths (e.g., for 120.000 total relationships and a maximum depth of the hierarchy at 4, the mean response time remains below 10 ms).

Moreover, we extended our solution to support distributed deployment by leveraging the OAuth 2.0 token exchange mechanism. This extension allows the definition of relationship tuples across multiple administrative domains, enabling decentralized management of access control policies while maintaining interoperability and cohesion. This distributed architecture significantly enhances the scalability and flexibility of our solution, making it well-suited for real-world data spaces involving numerous stakeholders with varying requirements and responsibilities. The performance evaluation determined that the average response time of contextual queries as a function of the number of relationships remains small, and remains below 10 ms for 120.000 total relationships.

A notable advantage of our solution is that the re-evaluation of the access rights also considers information provided by multiple, distributed PAPs, which allows the revocation of access rights even after the initial authorization has been completed. Together, these features demonstrate the robustness and practicality of our ReBAC solution, providing a strong foundation for secure, sovereign, and collaborative data sharing in data spaces.

Acknowledgements Not Applicable

Author contributions All authors contributed to the study conception and design. Implementation of the solution was performed by Nikos Fotiou, Chalima Dimitra Nassar Kyriakidou and Athanasia Maria Papathanasiou. Evaluation of the solution was performed by Vasilios Siris and George Polyzos. Related work was collected and analyzed by all authors. The first version of the manuscript was written by Nikos Fotiou and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Funding This work has been funded in part by EU's Horizon 2020 Programme, through the subgrant "Identity and Access Management for the Computing Continuum" (IAM4CC) of project FLUIDOS, under grant agreement No. 101070473.

Data availability Not Applicable

Declarations

Competing interests Not Applicable

References

1. Beverungen D, Hess T, Köster A, Lehrer C (2022) From private digital platforms to public data spaces: implications for the digital transformation. Electron Mark 32(2):493–501
2. Reuver M, Ofe H, Agahari W, Abbas AE, Zuiderwijk A (2022) The openness of data platforms: A research agenda. In: Proceedings of the 1st International Workshop on Data Economy. DE '22, 34–41. Association for Computing Machinery, New York, NY, USA
3. Gelhaar J, Otto B (2020) Challenges in the emergence of data ecosystems. In: Proceedings of the 24th Pacific Asia Conference on Information System, (PACIS), 175
4. Schmidt K, Munilla Garrido G, M¨uhle A, Meinel C (2022) Mitigating sovereign data exchange challenges: A mapping to apply privacy- and authenticityenhancing technologies. In: Katsikas S, Furnell S (eds) Trust, Privacy and Security in Digital Business. Springer, Cham, pp 50–65
5. Pang R, Caceres R, Burrows M, Chen Z, Dave P, Germer N, Golynski A, Graney K, Kang N, Kissner L, Korn JL, Parmar A, Richards CD, Wang M (2019) Zanzibar: Google's consistent, global authorization system. In: 2019 USENIX Annual Technical Conference (USENIX ATC '19), Renton, WA
6. Context Information Management (CIM) ETSI ISG: NGSI-LD API. Group Specification CIM-009v161, ETSI (2022)
7. Jones M, Nadalin A, Campbell B, Bradley J, Mortimore C (2020) OAuth 2.0 Token Exchange. RFC 8693, IETF
8. Fett (ed.) D (2023) Oauth 2.0 demonstrating proof of possession (dpop). RFC 9449, IETF
9. Heilman E, Mugnier L, Filippidis A, Goldberg S, Lipman S, Marcus Y, Milano M, Premkumar S, Unrein C, Merfeld J (2023) OpenPubkey: Augmenting OpenID Connect with User held Signing Keys. Cryptology ePrint Archive, Paper 2023/296. https://eprint.iacr.org/2023/296

10. Ahle U, Hierro JJ (2022) FIWARE for data spaces. Designing Data Spaces, 395 - 417 https://doi.org/10.1007/978-3-030-93975-5_24
11. Munoz-Arcentales A, López-Pernas S, Pozo A, Alonso A, Salvachúa J, Huecas G (2019) An architecture for providing data usage and access control in data sharing ecosystems. Procedia Computer Science 160, 590–597. The 10th International Conference on Emerging Ubiquitous Systems and Pervasive Networks (EUSPN-2019) / The 9th International Conference on Current and Future Trends of Information and Communication Technologies in Healthcare (ICTH-2019) / Affiliated Workshops
12. Munoz-Arcentales A, López-Pernas S, Pozo A, Alonso A, Salvachúa J, Huecas G (2020) Data usage and access control in industrial data spaces: implementation using FIWARE. Sustainability. https://doi.org/10.3390/su12093885
13. Carniani E, D'Arenzo D, Lazouski A, Martinelli F, Mori P (2016) Usage control on cloud systems. Futur Gener Comput Syst 63:37–55
14. Zrenner J, Möller FO, Jung C, Eitel A, Otto B (2019) Usage control architecture options for data sovereignty in business ecosystems. J Enterp Inf Manag 32(3):477–495
15. Ye Y, Zhang L, You W, Mu Y (2021) Secure decentralized access control policy for data sharing in smart grid. In: IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 1–6. IEEE
16. Bagheri S (2023) Towards a governance framework for data platform ecosystems in the construction industry 2:566–574
17. D'Hauwers R, Walravens N (2022) Do you trust me? Value and governance in data sharing business models. Proceedings of Sixth International Congress on Information and Communication Technology. Springer, Singapore, pp 217–225
18. Van Dijck J (2021) Seeing the forest for the trees: visualizing platformization and its governance. New Media Soc 23(9):2801–2819
19. Firdausy DR, Alencar Silva P, Sinderen M, Iacob M-E (2022) A data connector store for international data spaces. In: Sellami M, Ceravolo P, Reijers HA, Gaaloul W, Panetto H (eds) Cooperative Information Systems. Springer, Cham, pp 242–258
20. Torres N, Chaves A, Toscano C, Pinto P (2023) Prototyping the ids security components in the context of industry 4.0 - a textile and clothing industry case study. In: Wang G, Choo K-KR, Wu J, Damiani E (eds) Ubiquitous Security. Springer, Singapore, pp 193–206
21. Ansey R, Kempf J, Berzin O, Xi C, Sheikh I (2019) Gnomon: Decentralized identifiers for securing 5g IoT device registration and software update. In: 2019 IEEE Globecom Workshops (GC Wkshps), 1–6
22. Lee SU, Zhu L, Jeffery R (2018) A data governance framework for platform ecosystem process management. In: Weske M, Montali M, Weber I, Brocke J (eds) Business Process Management Forum. Springer, Cham, pp 211–227
23. Zhang Q, Sun X, Zhang M (2022) Data matters: a strategic action framework for data governance. Inf Manage 59(4):103642
24. Paparova D, Aanestad M, Vassilakopoulou P, Bahus MK (2023) Data governance spaces: the case of a national digital service for personal health data. Inf Organ 33(1):100451
25. Alonso Á, Pozo A, Choque J, Bueno G, Salvachúa J, Diez L, Marín J, Alonso PLC (2019) An identity framework for providing access to fiware OAuth 2.0-based services according to the eidas European regulation. IEEE Access 7:88435–88449
26. Seidel A, Wenzel K, Hänel A, Teicher U, Weiß A, Schäfer U, Ihlenfeldt S, Eisenmann H, Ernst H (2023) Towards a seamless data cycle for space components: considerations from the growing European future digital ecosystem Gaia-X. CEAS Space J 15. https://doi.org/10.1007/s12567-023-00500-4
27. Otto B (2022) A federated infrastructure for European data spaces. Commun ACM 65(4):44–45
28. Ionescu A, Patroumpas K, Psarakis K, Chatzigeorgakidis G, Collarana D, Barenscher K, Skoutas D, Katsifodimos A, Athanasiou S (2023) Topio: An open-source web platform for trading geospatial data. In: International Conference on Web Engineering, 336–351 . Springer
29. Koren I, Braun S, Van Dyck M, Jarke M (2021) Dynamic strategic modeling for alliance-driven data platforms: the case of smart farming. In: Nurcan S, Korthaus A (eds) Intelligent Information Systems. Springer, Cham, pp 92–99
30. Scheider S, Lauf F, Möller F, Otto B (2023) A reference system architecture with data sovereignty for human-centric data ecosystems. Business & Information Systems Engineering, 1–19
31. Pinto P, Sousa C, Cardeiro C (2023) Data spaces based approach for B2B data exchange: a footwear industry case. Procedia Comput Sci 219:933–940
32. Farahani B, Monsefi AK (2023) Smart and collaborative industrial IoT : a federated learning and data space approach. Digit Commun Networks 9(2):436–447