

Access control for Data Spaces

Nikos Fotiou,^{*} Vasilios A. Siris,^{*†} George C. Polyzos^{*†‡}

^{*} ExcID P.C., 11362 Athens, Greece

[†] Department of Informatics, School of Information Sciences and Technology,
Athens University of Economics and Business, 10434 Athens, Greece

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

Abstract—Data spaces represent an emerging paradigm that facilitates secure and trusted data exchange through foundational elements of data interoperability, sovereignty, and trust. Within a data space, data items, potentially owned by different entities, can be interconnected. Concurrently, data consumers can execute advanced data lookup operations and subscribe to data-driven events. Achieving fine-grained access control without compromising functionality presents a significant challenge. In this paper, we design and implement an access control mechanism that ensures continuous evaluation of access control policies, is data semantics aware, and supports subscriptions to data events. We present a construction where access control policies are stored in a centralized location, which we extend to allow data owners to maintain their own Policy Administration Points. This extension builds upon W3C Verifiable Credentials.

Index Terms—Interoperability, Verifiable Credentials, subscriptions, semantics, decentralization, sovereignty, NGS-LD

I. INTRODUCTION

Data spaces are emerging as a new form of digital platform aiming at eliminating silos and enabling data-driven innovations and shape the digital transformation [1]. A growing number of reports by commercial entities and governmental bodies highlight the business potential and the possible societal impact that can be achieved by embracing data spaces (see for example [2]). A data space is composed of building blocks that enable semantic interoperability of data, uniform data access methods, as well as increased data sovereignty and trust. Nevertheless, data usage policies and access control still remain a challenge for data spaces [3], [4]. Data owners do not want to lose control and sovereignty over their data; data users want to safeguard against low-quality or malicious data [5].

We use a data space for exchanging data generated by IoT devices as a motivating use case. This data space consists of IoT device *owners* who wish to share data generated by their devices, acting as data *suppliers*, data *consumers* who seek to access the provided data through a *client application*, and data *intermediaries* that offer a data *context broker* that implements the corresponding data management and access APIs.

A context broker maintains digital objects, which in our use case are *digital twins* of the corresponding IoT devices. Each such object is uniquely identifiable and serialized using JSON for *Linked Data* (JSON-LD) [6]. An object has a *type* and *attributes*. Accordingly, a type can be associated with many objects and an object may have multiple attributes. For

example, a data space may include objects of type *smart lamp*, with identities such as *lamp1* and *lamp2*, and attributes such as *consumption*, *status*, and *color*.

A context broker implements the ETSI standard *Next Generation Service Interfaces Linked Data* (NGSI-LD) API [7], which allows HTTP-based operations to digital objects stored in the broker, as well as subscriptions to events related to these objects. This API simultaneously allows fine-grained data access, e.g., a consumer may request only specific attributes of an object, as well as coarse-grained access, e.g., a consumer may request attributes of all objects of a specific type.

A. Access control

A context broker should be protected by an access control mechanism that allows access to stored data only by authorized data consumers. An access control solution should achieve:

- **Attack surface reduction:** An access control solution should strive to minimize potential security threats. The amount of security-sensitive information managed by data consumers should be minimal. Similarly, access verification should be simple and not prone to errors.
- **Usage control:** Access rights of a consumer should be re-evaluated, even after a consumer has been initially authorized by a data intermediary. For example, a consumer that has successfully subscribed to receive notifications about an object should stop receiving new notifications if its corresponding access rights are revoked.
- **Enhanced privacy:** An access control solution should prevent tracking of data consumers not only by third parties, but also by participants of the data space (e.g., by data intermediaries and owners).
- **Availability:** An access control solution should not depend on (Internet) connectivity, instead it should function correctly even if some of its components are unreachable.

B. Contributions

In this paper, we present the design and evaluation of an access control solution tailored to data spaces, making the following key contributions: (i) we introduce access control policies that are aware of data semantics (e.g., a consumer can be authorized to access all objects of a certain type); (ii) we enable continuous monitoring and re-evaluation of consumer access rights, supporting long-term operations such

as event subscriptions; (iii) we facilitate distributed deployments, allowing each data owner to maintain their own *Policy Administration Point*, thereby enhancing owner sovereignty.

In the remainder of this paper, we detail our design in Section II, present its implementation and evaluation in Section III, and our conclusions and future work in Section IV.

II. DESIGN

A. Underlay Data Space

In the considered data space, data owners create new objects and assign policies specifying the operations a data consumer can perform on an object. Objects and types are uniquely identified using a URL denoted by URL_{object} and URL_{type} , respectively. Similarly, object attributes are identified by a URL_{attr} , which has the form $URL_{object}/attributename$. Consumers are also identified by an identifier denoted by $Consumer_{id}$. Consumers can request access to all objects of a certain type, or to specific objects. Additionally, consumers can provide filters specifying which attributes they wish to access. Consumers can perform read, write, and subscription operations over stored objects using the ETSI NGSI-LD API and the corresponding endpoint of the context broker.

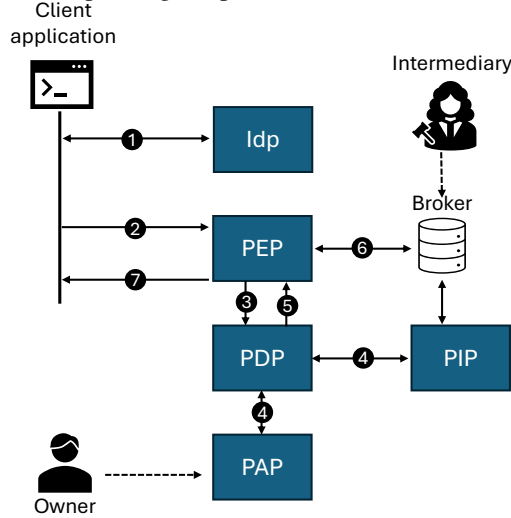


Fig. 1. High level overview of the authorization process.

B. Components and interactions

Our access control solution is composed of the following components:

a) *Identity Provider (IdP)*: A registry of consumer identifiers. This registry is maintained by a party trusted by owners, intermediaries, and consumers; this can be a Trusted Third Party or any of these three entities.

b) *Policy Administration Point (PAP)*: A registry where owners can store access control policies that define the access rights of each consumer.

c) *Policy Enforcement Point (PEP)*: A transparent HTTP proxy that intercepts API calls from consumers to the context broker. If a request originates from a consumer with the

appropriate access rights it is forwarded to the (context) broker, otherwise it is rejected.

d) *Policy Decision Point (PDP)*: A component that decides whether or not a request originates from an authorized consumer.

e) *Policy Information Point (PIP)*: A component that provides supplementary information used by a PDP to make an access control decision. Specifically, this component provides details about the attributes and type of an object.

From a high-level perspective, authorization using our solution is implemented as follows (see also Fig. 1). The IdP is configured with consumer identifiers, e.g., through a client registration process. Similarly, an owner configures the PAP with the appropriate access control policies. A consumer initially, through its client application, 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 the access control decision (step 3). The PDP collects the consumer's access rights from the PAP and, if necessary, information related to the requested object from the PIP (step 4). Using the retrieved information, the PDP makes an access control decision that it forwards to the PEP (step 5). Finally, if the consumer is authorized, the PEP forwards the request to the context broker (step 6) and relays its response back to the consumer (step 7).

C. Policies and enforcement

Our system implements capabilities-based access control, where an access control policy specifies the operations (i.e., Read, Write, Subscribe) that a consumer can perform on specific object types, and/or on specific objects, and/or on specific object attributes. For example, a consumer can be allowed to *Read all objects of type "smart lamp", and Write the attribute "energy consumption" of the objects with identifiers "smart lamp1" and "smart lamp2"*. More formally, we define an access control policy p as the tuple $[Consumer_{id}, operation, URL]$, where $operation$ can be Read, Write, or Subscribe, and URL can be a URL_{type} or URL_{object} or URL_{attr} . We define an access control decision as the following function:

$$Decide(Consumer_{id}, operation, URL) \rightarrow \{true, false\}$$

In order to enable access control decisions we define the $URL_A \supseteq URL_B$ operation which outputs *true* if one of the following conditions hold:

- $URL_A = URL_B$
- URL_A is a URL_{type} and URL_B a URL_{object} of an object of type URL_A
- URL_A is a URL_{type} and URL_B a URL_{attr} of an attribute of an object of type URL_A

- URL_A is a URL_{object} and URL_B a URL_{attr} of an attribute of object URL_A

The access control decision is implemented using Algorithm 1. Its semantics denote that if a consumer is authorized to perform an operation on a type, it can perform the same operation on all objects of this type and their attributes. Similarly, if a consumer is authorized to perform an operation on an object, it is authorized to perform the same operation on all its attributes.

Algorithm 1 Access control decision algorithm

```

Let  $P$  the set of all policies
procedure DECIDE( $Consumer_{id}, operation, URL$ )
  for all  $p \in P$  do
    if  $p[Consumer_{id}] = Consumer_{id}$  AND
       $p[operation] = operation$  AND
       $p[URL] \supseteq URL$  then
      return true
    end if
  end for
  return false
end procedure

```

In many cases, for the PDP to make an access control decision the type of the requested object must be known: this *type inference* functionality is provided by the PIP. For example, assume that consumer C has received authorization for the *Read* operation on object type T (e.g., authorization to *Read all objects of type "smart lamp"*). C makes a *Read* request for attribute A of object O_1 (e.g., a request to *Read the "status" of "smart lamp 1"*). For the PDP to make an access control decision it needs to infer the type of O_1 . This is achieved using the PIP, which communicates with the context broker and obtains the type of O_1 using the corresponding NGSI-LD API call. Then, the PDP uses this information to make a decision, i.e., if O_1 is of type T the request is accepted.

Finally, our solution provides *Automatic un-subscription*. Specifically, the PDP maintains a list of active subscriptions and automatically un-subscribes (invoking an API call) consumers that are no longer authorized to receive notifications (e.g., the access rights have expired or were revoked).

D. Distributed PAPs

We now present an extension allowing PAPs to be managed by the corresponding owners and provides increased security and improved sovereignty for data owners. We leverage W3C Verifiable Credentials (VCs) to enable intermediary's access control components to learn the capabilities of a consumer.

A VC is a W3C recommendation that allows an *issuer* to assert some claims about an entity, referred to as the *VC subject*. A VC includes information about the issuer, the subject, the asserted claims, as well as possible constraints (e.g., expiration date) [8]. This information is encoded in

a machine readable format (e.g., as a JSON object in our system). Then, a VC holder (usually, the VC subject itself) can prove to a verifier that it owns one or more VCs with certain characteristics. This is achieved by binding VCs to a subject identifier (e.g., a public key) that can be used for generating a *Verifiable Presentation* (VP) of the VC(s). A VP is an object (a JSON object in our system) that includes one or more VCs and it is signed in a way that can be verified using the subject identifier (specified in the included VC(s)). VP verification does not require communication with the issuer.

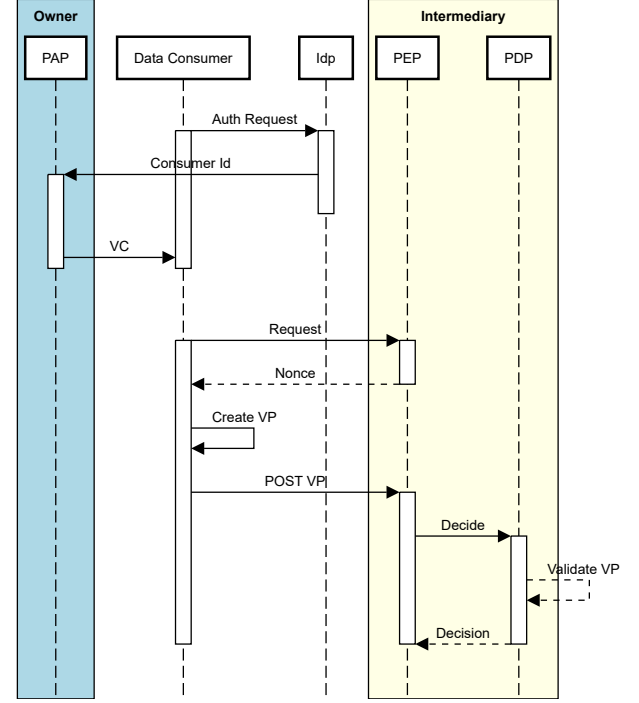


Fig. 2. Authorization using Verifiable Credentials.

In our data space solution, a VC can be used as a means for communicating consumer capabilities (see also Fig. 2). In this case, a VC is issued by a PAP to a consumer and is serialized as a JSON object that includes the consumer access rights and a public key provided by the consumer; this JSON object is digitally signed by the PAP. VC issuance is implemented using OpenID for Verifiable Credential Issuance [9]. Using this protocol a consumer authenticates to the PAP (acting as the VC issuer) using the IdP and OpenID connect. Then, the PAP issues the corresponding credential.

Data consumer authorization is implemented using OpenID for Verifiable Presentation [10] and involves the following steps: First, a consumer sends an *unauthorized* request and receives a *nonce*. Then, the consumer creates and submits a Verifiable Presentation (VP). The VP includes a valid VC, the received nonce, and the HTTP URL of the intermediary, and is signed using the private key of the consumer identifier. Then, the PDP endpoint performs the following validations:

- It verifies that the VC has been issued by a trusted PAP.

- It verifies the correctness and the validity of the VC by validating its digital signature, its issuance and expiration time, and by querying for its revocation status (see the following section.)
- It validates the VP proof using the consumer's public key included in the VC, and verifies that it includes a valid nonce and intermediary URL.

E. Usage control

Consumer capabilities are serialized using VCs that include an *expiration* time. Capabilities can be cached by the PEP up until their expiration time for two reasons: (i) to avoid consumer re-authorization, and (ii) to enable access control on subscriptions. However, while they are cached, capabilities may be revoked, thus a PDP needs to be able to verify their validity periodically, which is challenging when PAPs are distributed. For this reason our solution supports VC revocation. Specifically, we rely on a recent W3C draft [11] that defines an efficient revocation mechanism. To support revocation a PAP maintains a revocation list that covers all *non-expired* VCs that it has issued. This list is a simple bit string and each VC is associated with a position in the list. Consequently, each VC includes a property named *credentialStatus* that specifies the position of that VC in the revocation list, as well as a URL that can be used for retrieving the revocation list. A VC is simply revoked by setting the corresponding bit in the revocation list to 1. Since the list includes only non-expired VCs, its size is tolerable for most use cases. Similarly, since this list is expected to include many 0s and few 1s it can be efficiently compressed. A revocation list is included in a VC, issued by the PAP, which is periodically downloaded by the PDP. Nevertheless, since a revocation list covers all non-expired VCs issued by the PAP, the frequency at which a PDP downloads this list does not depend on the number of stored capabilities. On the other hand, this approach does not allow capability updates, neither fine-grained capabilities-management: once a VC is revoked, all consumer capabilities stored by the PDP are removed.

III. IMPLEMENTATION

We implemented a prototype of our data space for sharing smart building IoT device data. Data are stored in the backends of two companies (acting as the data suppliers). They are serialized using JSON-LD and stored in a context broker implemented using FIWARE Orion [12].

As an IdP we used keycloak.¹ A PAP has also been implemented following the iSHARE trust framework². Using this framework, owners can specify access control policies that define the access rights of each data consumer using the iSHARE policy definition language,³ which is inspired

by XACML. A policy defines the operations that a consumer can perform over a set of object types, and/or a set of object identifiers, and/or a set of object attributes. The corresponding PDP and PIP have been implemented as custom applications. Finally, for the PEP we rely on Ory Oathkeeper.⁴

A. Security evaluation

We now evaluate the security properties of our solution focusing on the distributed version by revisiting the security requirements defined in Section 1. Our solution makes the following security assumptions. The communication of all entities in our system takes place over secure communication channels; similarly, all cryptographic operations are secure. Furthermore, it is assumed that IdPs and PAPs are trusted, as well as legitimate intermediaries operate as expected. Finally, it is assumed that all secrets are properly secured.

1) *Attack surface reduction*: The W3C VC data model is a W3C recommendation, for which there are not many implementations available. In our VC-based approach, a data consumer needs to protect a secret used for authenticating to the IdP, as well as the private key used for signing VPs. Since a VC is received from a data consumer (and not from the PAP directly), a PDP needs to verify its integrity by validating its signature, which has been generated by the PAP.

Our threat model considers malicious entities trying to gain unauthorized access to a data space. These entities operate as (malicious) intermediaries and their goal is to gain access to a data space provided by another (legitimate) intermediary. More formally, 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,

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

The goal of I_{mal} is to achieve $Auth(I_{mal}, I_{leg}, o) \rightarrow true$. Due to the second assumption, I_{mal} has access to a VP generated by C . In order to achieve its goals it should be able to use them in a request towards I_{leg} . However, a VP includes the URL of I_{mal} and since I_{mal} cannot modify them, they will be rejected by I_{leg} .

2) *Usage Control*: Usage control is implemented using revocation lists. In this case some overhead is introduced.

a) *Initial authorization overhead*: VC issuance and authorization using a VP are asynchronous processes, therefore the PDP must verify the revocation status of a VC immediately after a VP has been received. This introduces additional delay the first time a consumer tries to interact with the data space.

b) *Authorization modification overhead*: In case the capabilities of a consumer are modified, old consumer capabilities should be revoked in order for the PDP to request a new VC from the consumer. Alternatively, a consumer can proactively “refresh” its VCs; this is left as future work.

¹<https://www.keycloak.org/>

²<https://ishare.eu/>

³<https://dev.ishare.eu/delegation/policy-sets.html>

⁴<https://www.ory.sh/oathkeeper/>

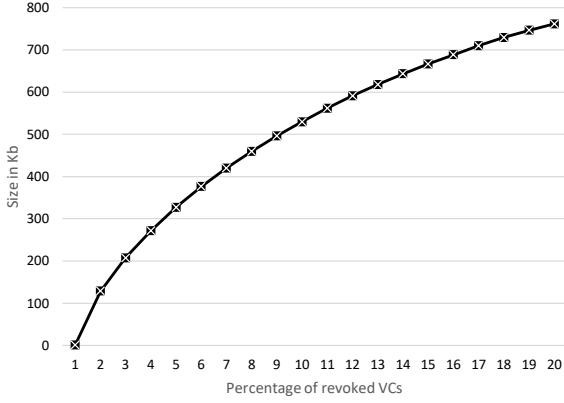


Fig. 3. Size of a VC revocation list with $1M$ VCs, as a function of the percentage of the revoked VCs.

c) *Communication overhead*: The communication overhead imposed to a PDP due to the usage control mechanisms of our solution depends on the number of PAPs that have issued the VCs included in the stored VPs. Particularly, if a PDP has stored the capabilities of $|C|$ consumers whose authorizations are provided by $|P|$ PAPs, the PDP has to send $|P|$ requests. Additionally, the size of the PAP response (i.e., the revocation list) depends on the number of VCs the PAP has issued (in general) and the percentage of the revoked VCs. Fig. 3 illustrates the size of a VC revocation list, provided by a single PAP, that includes 10^6 VCs, as a function of the percentage of the revoked VCs. The figure shows that, because the list is compressed, a smaller number of revoked VCs yields a smaller list size.

3) *Privacy*: A consumer has to communicate only once with the PAP in order to obtain a VC; in all subsequent requests the PAP is not involved. Even during VC revocation status verification, a PAP cannot distinguish which specific VC a revocation status check concerns, since the revocation list includes information about all non-expired VCs.

4) *Availability*: Our VC-based approach does not require any communication with the PAP during consumer authorization. Although the PDP needs access to the revocation list, this does not have to be provided directly by the PAP. Since the revocation list is included in a VC signed by the PAP, it can be provided by other entities or even the consumer. In that case, the PDP should verify that the list is “adequately fresh” and that it has been signed by the appropriate PAP.

IV. CONCLUSIONS AND FUTURE WORK

In this paper, we presented an access control solution tailored for Data Spaces. Our approach is Data Space API-aware and considers data semantics, enabling fine-grained policies. Furthermore, our Policy Enforcement Points are implemented as transparent HTTP proxies, making the protected resources oblivious to the access control mechanisms in place. This design ensures our solution can protect any Data Space system. We also presented a distributed approach that allows data

owners to manage Policy Administration Points, enhancing security and increasing data owner sovereignty.

Our solution leverages iSHARE’s authorization model to implement capabilities-based access control. Future work will explore other access control paradigms, such as relation-based access control, and will integrate additional frameworks like the Open Policy Agent. We aim to extend our PEPs to apply access control policies to outgoing data, such as filtering based on user capabilities. Furthermore, our solution currently uses public keys to bind tokens and verifiable credentials to consumers; future research will investigate alternative identification methods, such as Decentralized Identifiers [13]. Finally, we plan to enhance our solution to provide trustless PEPs by encrypting data with keys bound to access control policies, such as Attribute-Based Access Control (ABAC).

ACKNOWLEDGMENT

This work has been funded in part by EU’s Horizon 2020 Programme, through the subgrant “A real-time AI-enabled worker safety preservation system” (MILESTONE) of project Trialsnet, under grant agreement No. 10109587.

REFERENCES

- [1] D. Beverungen, T. Hess, A. Köster, and C. Lehrer, “From private digital platforms to public data spaces: implications for the digital transformation,” *Electronic Markets*, vol. 32, no. 2, pp. 493–501, 2022.
- [2] S. Scerri, T. Tuikka, I. L. de Vallejo, and E. Curry, “Common european data spaces: Challenges and opportunities,” *Data Spaces: Design, Deployment and Future Directions*, pp. 337–357, 2022.
- [3] J. Gelhaar and B. Otto, “Challenges in the emergence of data ecosystems,” in *Proc. PACIS*, 2020.
- [4] K. Schmidt, G. Munilla Garrido, A. Mühle, and C. Meinel, “Mitigating sovereign data exchange challenges: A mapping to apply privacy- and authenticity-enhancing technologies,” in *Trust, Privacy and Security in Digital Business*, 2022.
- [5] M. de Reuver, H. Ofe, W. Agahari, A. E. Abbas, and A. Zuiderwijk, “The openness of data platforms: A research agenda,” in *Proc. 1st Intl. Workshop on Data Economy*, 2022.
- [6] M. Sporny et al., “JSON-LD 1.1, A JSON-based Serialization for Linked Data,” W3C Recommendation, 2020.
- [7] Context Information Management (CIM) ETSI ISG, “NGSI-LD API,” ETSI, Group Specification CIM-009v161, 2022.
- [8] M. Sporny et al., “Verifiable credentials data model v1.1,” W3C Recommendation, 2022. [Online]. Available: <https://www.w3.org/TR/verifiable-claims-data-model/>
- [9] T. Lodderstedt, K. Yasuda, and T. Looker, “OpenID for Verifiable Credential Issuance,” OpenID Connect WG, Internet draft, 2023, https://openid.net/specs/openid-4-verifiable-credential-issuance-1_0.html.
- [10] O. Terbu, T. Lodderstedt, K. Yasuda, and T. Looker, “OpenID for Verifiable Presentations - draft 18,” OpenID Connect WG, Internet draft, 2023, https://openid.net/specs/openid-4-verifiable-presentations-1_0.html.
- [11] M. Sporny et al., “Verifiable credentials status list v2021,” W3C Draft Community Group Report, 2023, <https://www.w3.org/TR/vc-status-list/>.
- [12] U. Ahle and J. J. Hierro, “FIWARE for data spaces,” *Designing Data Spaces*, 2022.
- [13] W3C Credentials Community Group. (2020) A Primer for Decentralized Identifiers. [Online]. Available: <https://w3c-ccg.github.io/did-primer/>