A Preliminary Approach for Translating ODRL to Smart Policies

Stefano Bistarelli[1], Chiara Luchini[1,2,*] and Francesco Santini[1]

[1]Department of Mathematics and Computer Science, University of Perugia, Via Vanvitelli 1, 06123 Perugia (PG), Italy
[2]Department of Mathematics and Computer Science "Ulisse Dini", University of Florence, Viale Giovanni Battista Morgagni 67/a, 50134 Florence (FI), Italy

Abstract

Protecting sensitive data is a major challenge, particularly in sectors such as healthcare and industry, where cyberattacks can lead to unauthorized disclosure and operational disruptions. To address these issues, this paper combines *Self-Sovereign Identity* (*SSI*) and *Attribute-Based Access Control* (*ABAC*) to define fine-grained usage policies for digital credentials. We propose ABAC policies based on the *Open Digital Rights Language* (*ODRL*) standard, embedded within *Verifiable Credentials* (*VCs*), and a translation mechanism that converts them into *Smart Policies* (*SPs*) implemented as smart contracts, enabling transparent and automated policy enforcement.

Keywords

Self Sovereign Identity, ABAC, ODRL, Smart Policy

1. Introduction

Managing access to sensitive resources is a major challenge in several domains, particularly in healthcare. Beyond interrupting healthcare services, these attacks often lead to the unauthorized disclosure of *personal health information* (*PHI*), which can subsequently be sold on the Dark Web or exploited for identity theft. One possible solution to mitigate these issues is the adoption of *Self-Sovereign Identity* (*SSI*) systems, in which users retain control over their digital identities and credentials [1]. In healthcare, for example, an issuer could define usage conditions for a patient's health insurance by restricting access to services offered only by affiliated medical institutions. These usage constraints can be embedded directly within the issued credentials. Similar requirements arise in the industrial sector, where organizations manage sensitive assets such as intellectual property, projects, and employee information. At the same time, they must comply with privacy regulations, including the General Data Protection Regulation (GDPR) [2]. In this context, access control policies represent a promising solution for regulating access to protected resources. Traditional approaches, such as *Identity-Based Access Control* (*IBAC*), rely on *Access Control Lists* (*ACLs*) and grant permissions based on users' identities. More flexible models, such as *Attribute-Based Access Control* (*ABAC*), determine access rights according to a set of attributes. This work was presented at the 6th Workshop on Blockchain theoRy and ApplicatIoNs (BRAIN 2025) in conjunction with PerCom 2025 and is published at [3].

2. SSI and ABAC

The transition toward decentralized systems has reinforced the separation between *Service Providers* (*SPs*) and *Identity Providers* (*IdPs*). In this context, *Self-Sovereign Identity* (*SSI*) has emerged as a user-centric paradigm that gives individuals full control over their digital identities and personal data [4]. In SSI

W3CWorkshop on the Future of ODRL 20 & 21 July 2026 London, UK.
*Corresponding author.
stefano.bistarelli@unipg.it (S. Bistarelli); chiara.luchini@collaboratori.unipg.it (C. Luchini); francesco.santini@unipg.it (F. Santini)
https://bista.sites.dmi.unipg.it/ (S. Bistarelli); https://luchinzzz.github.io/ (C. Luchini); https://francescosantini.sites.dmi.unipg.it/ (F. Santini)
0000-0001-7411-9678 (S. Bistarelli); 0009-0001-6846-0922 (C. Luchini); 0000-0002-3935-4696 (F. Santini)

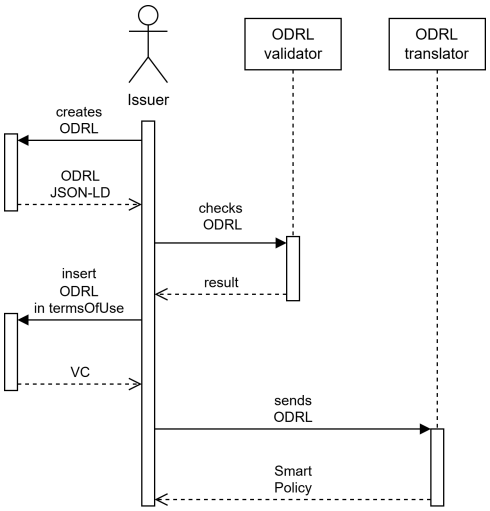

Figure 1: Translation ODRL in SP workflow.

ecosystems, users (holders) manage their own *Verifiable Credentials* (*VCs*), which are issued by trusted entities and whose authenticity can be verified through digital signatures. Collections of VCs, known as *Verifiable Presentations* (*VPs*), can be selectively shared with verifiers when required. Moreover, *Distributed Ledger Technologies* (*DLTs*) can be used to store public keys, providing a decentralized alternative to traditional identity management infrastructures [5, 6].

To regulate access to resources, *Access Control* (*AC*) systems define authorization policies based on user identities and permissions [7]. Among these approaches, *Attribute-Based Access Control* (*ABAC*) offers greater flexibility by granting access according to attributes related to users, resources, and environmental conditions, such as job roles, departments, security levels, or resource sensitivity [8, 9].

3. From ABAC-ODRL policies to Smart Contracts

ODRL is a human- and machine-readable language that can be used to define access and disclosure restrictions for resources, including *Verifiable Credentials* (*VCs*). In SSI systems, these policies can be embedded in the `termsOfUse` field, whose definition has recently been refined in the *VC Data Model v2.0* [10]. In [3], we propose defining ABAC policies in ODRL within the `termsOfUse` field and translating them into Solidity smart contracts for automatic enforcement. To this end, we developed a preliminary translator that allows issuers and holders to associate ODRL policies with smart contracts without requiring knowledge of Solidity. This approach combines the readability of ODRL with the benefits of blockchain technologies, including transparency, tamper resistance, and the absence of a single point of failure [11]. Compared to traditional approaches, ABAC offers greater flexibility by granting permissions based on attributes related to subjects, objects, or environmental conditions. Moreover, integrating ODRL semantics enables the specification of more expressive and fine-grained usage conditions, allowing complex access scenarios to be represented more effectively [12].

Figure 2 shows the workflow of SP creation. The issuer generates an ODRL document in JSON-LD format, similar to the example provided in Listing 1. Consequently, the issuer verifies the validity of the ODRL policy using an ODRL validator[1]. If the validation is successful, the issuer sends the ODRL policy to the ODRL translator, who creates a smart contract and inserts the policy into the holder's VC. More details on completing the smart contract can be found in Section 3.2.

[1]ODRL validator: https://odrlapi.appspot.com/.

3.1. ABAC policies in ODRL

The first challenge was to determine how *Attribute-Based Access Control* (*ABAC*) policies could be represented using ODRL. To this end, we analyzed the correspondence between the concepts defined in the ODRL [13] and ABAC [9] models.

In ODRL, policies are built upon three types of rules: permissions, prohibitions, and obligations, which respectively define allowed, forbidden, and mandatory actions. Each *Rule* generally targets an *Asset*, which corresponds to the ABAC notion of an *object*, namely the resource being protected. Similarly, ODRL *Action*s represent the operations that can be performed on an asset and correspond to ABAC operations such as reading, writing, or modifying a resource. ODRL also defines *Assigner* and *Assignee* entities, which are analogous to ABAC *subjects*.

Conditions are expressed through ODRL *Constraint*s, which compare operands using specific operators, and *Logical Constraint*s, which combine multiple constraints through logical operators. These mechanisms enable the definition of fine-grained authorization rules and the refinement of parties, assets, and actions, making ODRL suitable for representing expressive ABAC policies [13, 12].

```
{"@context": "http://www.w3.org/ns/odrl.jsonld",
"@type": "Offer",
"uid": "Policy:84112",
"permission": [
    {
        "uid": "P1",
        "target": "http://vc.com/document:8541",
        "assigner": "http://issuer.com/org:616",
        "action": "read",
        "constraint": [{
                "uid": "C1",
                "leftOperand": "dateTime",
                "operator": "gteq",
                "rightOperand": {
                    "value": "09:00:00",
                    "@type": "xsd:time"}},
            {
                "uid": "C2",
                "leftOperand": "dateTime",
                "operator": "lteq",
                "rightOperand": {
                    "value": "17:00:00",
                    "@type": "xsd:time"
    }}]}]}
```

Listing 1: ABAC in ODRL with environment conditions.

Listing 1 shows an example of an ABAC policy expressed in ODRL. The policy, defined by an assigner, grants read access to a document to all entities only within a specific time interval, demonstrating how environmental attributes can be used to restrict resource access. The policy includes two constraints on the same *leftOperand*, both of which must be satisfied during evaluation.

```
{
"@context": "http://www.w3.org/ns/odrl.jsonld",
"@type": "Set",
"uid": "Policy:96142",
"profile": "http://hospitalexample.com/odrl:profile:183201",
"permission": [
    {
    "uid": "P1",
    "target": "http://hospital.com/vc:patient:1",
    "action": "read",
    "assigner": "http://issuer.com/0x5B...ddC4",
    "assignee": {
        "@type": "PartyCollection",
```

```
        "partOf": {
            "source": "healthcare personnel",
            "logicalConstraints":[...],
            "refinement":[...],
}}]}
```

Listing 2: Disclosure Policy in ODRL.

Listing 2 presents an ABAC policy written in ODRL that refines the semantics of the *assignee*. The *profile* property allows the definition of domain-specific attributes that are not part of the standard ODRL vocabulary, such as healthcare-related requirements. In this example, the policy regulates access to a medical record by allowing only qualified cardiologists with more than three years of experience and a specialization in paediatrics to access the credential. This demonstrates how refinement and logical constraints can be combined to define fine-grained access conditions tailored to specific contexts.

3.2. ODRL translation in Smart Policy

This section describes the translation of an ODRL policy into a smart contract through a proof of concept (PoC). The PoC, developed in TypeScript, converts ODRL policies into Solidity smart contracts for Ethereum-based blockchains. TypeScript was chosen for its compatibility with modern web frameworks and SSI wallets, while Solidity was selected for its widespread adoption and support for access management and external data integration.

The translation focuses on ODRL constraints, which are Boolean expressions defined either directly within a rule or as refinements of *Party*, *Asset Collection*, or *Action*. In contrast, fields such as *target*, *assigner*, and *action* are treated as descriptive information and are therefore not translated into executable smart contract components, as they are assumed not to affect the execution of the *Smart Policy* (*SP*).

Figure 2: From ODRL refinement constraints into SP components.

The translation process focuses on converting ODRL policy constraints into smart contract components, targeting three main categories: subject conditions, assignee properties, and environmental

conditions. This design choice reflects the system's integration with Verifiable Credentials (VCs), where policy evaluation relies on user attributes as input. For instance, in an SSI scenario, a verifier seeking access to a patient's medical records stored in a holder's VC must satisfy role-based constraints, such as holding a valid medical credential.

The translation pipeline begins by parsing an ODRL policy expressed as a JSON document. For each rule, the translator inspects the `constraint` and `assignee` fields; when present, a dedicated function generates the corresponding Solidity code. Each constraint is mapped to an `evaluate` function that takes the `leftOperand` as input and returns a boolean indicating whether the constraint is satisfied. For example, in Figure 2, every constraint is translated into an evaluate function. This function takes the value of the *leftOperand* property as a parameter and verifies whether it meets the constraint, returning a boolean result. A true value indicates that the constraint is satisfied, while false means it is not.

To handle the heterogeneous data types supported by ODRL, the system adopts the W3C XML Schema Definition Language (XSD) [14] as a typing standard. Although XSD was designed for XML, its type definitions transfer naturally to JSON validation and provide a consistent basis for generating type-specific Solidity functions. The supported primitive types are `xsd:integer`, `xsd:boolean`, `xsd:string`, and `xsd:dateTime`.

String constraints are evaluated by comparing the hash of the `leftOperand` against the pre-computed hash of the `rightOperand`, avoiding plain-text storage on-chain. Numeric and boolean constraints apply the relational operator specified in the ODRL `operator` field (*eq*, *neq*, *gt*, *lt*, *gteq*, or *lteq*) directly between the two operands. Temporal constraints leverage Solidity's `block.timestamp` to retrieve the current on-chain time, which is then compared against the `rightOperand` converted to a UNIX timestamp. For membership constraints using the *isPartOf* operator, a set structure is emulated via a Solidity `mapping` backed by an array, since native set types are not available in the language.

Logical constraints (for instance *and* or *or*) are handled by generating a composite function that invokes all relevant single-constraint functions and combines their results according to the specified operand. When no logical constraint is declared, all individual constraints must be satisfied simultaneously.

3.3. Smart Policy Test

We evaluated the proposed PoC by deploying and testing smart contracts generated from four different ABAC-ODRL policies in Remix[2]. The considered scenarios included: a role-based policy using set membership constraints, the time-based policy of Listing 1, a simplified zkABAC use case, and the healthcare policy presented in Listing 2. In all cases, access was granted only when all constraints were satisfied.

Table 1
Gas costs of smart policies.

Example	*Deployment*	*Satisfied*	*Not Satisfied*
Set	1769328	5012	5012
Listing 1	537876	3335	2990
zkABAC	1136850	6237	6215
Listing 2	1287194	8471	8449

Table 1 reports the gas costs for contract deployment and policy evaluation. The results show that deployment costs increase with both the number of constraints and the complexity of the data types involved. For instance, compared to the zkABAC scenario with five constraints, deployment costs increased by 41% and 130% when extending the policies to 10 and 20 constraints, respectively. A possible optimization consists of grouping constraints sharing the same *leftOperand* into a single Solidity function.

[2]Remix - Ethereum IDE: https://remix.ethereum.org/.

4. Conclusion

This paper presented a methodology for defining ABAC policies using ODRL to regulate access to sensitive information, such as *Verifiable Credentials* (*VCs*), within an SSI system. Furthermore, we proposed a mechanism to translate these policies into smart contracts capable of automatically evaluating the specified constraints. This approach allows issuers to transparently define and enforce usage conditions, improving both policy transparency and the overall evaluation process.

Future work will focus on addressing policy conflicts, such as situations where permissions and prohibitions coexist, and on extending the system to support multiple data types, increasing its applicability across different domains. Another promising direction is the development of a translation tool that converts natural language policies into ODRL expressed in JSON-LD format, simplifying and accelerating the policy authoring process for issuers.

References

[1] M. Shuaib, S. Alam, M. Shabbir Alam, M. Shahnawaz Nasir, Self-sovereign identity for healthcare using blockchain, Materials Today: Proceedings 81 (2023) 203–207. International Virtual Conference on Sustainable Materials (IVCSM-2k20).

[2] European Parliament, Council of the European Union, Regulation (EU) 2016/679 of the European Parliament and of the Council, 2016. URL: https://data.europa.eu/eli/reg/2016/679/oj.

[3] S. Bistarelli, C. Luchini, F. Santini, A preliminary approach for translating ODRL to smart policies, in: IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events, PerCom 2025 - Workshops, Washington DC, USA, March 17-21, 2025, IEEE, 2025, pp. 44–49. URL: https://doi.org/10.1109/PerComWorkshops65533.2025.00039. doi:10.1109/PERCOMWORKSHOPS65533.2025.00039.

[4] A. Mühle, A. Grüner, T. Gayvoronskaya, C. Meinel, A survey on essential components of a self-sovereign identity, Comput. Sci. Rev. 30 (2018) 80–86.

[5] Q. Stokkink, J. Pouwelse, Deployment of a blockchain-based self-sovereign identity, in: iThings/-GreenCom/CPSCom/SmartData, IEEE, 2018, pp. 1336–1342.

[6] A. S. Rajasekaran, M. Azees, F. Al-Turjman, A comprehensive survey on blockchain technology, Sustainable Energy Technologies and Assessments 52 (2022) 102039.

[7] Chung, D. Ferraiolo, D. Kuhn, Assessment of Access Control Systems, NIST Interagency/Internal Report (NISTIR), National Institute of Standards and Technology, Gaithersburg, MD, 2006. doi:https://doi.org/10.6028/NIST.IR.7316.

[8] V. C. Hu, D. R. Kuhn, D. F. Ferraiolo, Attribute-based access control, Computer 48 (2015) 85–88.

[9] V. Hu, D. Ferraiolo, R. Kuhn, A. Schnitzer, K. Sandlin, R. Miller, K. Scarfone, Guide to Attribute Based Access Control (ABAC) Definition and Considerations, Technical Report, National Institute of Standards and Technology (NIST), 2014.

[10] M. Sporny, D. Longley, D. Chadwick, O. Steele, Verifiable Credentials Data Model v2.0, Technical Report, World Wide Web Consortium (W3C), 2024.

[11] V. Hu, Blockchain for Access Control Systems Share to Facebook Share to Twitter Share to LinkedIn, Technical Report, National Institute of Standards and Technology (NIST), 2021.

[12] R. Iannella, M. Steidl, S. Myles, V. Rodríguez-Doncel, ODRL Vocabulary & Expression 2.2, Technical Report, World Wide Web Consortium (W3C), 2018.

[13] R. Iannella, S. Villata, ODRL Information Model 2.2, Technical Report, World Wide Web Consortium (W3C), 2018.

[14] D. Peterson, S. S. Gao, A. Malhotra, C. M. Sperberg-McQueen, H. S. Thompson, W3C XML Schema Definition Language (XSD), Technical Report, World Wide Web Consortium (W3C), 2012.